2021-05-01

Robust Variable Selection In Multiple Linear Regression Via Penalized Least Trimmed Squares.

Reagan Kesseku
University of Texas at El Paso

ROBUST VARIABLE SELECTION IN MULTIPLE LINEAR REGRESSION VIA PENALIZED LEAST TRIMMED SQUARES

REAGAN KESSEKU

Master's Program in Statistics

APPROVED:

Michael Pokojovy, Ph.D., Chair

Ming-Ying Leung, Ph.D.

Thompson Sarkodie-Gyan, Ph.D.

Stephen L. Crites Jr., Ph.D.
Dean of the Graduate School

To my

ADVISOR Dr. Pokojovy, BROTHER Redeemer, PARENTS Gladys and Nicholas and my SISTERS Jade and Sarah

with love

ROBUST VARIABLE SELECTION IN MULTIPLE LINEAR REGRESSION VIA PENALIZED LEAST TRIMMED SQUARES

by

REAGAN KESSEKU

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Mathematical Sciences

THE UNIVERSITY OF TEXAS AT EL PASO

May 2021

Acknowledgements

"All's well that ends well". First and foremost, I give glory to Almighty God for His abundance of grace and absolute sustenance in my academic works.

Also, I express my sincere gratitude to my advisor, Dr. Michael Pokojovy of the Department of Mathematical Sciences at The University of Texas at El Paso, for his guidance throughout this work. In fact, his objective comments and tireless support contributed immensely to the success of this work and my academic life as a whole. Additionally, my heartfelt appreciation goes to my other committee members, Dr. Ming-Ying Leung of the Department of Mathematical Sciences and Dr. Thompson Sarkodie-Gyan of the Electrical and Computer Engineering Department, both at The University of Texas at El Paso, for their time and immeasurable contributions to this work.

Finally, I am thankful to my colleagues and everyone who through their moral support made this journey a memorable one.

Abstract

Variable selection has been studied using different approaches. Its growing importance lies in numerous applications to high-dimensional data from experiments and natural phenomena. Often, models are to be constructed from such data based on significant variables for estimation or prediction purposes. This demands not just any variable selection method, but one that is robust, computationally efficient and with other desirable statistical properties. Besides the high-dimensionality of such data, the presence of outliers is common due to heterogeneous sources. Though outliers often contain useful information, they can unduly influence non-robust estimators to produce misleading results. This is the case for ordinary least squares regression which is biased and inefficient under assumption violations. Many robust loss functions and penalization selection techniques have been proposed in literature. However, the choice of loss function, penalty function, optimal tuning parameter and their implementation are paramount to the robustness and efficiency of variable selection. This work proposes a penalized robust variable selection method for multiple linear regression through the least trimmed squares loss function. The proposed method employs a robust tuning parameter criterion constructed through BIC for model selection. It is implemented via a fast computation algorithm with high breakdown point which does not depend on the number of predictors in the data.

Key words and phrases: Breakdown point, penalization, outliers, robust, contamination.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Robust variable selection is a topic of high importance in modern data science and Statistics due to the increasing availability of data with large number of variables (Alfons et al., 2013). Attempts to use all variables in multivariate data often become tedious and result in over-fitting when modeling. Though over-fitting is known to provide unbiased estimates for parameters, it also produces inflated variance which inflates the mean square error (MSE) of the estimator. As such, there is no reason for all variables in a dataset to be included in a model since some are noise and others are redundant. Moreover, not only does the high-dimensionality of such data become a challenge but also, the presence of outliers (Alfons et al., 2013) and other sources of heterogeneity. Modeling based on such data to either study the relationship between the response variable and predictor variables or for prediction purposes often encounters issues like inaccurate estimation or misleading predictions. Such problems arise from model misspecification and non-robustness of variable selection techniques in the presence of severe outliers, heavy-tailed data or under other model assumption violations. For instance, the commonly used ordinary least squares (OLS) procedure becomes problematic when the errors deviate from the normal distribution or severe outliers are present in the data.

In attempt to address the above issues, other variable selection techniques have been proposed in literature referred to as penalization methods. Penalization methods known in literature include ridge regression (Hoerl and Kennard, 1970), bridge estimator (Frank and Friedman, 1993), Least Absolute Shrinkage and Selection Operator (Lasso) (Tibshirani, 1996), the Smoothly Clipped Absolute Deviation (SCAD) (Fan and Li, 2001), the elastic net or Enet (Zou and Hastie, 2005), the adaptive Lasso (Zou, 2006), the Minimax Concave

Penalty (MCP) approach (Zhang, 2010) and many others. However, most penalization methods add a penalty function to a classical loss function like that of the OLS. As such, these classical-based variable selection methods also break down when variance of the error in the regression model tends to infinity (Chen et al., 2014). According to Lambert-Lacroix et al. (2011), not only do these methods break down in the presence of outliers but also the effect of outliers is not well studied for many variable selection techniques. Researchers have therefore, proposed to use robust loss functions including Huber's loss function (Fan and Li, 2001), least absolute deviation (Wang et al., 2007; Gao and Feng, 2018; Jiang and Liao, 2020), quantile loss (Li and Zhu, 2008; Zou et al., 2008; Wu and Liu, 2009), Jaeckel's dispersion function (Johnson and Peng, 2008; Leng, 2010), exponential squared loss (Wang et al., 2013) and density power divergence function (Mandal and Ghosh, 2019) with one or more penalties like Lasso, SCAD or adaptive Lasso on the regression coefficients. Some of these existing approaches are designed to handle specific nature of the data like the penalized least-squares or penalized likelihood (Fan and Lv, 2011) which is suitable for light-tailed distributions. The choice of loss and penalty function is paramount for both estimation and variable selection accuracy. Robust variable selection methods that can withstand the effect of assumption violations, produce sparse estimates and enjoy the statistical oracle properties are important to model and analyze such data. This work therefore, proposes a *robust variable selection for multiple linear regression via penalized least trimmed squares*.

The rest of the thesis is organized as follows. Chapter 2 gives a literature review on some variable selection methods, loss functions, outliers and penalty functions. The proposed methodology for robust variable selection is presented in Chapter 3 including the algorithm for its computation. A simulation study and a real data analysis are respectively presented in Chapter 4 and Chapter 5 to explore the effectiveness of the proposed method. Finally, Chapter 6 gives the discussion and summary of results.

Chapter 2

Literature Review

This chapter discusses and identifies the gap in robust variable selection methods in known literature. It covers loss functions, outliers, breakdown point and also provide insight on methods of penalization.

2.1 Variable Selection Methods

New variables often emerge during the study of natural or experimental phenomena. This has created interest to study and understand such new variables either independently or in relation to existing variables. The latter case often produces high-dimensional data when used as predictors as they add to existing predictor variables in some data. Also, high-dimensional data occur naturally in studies like genome sequencing. In fields like ecology, medicine and biology, most data contain large number of predictor variables where the predictors sometimes far exceed the number of observations. Nonetheless, since some predictors may not be significantly related to the response, there is the need to select relevant ones. As a result, the choice of predictors in linear regression has attracted considerable attention in literature (Brown et al., 1999) and this has become necessary due to the high-dimensionality of most data. In practice, the best model is hidden in a subset of all variables and this is particularly true when there are many predictor variables. According to Alfons et al. (2013), not only does the high dimension of such data become a challenge but also the presence of outliers. The presence of outliers is also contributed by other sources of heterogeneity of the data.

2.2 Outliers

One way of achieving a coherent data analysis from large number of variables during data collection is the detection of outlying observations (Ben-Gal, 2005). Outlier detection is a basic step in many statistical analyses often performed when exploratory the data. According to Ben-Gal (2005), an exact definition of outlier often depends on hidden assumptions regarding the data structure and the applied detection method. However, some definitions are considered general enough to cope with various types of data and methods. Enderlein (1987) defines outlier as an observation which deviates from other observations so significantly to arouse suspicions that it was generated by a different mechanism. Also, Gladitz (1988) indicates that an outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs. Outliers often carry relevant information despite being regarded as errors. Detected outliers are candidates for aberrant data that may otherwise adversely lead to model misspecification, biased parameter estimation and incorrect results (Ben-Gal, 2005). Chatterjee and Hadi (2009) report that the presence of outliers can lead to biased parameter estimation and inappropriate predictions for classical methods. In fact, outliers can mislead regression results as they pull the regression line towards themselves and this can result in a solution that is more precise for the outlier and imprecise for other cases in the data set (Ben-Gal, 2005). It is therefore important to identify them prior to modeling and analysis (Williams et al., 2002; Liu et al., 2004).

Outliers can occur in either the response variable, predictor variables or both (Chatterjee and Hadi, 2009). Real dataset may contain about 1% to 10% outliers (Hampel et al., 2011). That notwithstanding, detection of outliers is dimensional related. That is, it can either be *univariate* or *multivariate* outlier detection. Appropriate methods are designed to handle them. Another fundamental taxonomy of outlier detection methods is between *parametric* methods and *non-parametric* methods that are model-free (e.g., see (Williams et al., 2002)). Statistical parametric methods assume a known underlying distribution of the observations

(e.g. (Hawkins, 1980; Leroy and Rousseeuw, 1987; Barnett and Lewis, 1984)) or, at least, they are based on statistical estimates of unknown distribution parameters (Hadi, 1992; Caussinus and Ruiz, 1990). Observations that deviate from the model assumptions are flagged as outliers by these methods. They are often unsuitable for high-dimensional data sets and for arbitrary data sets without prior knowledge of the underlying data distribution (Papadimitriou et al., 2003).

2.2.1 Univariate Outlier Detection Methods

Most of the earlier univariate methods including current ones for outlier detection rely on the assumption of an underlying known distribution of the data, which is assumed to be identically and independently distributed. Moreover, many discordance tests for detecting univariate outliers further assume that the distribution parameters and the type of expected outliers are also known (Barnett and Lewis, 1984). Needless to say, in real world data mining applications, these assumptions are often violated (Ben-Gal, 2005).
Tukey et al. (1977) introduced the "boxplot" as a graphical display on which outliers can be indicated. The boxplot, which is being extensively used up to date, is based on the distribution quadrants. Liu et al. (2004) also proposed an outlier-resistant data filter-cleaner based on the earlier work of (Martin and Thomson, 1982). The proposed data filter-cleaner includes an on-line outlier-resistant estimate of the process model and combines it with a modified Kalman filter to detect and "clean" outliers which does not require an apriori knowledge of the process model.

2.2.2 Multivariate Outlier Detection Methods

In many cases, multivariate observations can not be detected as outliers when each variable is considered independently. Outlier detection is possible only when multivariate analysis is performed, and the interactions among different variables are compared within the class of data (Ben-Gal, 2005). Data sets containing multiple outliers are subject to *masking*

and *swamping* effects. The concept of masking (Acuna and Rodriguez, 2004; Hawkins, 1980; Iglewicz and Martinez, 1982; Davies and Gather, 1993; Barnett and Lewis, 1984) and swamping (Iglewicz and Martinez, 1982) are well discussed by these authors. The Mahalanobis distance is one well-known criterion for multivariate outlier detection which depends on estimated parameters of the multivariate distribution. However, this technique uses the mean and variance-covariance matrix which are all influenced by outliers and hence, not robust. Many researchers have therefore, proposed robust estimators of the variance-covariance matrix or used other efficient techniques. Caussinus and Ruiz (1990) proposed a robust estimate for the covariance matrix based on weighted observations according to their distance from the center. The authors also proposed a method for low-dimensional projections of the dataset. They used the Generalized Principal Component Analysis to reveal dimensions which display outliers. Other robust location and scatter (covariance matrix) estimators include minimum covariance determinant (MCD) and the minimum volume ellipsoid (MVE) (Rousseeuw, 1985; Leroy and Rousseeuw, 1987; Acuna and Rodriguez, 2004).

2.3 Statistical Setting

Suppose the pair $(\boldsymbol{x}_i, y_i)$ denotes the measurement from the i-th experimental unit, where $y_i \in \mathbb{R}$ is the response variable and $\boldsymbol{x}_i \in \mathbb{R}^p$ is the vector of predictors with $\boldsymbol{x}_i$ typically modified to include the model intercept. Consider the following linear regression model:

$$y_i = \boldsymbol{x}_i^T \boldsymbol{\beta} + \varepsilon_i \qquad \text{for} \quad i = 1, 2, \ldots, n \ \text{ such that } n > p \tag{2.1}$$

where $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^T$ are the regression coefficients, $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^T$ is the response vector, $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)^T$ is the design matrix and ε_i are the random errors. For *non-contaminated* data, assume $\varepsilon_i \overset{\text{i.i.d}}{\sim} \mathcal{N}(0, \sigma^2)$ and that $y_i \sim \mathcal{N}(\boldsymbol{x}_i^T \boldsymbol{\beta}, \sigma^2)$. Conversely, ε_i are independently distributed with unknown distribution, say, $\mathcal{G}$, where $\mathcal{G}$ is assumed to be symmetric about 0 as it is often the case in literature, see (Wang et al., 2013).

2.4 Classical Variable Selection Methods

OLS is a basic and commonly used variable selection method which minimizes the sum of squared error of the residuals. Under the classical setup when the dimension is low ($n \gg p$), the OLS estimate of $\boldsymbol{\beta}$ is obtained by minimizing the squared error loss function $\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2$, where $\|\cdot\|$ is the standard ℓ_2 norm. The OLS estimator is unbiased, but for small to moderate sample sizes, it often exhibits large variance. It is easy to see that this estimator is not robust, because single observation can change the estimator to any desired value. When there are outliers in the response or influential points in the predictors, which the latter can be encountered in, for example microarray gene expression studies, regular procedures are not appropriate (Wu and Ma, 2015). Therefore, OLS has both a low breakdown point and unbounded influence function, however, with the advantage of having the highest possible efficiency when the error distribution function, F, is Gaussian (Coakley and Hettmansperger, 1993). One of the goals of robust regression estimation, according to Yohai and Zamar (1988), is to simultaneously achieve (a) a breakdown point of roughly 0.50; (b) a bounded influence function; and (c) a high efficiency (say, 0.95) versus least squares when F is Gaussian. Since possible data contamination and model misspecification can affect the result of most methods, the development of robust modeling must be sought. Under data contamination, the idea is to down weigh the influence of outliers while model misspecification tends to build a loss function that can accommodate a class of models (as opposed to a single specific one) (Wu and Ma, 2015).

2.5 Breakdown Point

One global measure of an estimator's robustness is the breakdown value or point (Maronna et al., 2019). The proportion of outliers or contamination in a data affects the breakdown point of estimators in variable selection; a very essential component for estimators to achieve robustness. Hampel (1971, 1974) introduced the concept of breakdown point as a measure

for robustness of an estimator against outliers. The breakdown point of an estimator is roughly, the smallest amount of contamination that may cause it to take on arbitrarily large aberrant values (Donoho and Huber, 1983). The range of possible breakdown value is between $\frac{1}{n}$ and $\frac{1}{2}$. So ideally, the higher the breakdown value, the more robust a regression approach or estimator is. Donoho and Huber (1983) further report that, breakdown point provides only a very crude quantification of the robustness properties of an estimator in which lies its generality and strength. Many methods have been developed to detect outliers in the multiple linear regression (Gürünlü Alma et al., 2011; Riani et al., 2009; Nurunnabi et al., 2014; Millar and Hamilton, 1999; Jobe and Pokojovy, 2015; Imon and Hadi, 2013). These and other robust methods are important and effective estimation techniques for analyzing high-dimensional data contaminated with outliers.

2.6 Robust Loss Functions

One means of achieving robust modeling is through the development of a robust loss function say, $L(\boldsymbol{\beta}; \boldsymbol{y}, \boldsymbol{x})$ with $\boldsymbol{y}$ and $\boldsymbol{x}$ as defined in Equation (2.1). According to Wu and Ma (2015), a robust loss function differs from an ordinary loss function by its ability to accommodate irregularities in data and model settings. Under asymptotic settings, the loss function produces consistent estimate of the unknown regression parameter when minimized. As such, the construction of loss function has no strict rules but with the basic requirement to possess consistency (Wu and Ma, 2015). Now, when consistency is available, statistical and computational efficiency is also of interest. The following subsections discuss some well known loss functions in literature.

2.6.1 Check Loss Function

Perhaps the most extensively examined robust loss function is the check loss function in quantile regression (QR). This idea was proposed by (Koenker, 2005). Define $\rho_\tau(t) = t\{\tau - \mathbb{I}(t < 0)\}$ at any given quantile level $0 < \tau < 1$. With such choice, the loss function

for the i-th subject becomes

$$\rho_\tau(y_i - \boldsymbol{x}_i^T\boldsymbol{\beta}) = \begin{cases} \tau(y_i - \boldsymbol{x}_i^T\boldsymbol{\beta}), & \text{if } y_i - \boldsymbol{x}_i^T\boldsymbol{\beta} > 0, \\ -(1-\tau)(y_i - \boldsymbol{x}_i^T\boldsymbol{\beta}), & \text{otherwise.} \end{cases} \tag{2.2}$$

The overall loss function is defined as $\sum_{i=1}^n \rho(y_i - \boldsymbol{x}_i^T\boldsymbol{\beta})$. By emphasizing more on the relative rank as opposed to absolute magnitude, this loss function is able to 'tolerate' outliers and influential points to a much greater extent than the OLS. The unique advantage of robust procedures built on QR lies in the ability to capture the heterogeneity of data through different quantiles. For example, with the presence of data heterogeneity, different sets of important genes can be associated with the response at different quantiles (Wu and Ma, 2015). Quantile-based regression aims to estimate the conditional "quantile" of a response variable given certain values of predictor variables. Researchers have extended the QR approach to multiple dimension. The check loss function has been adopted in multiple studies (Zou and Yuan, 2008; Zou et al., 2008) with multiple conditional quantile functions estimated simultaneously.

2.6.2 LAD Loss Function

The least absolute deviation (LAD) regression is known to be robust when there are outliers in the response variable or for heavy-tailed error data. The LAD loss function is defined as follows:

$$\sum_{i=1}^n |y_i - \boldsymbol{x}_i^T\boldsymbol{\beta}|. \tag{2.3}$$

This loss function have been used in many studies such as (Wang et al., 2007; Gao and Huang, 2010; Wang et al., 2013; Jiang et al., 2021). LAD regression provides estimate for the conditional median function. It is the foundation for the development of QR and can be viewed as a special case of QR. Despite the robustness of the LAD loss function, Lambert-Lacroix et al. (2011) pointed out that this criterion suffers a loss of efficiency with normally distributed data.

2.6.3 Huber's Loss Function

The Huber's loss function is a loss function used in robust regression that is less sensitive to outliers in data than the squared error loss. The Huber's loss function describes the penalty incurred by an estimation procedure f. Huber (1992) defines the loss function piecewise by

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2, & \text{for } |a| \leq \delta, \\ \delta(|a| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases} \tag{2.4}$$

This function is quadratic for small values of a, and linear for large values, with equal values and slopes of the different sections at the two points where $|a| = \delta$. The Huber's loss function is strongly convex in a uniform neighborhood of its minimum $a = 0$; at the boundary of this uniform neighborhood, the Huber's loss function has a differentiable extension to an affine function at points $a = -\delta$ and $a = \delta$. These properties allow it to combine much of the sensitivity of the mean-unbiased, minimum-variance estimator of the mean (using the quadratic loss function) and the robustness of the median-unbiased estimator (using the absolute value function). The variable a often refers to the residuals, that is to the difference between the observed and predicted values $a = y - f(x)$, so the former according to Hastie et al. (2009), can be expanded to

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2, & \text{for } |y - f(x)| \leq \delta, \\ \delta(|y - f(x)| - \frac{1}{2}\delta^2), & \text{otherwise.} \end{cases} \tag{2.5}$$

Pseudo-Huber loss function is another approach that can be used as smooth approximation of the Huber's loss function. It combines the best properties of ℓ_2 squared loss and ℓ_1 absolute loss by being strongly convex when close to the target or minimum and less steep for extreme values. The scale at which the pseudo-Huber loss function transits from ℓ_2 loss for values close to the minimum to ℓ_1 loss for extreme values and the steepness at extreme values can be controlled by the δ value. The Pseudo-Huber loss function ensures that derivatives are continuous for all degrees defined according to Charbonnier et al. (1997);

Schmid and Zisserman (2000) as

$$L_\delta(a) = \delta^2\Big(\sqrt{1+(a/\delta)^2} - 1\Big) \tag{2.6}$$

As such, this function approximates $\frac{a}{2}$ for small values of a, and approximates a straight line with slope δ for large values of a. While the above is the most common form, other smooth approximations of the Huber's loss function also exist (Lange, 1990). Sometimes a variant for classification is also used.

2.6.4 Rank-Based Loss Function

The rank-based loss function is another type of robust procedures in regression. Jaeckel (1972) was among the first authors to propose rank-based regression as a robust and non-parametric alternative to classical OLS and likelihood-based approaches. The dispersion function proposed by Jaeckel (1972) has been adopted by Johnson and Peng (2008); Leng (2010) as a loss function

$$\sum_{i=1}^{n} \varepsilon\left[\frac{R(\varepsilon_i)}{n+1} - \frac{1}{2}\right]\varepsilon_i \tag{2.7}$$

where $\varepsilon_i = y_i - \boldsymbol{x}_i^T\boldsymbol{\beta}$, $\varepsilon(\cdot)$ is a non-decreasing weight function and $R(\varepsilon_i)$ is the rank of ε_i in $\{\varepsilon_i, \ldots \varepsilon_n\}$. Equation (2.7) reduces to the regular Wilcoxon statistic when $\varepsilon(\cdot)$ is the identity function. In an independent work, Wang and Li (2009) investigated the weighted Wilcoxon loss function

$$\frac{1}{n}\sum_{i<j} b_{ij}|\varepsilon_i - \varepsilon_j| \tag{2.8}$$

where b'_{ij}s are the positive and symmetric weights. They pointed out that when b'_{ij}s are constants, the minimization of Equation (2.8) is equivalent to that of Equation (2.7) with the identity weight function.

2.6.5 Density Power Divergence Function

Mandal and Ghosh (2019) proposed the density power divergence (DPD) function between the model density $f_{\boldsymbol{\theta}}$ with $\boldsymbol{\theta} \in \Theta$ and the empirical or true density g given by

$$d_\alpha(f_{\boldsymbol{\theta}}, g) = \begin{cases} \int_y \{f_{\boldsymbol{\theta}}^{1+\alpha}(y) - (1+\frac{1}{\alpha}) f_{\boldsymbol{\theta}}^{\alpha} g(y) + \frac{1}{\alpha} g^{1+\alpha}(y)\} dy, & \text{for } \alpha > 0 \\ \int_y g(y) \log\left(\frac{g(y)}{f_{\boldsymbol{\theta}}(y)}\right) dy, & \text{for } \alpha = 0 \end{cases} \tag{2.9}$$

where α is a tuning parameter (Basu et al., 1998). For $\alpha = 0$, the DPD is obtained as a limiting case of $\alpha \to 0^+$; and the measure reduces to Kullback-Leibler divergence. For a parametric model, Mandal and Ghosh (2019) estimated $\boldsymbol{\theta}$ by minimizing the DPD measure with respect to $\boldsymbol{\theta}$ over its parametric space and referred to the estimator as the minimum density power divergence estimator (MDPDE). For $\alpha = 0$, they found it to be equivalent to maximizing the log-likelihood function and concluded that the MLE is a special case of the MDPDE. Moreover, the tuning parameter α controls the trade-off between efficiency and robustness of the MDPDE – robustness measure increases if α increases, but at the same time efficiency decreases.

2.6.6 Least Trimmed Squares

Least trimmed squares (LTS) also known as least trimmed sum of squares is one of the number of methods for robust regression. It is a robust statistical method that fits a function to a set of data whilst not being unduly affected by the presence of outliers. Instead of the standard least squares method, which minimizes the sum of squared residuals over n points, the LTS method attempt to minimize the sum of squared residuals over a subset, say, k of those points where $k < n$ and the $(n - k)$ unused points do not influence the fit. This method has no closed-form solution due to its binary nature for data points as points are either included or excluded. As a result, methods for finding the LTS solution sift through combinations of the data, attempting to find the k subset that yields the lowest sum of squared residuals. Methods exist for finding the exact solution at low n; however, as n increases, the number of combinations grow rapidly yielding methods that provide

approximate but generally sufficient solutions. Jung (2007) proposed Orthogonal Least Trimmed Squares estimator which has high breakdown point and appropriate equivariance properties.

2.7 Penalization Methods

Intuitively, it has been shown that shrinking or setting some regression coefficients to zero improves the mean squared error and that is what the penalization methods do. Variable selection and estimation of $\boldsymbol{\beta}$ in Equation (2.1) are achieved simultaneously by computing $\hat{\boldsymbol{\beta}}$, the estimate of $\boldsymbol{\beta}$ under subjected penalty constraints which minimizes the penalized loss function. Under penalization, small bias is incorporated into a loss function in the form of a penalty to achieve greater reduction in the variance term which then improves the mean squared error. Ridge regression is an example of such penalization methods classified as a special case of Tikhonov regularization. Ridge regression is particularly useful for linear regression of data that has multicollinearity and usually large numbers of parameters. In general, the method provides improved efficiency in parameter estimation problems in exchange for a tolerable amount of bias (Gruber, 1998). A penalized variable selection method uses a regularized parameter in the penalty function which controls the model complexity. The general framework is

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^{p+1}}{\arg\min} \left[L(\boldsymbol{\beta}; \boldsymbol{y}, \boldsymbol{x}) + P_\lambda(\boldsymbol{\beta})\right] \tag{2.10}$$

where $L(\boldsymbol{\beta}; \boldsymbol{y}, \boldsymbol{x})$ is the loss function and $P_\lambda(\boldsymbol{\beta})$ is the penalty function depending on the tuning parameter $\lambda \in (0, \infty)$. Here, $P_\lambda(\boldsymbol{\beta})$ controls the model complexity by forcing some components of $\hat{\boldsymbol{\beta}}$ to exactly zero. For many penalties, $P_\lambda(\boldsymbol{\beta}) = \lambda P(\boldsymbol{\beta})$. This shows how λ balances the model complexity and the goodness of fit at least on training data. Thus, more predictors are included as $\lambda \to 0$, which leads to models with better goodness of fit. However, prediction performance and interpretability can be sacrificed under more complex models. On the other hand, fewer predictors remain in the model as $\lambda \to \infty$. Now, with

a properly tuned λ, satisfactory prediction performance and model interpretability can be achieved. A similar pattern follows when $P_\lambda(\cdot)$ takes other formats.

A commonly used method to select the tuning parameter is the cross-validation technique where the model parameters are estimated from the training data, and then the regularized parameter is selected from the remaining test data (Golub et al., 1979). The generalized cross validation (GCV) criterion is often used to compute the optimal λ especially in ridge regression and Lasso. That notwithstanding, both estimation and inference may be severely affected if there are outliers in the data. Therefore, the classical cross-validation technique may not work properly in the presence of outliers. Besides, the cross-validation technique is computationally intensive. For the same reason, the classical bootstrap-based methods may also fail in the presence of outliers.

Another widely used technique is the information based criteria for the model selection. The Mallows' C_p statistic (Mallows, 1973), the Akaike information criterion (AIC) (Beatty et al., 2018) and the Bayes information criterion (BIC) (Schwarz et al., 1978) play an important role in high-dimensional data analysis. Unfortunately, as most selection criteria are developed based on the ordinary least squares estimates, their performance under heavy-tailed errors is very poor. Ronchetti (1985), Ronchetti and Staudte (1994) modified the classical selection criteria using the Huber's M-estimator. Consequently, Hurvich and Tsai (1990) derived a set of useful model selection criteria based on the LAD estimates. Despite their usefulness, these LAD-based variable selection criteria, have some limitations with the major one being the computational burden (Wang et al., 2007).

2.7.1 Penalty Functions

Penalization, also referred to as regularization, works by biasing parameter estimator towards particular values (such as small values near zero) by the use of a tuning constant. Many of such penalty terms added to the classical methods have been proposed with few major ones captured below.

ℓ_1 **regularization:** This approach adds a penalty equal to the sum of the absolute values of the more prominent coefficients. It usually yields sparse models (models with few coefficients) since some coefficients become zero and, thus, are eliminated. Example of regularization method that uses this penalty function is that of the Lasso.

ℓ_2 **regularization:** This method adds a penalty equal to the sum of squares of the magnitude of the coefficients. In this context, sparse models are not produced since coefficients only get very close to zero but unlikely to vanish. Ridge regression and Support Vector Machines are known methods that use this approach.

A combination of both ℓ_1 and ℓ_2 regularization is also possible which leads to the elastic net. However, it achieves that by introducing additional hyperparameter whose value define the form of regularization applied.

2.7.2 Lasso

Actually, in the sense of Lagrange's necessary optimality conditions, the Lasso regression (Tibshirani, 1996) is merely a constrained variant of least squares regression. Comparatively, the ridge regression makes the selection process continuous by varying a shrinkage parameter which makes it more stable. On the other hand, since ridge regression does not set any coefficients to 0, it does not give an easy interpretable model as in subset selection. The Lasso technique is intended to balance and retain the favorable features of both subset selection and ridge regression by shrinking some coefficients and setting others to 0. The Lasso estimator of $\boldsymbol{\beta}$ is obtained by

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^{p+1}}{\arg\min} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda \sum_{i=1}^{p} |\beta_i| \ \text{ s.t } \ \sum_{i=1}^{p} |\beta_i| \ \leq t \ \text{ for } \ t > 0. \tag{2.11}$$

In this method, the estimates are the same as the least square estimates when t is chosen to be greater than or equal to $\sum_{i=1}^{p} |\beta_i|$. Conversely, choosing t smaller than $\sum_{i=1}^{p} |\beta_i|$ shrinks the solutions towards 0.

2.7.3 Enet

The elastic net proposed by Zou and Hastie (2003) considers for fixed non-negative λ_1 and λ_2 the objective function

$$L(\lambda_1, \lambda_2, \boldsymbol{\beta}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda_2\|\boldsymbol{\beta}\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1 \tag{2.12}$$

yielding the minimizer for $\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^{p+1}}{\arg\min}\, L(\lambda_1, \lambda_2, \boldsymbol{\beta})$. This procedure can be viewed as a constrained ridge regression. Here, letting $\alpha = \frac{\lambda_2}{\lambda_1+\lambda_2}$ and solving for $\boldsymbol{\beta}$ in Equation (2.12) is equivalent to the constrained optimization problem

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^{p+1}}{\arg\min}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 \quad \text{subject to} \quad \alpha\|\boldsymbol{\beta}\|_1 + (1-\alpha)\|\boldsymbol{\beta}\|_2^2 \leq t \ \text{ for some } \ t > 0. \tag{2.13}$$

The function $\alpha\|\boldsymbol{\beta}\|_1 + (1-\alpha)\|\boldsymbol{\beta}\|_2^2$ is called the *"elastic net penalty"* which is a convex combination of the Lasso and ridge penalties. Now, when $\alpha = 0$, the elastic net simply becomes a ridge regression and reduces to LASSO regression when $\alpha = 1$.

2.7.4 Alasso

Zou (2006) proposed the Adaptive Lasso (Alasso) penalty to improve the performance of Lasso. Consider the Lasso with penalty weights ($\boldsymbol{w}$), where $\boldsymbol{w}$ is a known weights vector. Zou (2006) showed that if the weights are data-dependent and suitably chosen, the weighted Lasso can possess the so-called oracle property. That is, the asymptotic consistency (almost sure) in correct variable selection with large samples when selecting the penalty parameter $\lambda \equiv \lambda(n)$ appropriately. The methodology is defined as follows:
Suppose $\hat{\boldsymbol{\beta}}$ is an $\sqrt{n}$-consistent estimator of $\boldsymbol{\beta}$. Pick $\gamma > 0$, where often $\gamma = 1$ is chosen and define the weight vector $\hat{\boldsymbol{w}}$ componentwise as $\hat{w}_i = |\hat{\beta}_i|^{-\gamma}$. The Alasso estimator $\hat{\boldsymbol{\beta}}$ is given by

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^{p+1}}{\arg\min}\left(\left\| \boldsymbol{y} - \sum_{i=1}^{p} \beta_i x_i \right\|^2 + \lambda \sum_{i=1}^{p} w_i|\beta_i| \right). \tag{2.14}$$

It is worth emphasizing that Equation (2.14) is a convex optimization problem and its global minimizer can be efficiently computed.

2.7.5 SCAD and MCP

The statistical properties of Lasso have been explored in depth. It has been found that ℓ_1 regularization may not provide satisfactory variable selection, especially when predictors are highly correlated. Moreover, stringent conditions are needed in order for Lasso to enjoy oracle properties. From the perspective of sparse estimation, a desirable penalty function is expected to help achieve three key goals as spelled out in Fan and Li (2001): (1) unbiasedness in estimating nonzero parameters, (2) sparsity in terms of enforcing zero estimates, and (3) continuity in terms of the model spectrum under consideration. This motivated them to propose the SCAD non-convex penalty. Later on, Zhang (2010) proposed a similarly shaped penalty called MCP. Both SCAD and MCP yield very similar empirical performance in terms of sparse estimation, leading to substantial improvement over Lasso. Both penalties are even functions. For $\beta > 0$, the SCAD penalty function is given by

$$w_{a,b}(\beta) = \begin{cases} \alpha\beta & \text{if } \beta \leq a, \\ \frac{2ab\beta - a^2 - \beta^2}{2(b-1)} & \text{if } a < \beta \leq ab, \\ a^2(b+1)/2 & \text{if } \beta > ab \end{cases} \tag{2.15}$$

with the first derivative $\frac{\partial}{\partial\beta} w_{a,b}(\beta) = a\{\mathbb{I}(\beta \leq a) + \frac{(ab-\beta)}{a(b-1)} + \mathbb{I}(\beta > a)\}$, for $a \geq 0$ and $b > 2$. It corresponds to a quadratic spline function with knots at a and ab.

For $\beta > 0$, the MCP penalty is given by

$$w_{a,b}(\beta) = \begin{cases} a\beta - \frac{\beta^2}{2b} & \text{if } \beta \leq ab, \\ \frac{ba^2}{2} & \text{if } \beta \geq ab, \end{cases} \tag{2.16}$$

with first derivative $\frac{\partial}{\partial\beta} w_{a,b}(\beta) = (a - \frac{\beta}{b})(\beta \leq \{\mathbb{I}(\beta \leq a) + \frac{(ab-\beta)}{a(b-1)} + \mathbb{I}(\beta > a)\}$, for $a \geq 0$ and $b > 1$. Both SCAD and MCP penalties are smooth in $\beta \geq 0$ with singularity at $\beta = 0$.

The SCAD or MCP estimator is given as

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^{p+1}}{\arg\min} \Big(\| \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} \|^2 + \lambda \sum_{i=1}^{p} w_{a,b}(\beta_i) \Big). \tag{2.17}$$

The oracle properties of $\hat{\boldsymbol{\beta}}$, which imply both consistency in variable selection and efficiency in estimating non-zero coefficients have been established.

Chapter 3

Proposed Method

We consider the linear regression model in Equation (2.12) for some given data $(\boldsymbol{x_i}, y_i)$ with $\boldsymbol{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$ and propose the Robust Penalized Least Trimmed Squares (RPLTS) method that minimizes the penalized objective function

$$(\hat{\boldsymbol{\beta}}_{\text{rob}}, \hat{\boldsymbol{w}}) = \underset{\substack{\boldsymbol{\beta}\in\mathbb{R}^{p+1} \\ \boldsymbol{w}\in\{0,1\}^n:\sum_{i=1}^n w_i=h}}{\arg\min} \left\{ \sum_{i=1}^{n} w_i(y_i - \boldsymbol{\beta}^T x_i)^2 + \lambda P(\boldsymbol{\beta}) \right\} \tag{3.1}$$

where $\boldsymbol{\beta} \in \mathbb{R}^{p+1}$, $\lambda P(\boldsymbol{\beta})$ is a penalty term which shrinks regression coefficients towards zero. Here, the penalty function can be Lasso, SCAD or MCP. For $1 \leq i \leq n$, the weights w_i represents the heterogeneity of the errors. A good observation is assigned a value of 1 and an outlier a value of 0. That is, $\boldsymbol{w} \in \{0,1\}^n$ represents the weights quantifying the outlying effect of each observation with $\hat{\boldsymbol{w}}$ the minimizer for $\boldsymbol{w}$. Moreover, h denotes the total size of good observations which is simply the sum of all 1's in $\hat{\boldsymbol{w}}$ with $\frac{n+1}{2} \leq h \leq n$. We define the residual estimates as

$$\hat{\varepsilon}_i = y_i - \hat{\boldsymbol{\beta}}_{\text{rob}}^T \boldsymbol{x}_i \;\text{ for }\; i = 1, 2, \ldots, n \tag{3.2}$$

and compute the degrees of freedom $(\hat{\nu})$ as the number of non-zero regression coefficients at the optimal tuning parameter. Clearly, $\hat{\nu}$ takes values from 0 to p. We further estimate the error scale σ via

$$\hat{\sigma}^2_{\text{rob}} = \frac{c^2_{n,h,\hat{\nu}}}{h - \hat{\nu}} \sum_{i=1}^{n} \hat{w}_i \hat{\varepsilon}_i^2 \tag{3.3}$$

According to Croux and Haesbroeck (1999), $c^2_{n,h,\hat{\nu}}$ in Equation (3.3) is given by

$$c^2_{n,h,\hat{\nu}} = \frac{1}{\int_{-\xi}^{\xi} x^2 \phi(x)\, dx} = \frac{1}{P(\chi^2_3 \leq \chi^2_{1,1-\alpha})} \tag{3.4}$$

with $\phi(\cdot)$ being the standard Gaussian density and $\Phi_0^{-1}(\cdot)$ the standard Gaussian cumulative distribution function. Also, $\xi = \Phi_0^{-1}(1-\frac{\alpha}{2})$ where $\alpha = 1 - \frac{h-\hat{\nu}}{n-\hat{\nu}}$ and that simplifies to $\xi = \Phi_0^{-1}(1-\frac{n-h}{2(n-\hat{\nu})})$ by substituting the value of α. The estimator $c^2_{n,h,\hat{\nu}}$ was chosen to assure the asymptotic unbiasedness of $\hat{\sigma}^2_{\text{rob}}$ after trimming with the weights.

By simply writing $c^2_{n,h,\hat{\nu}}$ as c^2, Equation (3.3) can be written as

$$\hat{\sigma}^2_{\text{rob}}(h-\hat{\nu}) = c^2 \sum_{i=1}^{n} \hat{w}_i \hat{\varepsilon}_i^2 \tag{3.5}$$

Under the classical setting,

$$\begin{aligned} \log L(\boldsymbol{\theta}|\boldsymbol{y},\boldsymbol{X}) &= \sum_{i=1}^{n} \log \phi(y_i, \boldsymbol{x}_i|\boldsymbol{\theta}) \quad \text{where} \quad \boldsymbol{\theta} = (\boldsymbol{\beta}, \sigma^2) \\ &= \sum_{i=1}^{n} \log \left[\frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} \exp\Big(-\frac{1}{2\sigma^2}(y_i - \boldsymbol{\beta}^T\boldsymbol{x}_i)^2 \Big) \right] \\ &= -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \boldsymbol{\beta}^T\boldsymbol{x}_i)^2. \end{aligned} \tag{3.6}$$

For the trimmed log-likelihood (naïve version), we have

$$\begin{aligned} \log L_{\text{tr}}(\boldsymbol{\theta}|\boldsymbol{y},\boldsymbol{X}) &= \sum_{i=1}^{n} w_i \log \phi(y_i, \boldsymbol{x}_i|\boldsymbol{\theta}) \\ &= -\frac{h}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n} w_i(y_i - \boldsymbol{\beta}^T\boldsymbol{x}_i)^2. \end{aligned} \tag{3.7}$$

Also, for the trimmed log-likelihood (asymptotically unbiased version)

$$\log L_{\text{tr}}(\boldsymbol{\theta}|\boldsymbol{y},\boldsymbol{X}) = -\left[\frac{h}{2}\log(2\pi\sigma^2) - \frac{c^2}{2\sigma^2}\sum_{i=1}^{n} w_i(y_i - \boldsymbol{\beta}^T\boldsymbol{x}_i)^2 \right] \tag{3.8}$$

Plugging Equation (3.1) and (3.2) into Equation (3.5) gives

$$\begin{aligned} \log L_{\text{tr}}(\hat{\boldsymbol{\theta}}_{\text{rob}}|\boldsymbol{y},\boldsymbol{X}) &= \frac{n}{h}\left[-\frac{h}{2}\log(2\pi\hat{\sigma}^2_{\text{rob}}) - \frac{c^2}{2\hat{\sigma}^2_{\text{rob}}}\sum_{i=1}^{n}\hat{w}_i(y_i - \hat{\boldsymbol{\beta}}^T_{\text{rob}}\boldsymbol{x}_i)^2 \right] \\ &= \frac{n}{h}\left[-\frac{h}{2}\log(2\pi\hat{\sigma}^2_{\text{rob}}) - \frac{1}{2}(h-\hat{\nu}) \right] \\ &= -\frac{n}{2}\left[\log(2\pi\hat{\sigma}^2_{\text{rob}}) + 1 - \frac{\hat{\nu}}{h} \right]. \end{aligned} \tag{3.9}$$

3.1 Optimal Tuning Parameter Selection

The most common approach to the comparison and selection of statistical models is standard significance testing of nested models since such test are theoretically well understood and almost universally established in quantitative data analysis (Kuha, 2004). However, the author further opines that for large sample size, significance tests are sensitive to quite small deviations from the null hypothesis and that all reasonably parsimonious models may be rejected as having a statistically significant lack of fit which is an undesirable property. As such, one common alternative has been the use of so-called penalized model selection criteria. Thus, three penalized tuning parameter selection criteria were developed based on the Bayesian Information Criterion (BIC), Akaike Information Criterion (AIC) and Mallows' C_p using estimators from the proposed method. This was necessary to see which criterion provides the optimal tuning parameter under the same condition.

3.1.1 Robust BIC for Tuning Parameter Selection

Bayesian information criterion (BIC) or Schwarz information criterion is a criterion for model selection among a finite set of models whereby the model with the lowest BIC value is preferred. It is based, in part, on the likelihood function and it is closely related to the Akaike information criterion (AIC). When fitting models, it is possible to increase the likelihood by adding parameters, but doing so may result in overfitting. Both BIC and AIC attempt to resolve this problem by introducing a penalty term for the number of parameters in the model; the penalty term is larger in BIC than in AIC. BIC is classically defined as

$$\text{BIC} = k \log(n) - 2 \log L(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{X}) \tag{3.10}$$

where k is the number of non-zero coefficients in the regression model and L, the estimated likelihood function. Despite the many desired properties, Giraud (2014) reports (a) the above approximation is only valid for sample size n much larger than the number k of parameters in the model and (b) the BIC cannot handle complex collections of models

as in the variable selection (or feature selection) problem in high-dimension as the two main limitations of the criterion. We arrive at the robust BIC by substituting the robust log-likelihood function from Equation (3.9) into the classical log-likelihood function as below.

$$\begin{aligned} \mathrm{BIC}_{\mathrm{rob}}(\nu) &= \nu \log(n) - 2 \log L_{\mathrm{tr}}(\hat{\boldsymbol{\theta}}_{\mathrm{rob}} | \boldsymbol{y}, \boldsymbol{X}) \\ &= \nu \log(n) + n \left[\log(2\pi\hat{\sigma}^2_{\mathrm{rob}}) + 1 - \frac{\nu}{h} \right]. \end{aligned} \tag{3.11}$$

3.1.2 Robust AIC for Tuning Parameter Selection

Similarly, we define and make substitution for the AIC criterion. AIC is classically defined as

$$\mathrm{AIC} = 2k - 2 \log L(\boldsymbol{\theta} | \boldsymbol{y}, \boldsymbol{X}) \tag{3.12}$$

where k is the number of non-zero coefficients in the regression model and L, the estimated likelihood function. Thus, upon substituting the appropriate estimators gives

$$\begin{aligned} \mathrm{AIC}_{\mathrm{rob}}(\nu) &= 2\nu \log(n) - 2 \log L_{\mathrm{tr}}(\hat{\boldsymbol{\theta}}_{\mathrm{rob}} | \boldsymbol{y}, \boldsymbol{X}) \\ &= 2\nu + n \left[\log(2\pi\hat{\sigma}^2_{\mathrm{rob}}) + 1 - \frac{\nu}{h} \right]. \end{aligned} \tag{3.13}$$

3.1.3 Robust Mallows' C_p for Tuning Parameter Selection

Mallows' C_p is the other estimator for the tuning parameter selection. From the classical definition, $C_p = \frac{\mathrm{SSE}_p}{s^2} - n + 2(p+1)$ where $\mathrm{SSE}_p = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$. Here, SSE_p is the sum of squared error for the selected k predictors in the fitted model and s^2 is the mean squared error of the full model. In the robust case using the asymptotically unbiased estimator, we have $\mathrm{SSE}_{\mathrm{p,\,rob}} = c^2 \sum_{i=1}^{n} \hat{w}_i (y_i - \hat{\boldsymbol{\beta}}^T_{\mathrm{rob}} \boldsymbol{x}_i)^2 = \hat{\sigma}^2_{\mathrm{rob}}(h - \nu)$. Since, s^2 can be estimated by $\hat{\sigma}^2_{\mathrm{rob}}$, we now have

$$\begin{aligned} C_{\mathrm{p,\,rob}}(\nu) &= \frac{\hat{\sigma}^2_{\mathrm{rob}}(h-\nu)}{\hat{\sigma}^2_{\mathrm{rob}}} - n + 2(p+1) \\ &= (h - \nu) - n + 2(\nu) \\ &= h + \nu - n. \end{aligned} \tag{3.14}$$

3.2 Computation Algorithm

The algorithm for computing the Robust Penalized Least Trimmed Squares for variable selection in multiple linear regression is presented below.

Algorithm 1: RPLTS algorithm

Input: Data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^p \times \mathbb{R}\}_{i=1}^n$

Given: Initial estimate $\hat{\boldsymbol{\beta}}^{(0)}$, $\lambda > 0$

begin

- Set $k \leftarrow 1$
- Compute residuals
- $\hat{\varepsilon}_i = y_i - (\hat{\boldsymbol{\beta}}^{(k-1)})^T \boldsymbol{x}_i$, $i = 1, 2, \ldots, n$
- *Concentration step*
- Find the smallest h residuals $\hat{\varepsilon}_{i_1}, \ldots, \hat{\varepsilon}_{i_h}$ where $\frac{n+1}{2} \leq h \leq n$
- Let $\hat{w}_i^{(k)} = 1$, if $i \in \{i_1, \ldots, i_h\}$ and $\hat{w}_i^{(k)} = 0$, otherwise
- Fit the usual non-robust penalized regression model with $\lambda P(\cdot)$ to $\{\boldsymbol{x}_{i_j}, y_{i_j}\}_{j=1}^h$
- Update $\hat{\boldsymbol{\beta}}^{(k)}$ with the resulting $\hat{\boldsymbol{\beta}}$
- If $\|\hat{\boldsymbol{\beta}}^{(k)} - \hat{\boldsymbol{\beta}}^{(k-1)}\|_\infty < \varepsilon$ stop, otherwise $k \leftarrow k + 1$ and go to 5

end

Output $(\hat{\boldsymbol{\beta}}^{(k)}, \hat{\boldsymbol{w}}^{(k)})$

The `R` implementation was provided by Pokojovy (2021).

3.3 Breakdown Point of the Proposed Method

According to the work of Alfons et al. (2013), the breakdown point (*bdp*) of the RPLTS method was determined to be $\frac{n-h+1}{n}$ for $\frac{n+1}{2} \leq h \leq n$ since loss function of the proposed method (LTS) satisfies their proposed assumptions. This conclusion was based on the inference for the LTS loss function employed in their method. Clearly, the breakdown

point does not depend on the dimension p. However, the breakdown point increases with increasing value of h. According to Alfons et al. (2013), taking h small enough can possibly produce a breakdown point larger than 50%. It is required for $h > \frac{n}{2}$ since robust statistics aim for models that fit the majority of the data by taking h equal to a fraction α of the sample size, with $\alpha = 0.75$ such that the final estimate is based on a sufficiently large number of observations. This suggestion was shown to guarantee a sufficiently high statistical efficiency in their simulation study with a resulting breakdown point of about $1 - \alpha = 25\%$. As such, with larger predictor variables than the sample size, a high breakdown point is guaranteed.

3.4 Bulk Size Selection

According to Chatterjee and Hadi (2009), the presence of outliers can lead to biased parameter estimation and inappropriate predictions for classical methods. Since outliers pull the regression line towards themselves, they can interfere and produce misleading regression results. The proposed method takes such undesirable effect into account by employing the LTS loss function with an assumption on the maximal percentage of outliers. With this loss function, observations detected as potential outliers are trimmed leaving good observations for modeling. One common way of modeling outliers is through the weights of the observations. The RPLTS algorithm uses (`robustbase::h.alpha.n(0.5 + bdp, n, 1L`) from the robustbase package. A corresponding potential outlier is then assigned 0 and a good observation assigned 1 to form the the minimizer for the weight vector in trimming the data. Thus, observations with weight $w = 1$ are in the bulk whilst those with weight $w = 0$ are not in the bulk. Also, the method only determines bulk points and thus, more careful analysis is needed to decide if they are actually outliers. As seen in the arguments of (`robustbase::h.alpha.n(0.5 + bdp, n, 1L`) is the breakdown point represented as *bdp*. The *bdp* influences the amount of potential outliers the algorithm can accommodate. For this study, the *bdp* takes values from 0 to 0.5. Also, it is worth mentioning that the

primer of the proposed algorithm is adaptive to the `nvcreg` package when there are no outliers in the data (i.e, $h = n$ and bdp set to 0) using the same optimal tuning parameter value. Since robust statistics aim for models that fit the majority of the data (Alfons et al., 2013), the maximum threshold value of 0.5 was carefully chosen such that there will not be more potential outliers than 50% of the sample size.

3.5 Performance Metrics for the Proposed Method

One way to measure the performance of an algorithm is through some performance metrics. Botchkarev (2018b) report that performance metrics (error measures) are vital components of the evaluation frameworks in various fields. A performance metric can be defined as a logical and mathematical construct designed to measure how close the actual results are from what has been expected or predicted (Botchkarev, 2019). With regression experiments in machine learning, performance metrics are used to compare the trained model predictions with the actual data from the testing data set (Makridakis et al., 2018; Botchkarev, 2018a). Forecasting employ performance metrics to measure how much forecasts deviate from observations in order to assess quality and choose forecasting methods, especially in support of supply chain or predicting workload for software development (Carbone and Armstrong, 1982; De Gooijer and Hyndman, 2006).

A vast variety of performance metrics have been described in academic literature. The most commonly mentioned metrics in research studies are Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), etc. Knowledge about a metric's properties need to be systematized to simplify the design and use of the metric (Botchkarev, 2019).

3.5.1 Confusion Matrix

According to Algamal and Lee (2015), confusion matrix is one of the ways to measure the performance of a classification problem where the output can be of two or more type of classes. We restrict ourselves to two classes here. In the field of machine learning

and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (Stehman, 1997). Confusion matrix is formed from four outcomes of binary classification of all data instances of a test dataset as either positive or negative. This classification (or prediction) produces four outcomes which are termed as *true positive (TP)*, *true negative (TN)*, *false positive (FP)* and *false negative (FN)*. Here, true positive refers to the correct positive prediction whilst false positive represents the incorrect positive predictions. Also, true negative are the correct negative prediction while false negative indicate the incorrect negative prediction. This metric was used to assess how well our method can estimate an initial set number of known or true (actual) zero coefficients in a given model. Other measures are computed from the four outcomes and are detailed below.

1. *Error rate (ERR)* is calculated as the number of all incorrect predictions divided by the total size of the data. The best error rate is 0.0, whereas the worst is 1.0.

2. *Accuracy (ACC)* is calculated as the number of correct predictions divided by the total size of the data. The best accuracy is 1.0, whereas the worst is 0.0. It can also be calculated by 1 – ERR. Error costs of positives and negatives are usually different. For instance, one wants to avoid false negatives more than false positives or vice versa. Other basic measures, such as sensitivity and specificity, are more informative than accuracy and error rate in such cases.

3. *Sensitivity (SN)* is calculated as the number of correct positive predictions divided by the total number of positives. It is also called recall (REC) or true positive rate (TPR). The best sensitivity is 1.0, whereas the worst is 0.0.

4. *Specificity (SP)* is calculated as the number of correct negative predictions divided by the total number of negatives. It is also called true negative rate (TNR). The best specificity is 1.0, whereas the worst is 0.0.

5. *Precision (PREC)* is calculated as the number of correct positive predictions divided by the total number of positive predictions. It is also called positive predictive value (PPV). The best precision is 1.0, whereas the worst is 0.0.

6. *False positive rate (FPR)* is calculated as the number of incorrect positive predictions divided by the total number of negatives. The best false positive rate is 0.0 whereas the worst is 1.0. It can also be calculated as 1 – *Specificity*.

3.5.2 Mean Squared Error

Mean squared error (MSE) or mean squared deviation (MSD) is a measure of the quality of an estimator. It measures the average squared difference between the estimated values and the actual value. In fact, MSE is a risk function which is equal to the expected value of the squared error loss. According to Lehmann and Casella (2006), the fact that MSE is almost always strictly positive (and not zero) is because of the randomness or the estimator not accounting for information that could produce a more accurate estimate. It is worth noting that MSE is always non-negative due to the squared function and *values closer* to *zero* are *better*. MSE is the second moment about the origin of the error, and thus, account for both variance and squared bias of the estimator.

3.5.3 Mean Squared Prediction Error

Mean squared prediction error (MSPE) summarizes the predictive ability of a model. It is a measure of a predictor's fit or how well it predicts the true value. Ideally, the closer the value to zero, the closer the prediction is to the true value. For this study, the trimmed MSPE was used to account for the presence of potential outliers.

Chapter 4

Simulation Study

This chapter presents a simulation study to validate our proposed method. The efficiency and robustness of our method is compared with two variable selection methods under the same data condition to demonstrate its advantages.

4.1 Simulated Data: Case Studies

The simulated data was based on the regression model in Equation (2.1) with varying number of predictors (p) and observations (n) where $n > p$. Representations of the variables, their dimensions and statistical distribution follow the description in Section 2.3. The method was evaluated using clean and contaminated data. Two different samples ($n = 200, p = 40$) and ($n = 800, p = 200$) were chosen. These sample sizes were selected to observe the performance of the proposed method for both small and large sample sizes. First, regression coefficients were independently generated from the uniform distribution, $\mathcal{U}(a = 2, b = 10)$, where $a < b$. Here a denotes the minimum bound or value whilst b represents the maximum value for the distribution. For estimation purposes, 40% of the predictors were chosen to be zeroes as a means to check how well the methods estimate the zero coefficients, i.e, assuming those zero coefficients were the true regression coefficients. The remaining 60% predictors were then taken to be greater than zero so not to interfere with the fixed number of zero predictors. This specific idea and the `R` program was obtained from (Mandal and Ghosh, 2019). The predictor variables were then generated from a multivariate normal distribution where each $\boldsymbol{x_i}$ follows the standard normal distribution. The values of ε_i were chosen from standard normal distribution.

4.1.1 Clean Data

Under *clean data* condition, ε_i were chosen from standard normal distribution without any noise added. The figure below shows the distribution of the response variable in the clean data for the sample $(n = 800, p = 200)$.

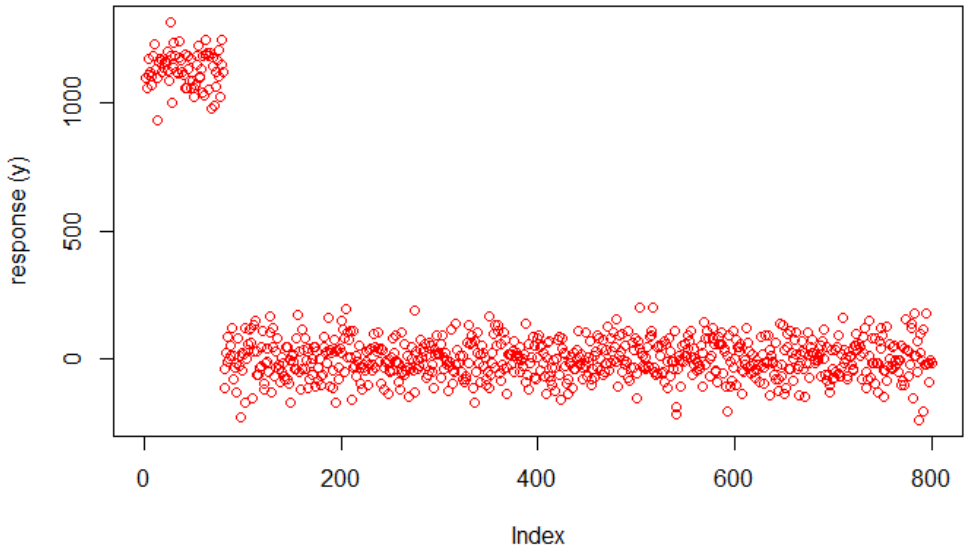

Figure 4.1: Plot of the response variable for the large sample in the clean data.

Figure 4.1 above shows no considerable outliers from a boxplot. This provides satisfactory evidence of no contamination in the response variable.

4.1.2 Data Contamination

To model contaminated data, 0.95-quantile of the y values were taken as noise and added to randomly sampled response values in the data to make them outliers. Exactly 10% of these extreme values were generated in each sample. The plot of the simulated error for the sample $(n = 800, p = 200)$ is shown in Figure 4.2 below.
From Figure 4.2, the outliers were large positive values and clustered at the left top corner.

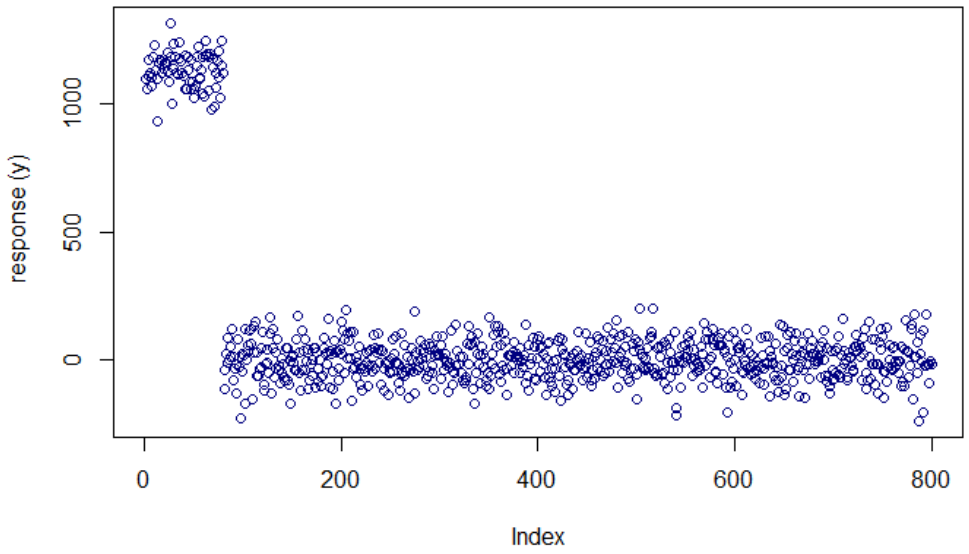

Figure 4.2: Plot of the response variable for the large sample after contamination.

4.2 Tuning Parameter Selection Criterion Used

The behavior of the three proposed tuning parameter criteria were all explored under both data conditions with different samples. However, only the robust BIC criterion showed consistency with its optimal tuning parameter selection and enforced sparsity in the models. The method used the BIC criterion through a robust analog of the *homotopy* path, where it chooses the optimal lambda from a range of provided lambdas that gives the best model fit. Figure 4.3 shows the robust BIC path using the sample ($n = 800, p = 200$) at $bdp = 25\%$ with contaminated data.

In Figure 4.3, the graph exhibits convexity and a local minimum which aligns with the convex nature of Lasso. The red line indicates the optimal value to use in the penalty function of the proposed method. Figure 4.4 also shows the number of significant predictors selected by the method for the regression model. Therefore, the robust BIC criterion was used in the proposed method for the analyses.

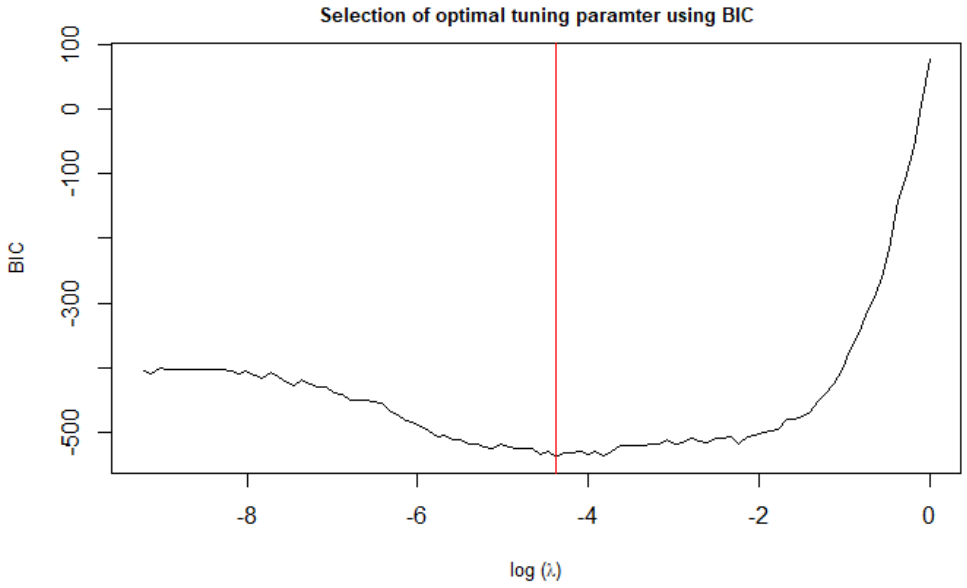

Figure 4.3: Robust BIC optimal tuning parameter selection.

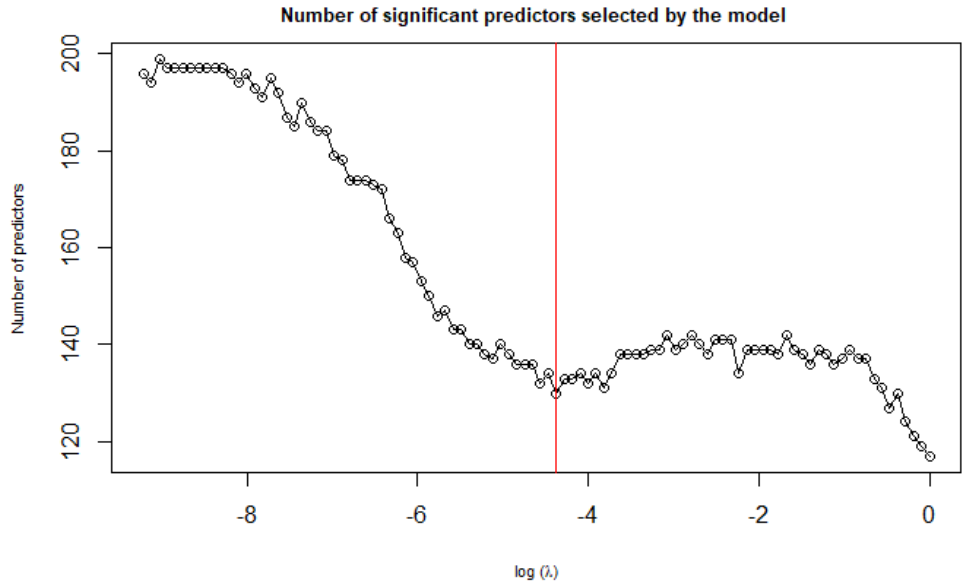

Figure 4.4: Number of significant predictors selected at the optimal tuning parameter.

4.2.1 Method of Evaluation

The performance of the proposed method was tested against LAD-Lasso and the `ncvreg` using statistics from their confusion matrix and MSE as discussed in 3.3. The LAD-Lasso and `ncvreg` were chosen for comparison based on the loss function employed by them. In the proposed method, least trimmed squares is the loss function whereas least absolute deviation is the loss the function in LAD-Lasso. Moreover, the `ncvreg` uses the least squares loss function. Despite all methods being penalization techniques and, thus, uses penalty functions, the `ncvreg` and proposed method accommodate variable penalty functions which are Lasso, SCAD and MCP while the LAD-Lasso uses only the Lasso penalty function. Thus, for fair comparison, the common penalty function employed by them, Lasso, was used in all three methods under each data condition and sample.

4.3 Simulation Results

This section presents the results obtained from the simulation studies under each model. The first subsection covers the results for the clean data.

4.3.1 Results for Clean Data Samples

The selected samples were analyzed among the three penalized variable selection methods. Here, 15 of the initial regression coefficients (true) were set to zero to in $(n = 200, p = 40)$ whilst 80(10%) for the sample $(n = 800, p = 200)$ was chosen to be zeroes. This was done to see how well the methods can estimate these zero regression coefficients. To further check the effect of the breakdown point on the proposed method, the *bdp* was set at 10% and 25% while keeping other values constant. The results are as shown in Table 4.1. Table 4.1 contains the confusion matrix for each method in each situation. Crucial statistics computed from the confusion matrix as performance measures are displayed in Table 4.2.

Table 4.1: Confusion matrix for the methods for clean data (one sample).

		LAD-Lasso		`ncvreg`		Proposed method at $bdp = 10\%$		Proposed method at $bdp = 25\%$	
		True	False	True	False	True	False	True	False
$n = 200$	True	15	0	14	0	0	0	3	0
$p = 40$	False	0	25	1	25	15	25	12	25
$n = 800$	True	80	1	78	0	3	0	3	0
$p = 200$	False	0	119	2	120	77	120	77	120

Table 4.2: Statistics from the confusion matrix for clean data (one sample).

		LAD-Lasso	`ncvreg`	Proposed method at $bdp = 10\%$	Proposed method at $bdp = 25\%$
	Accuracy	1.000	0.975	0.625	0.700
$n = 200$	*Sensitivity*	1.000	0.933	0.000	0.200
$p = 40$	*Specificity*	1.000	1.000	1.000	1.000
	$MSE(\hat{\beta})$	1.17523	0.00024	0.01079	0.01099
	Accuracy	0.995	0.990	0.615	0.615
$n = 800$	*Sensitivity*	1.000	0.975	0.038	0.038
$p = 200$	*Specificity*	0.992	1.000	1.000	1.000
	$MSE(\hat{\beta})$	1.23627	0.00025	0.00044	0.00043

Results for the sample $(n = 200, p = 40)$ clearly indicate that `ncvreg` was most efficient and accurate as it recorded higher scores with the minimum MSE. Though, LAD-Lasso recorded perfect scores for the sensitivity, accuracy and specificty, however, its MSE was the largest among the competitors. The proposed method performed good in terms of accuracy, perfect with specificity but poor at sensitivity. For the same sample size, the performance was better with large *bdp* value for the proposed method. Similarly, the results for $(n = 800, p = 200)$ show that `ncvreg` performed very closely with LAD-Lasso for accuracy, sensitivity and specificity despite LAD-lasso having the maximum for accuracy

and sensitivity. However, `ncvreg` had very small MSE and that should be preferred. The accuracy and specificity score slightly reduced for both `ncvreg` and the proposed method as sample size and number of predictors increased. However, the MSE for the methods reduces with large p and n except for `ncvreg`.

4.3.2 Result for Contaminated Data Samples

Same analyses were performed for both samples, however, under the condition of data contamination. The confusion matrix obtained and its statistics for the three methods are displayed in Tables 4.3 and 4.4, respectively.

Table 4.3: Confusion matrix for the methods under contamination (single sample).

		LAD-Lasso		`ncvreg`		Proposed method at $bdp = 10\%$		Proposed method at $bdp = 25\%$	
		True	False	True	False	True	False	True	False
$n = 200$	True	6	4	15	25	6	0	8	0
$p = 40$	False	9	21	0	0	9	25	7	25
$n = 800$	True	8	13	75	113	45	0	70	0
$p = 200$	False	72	107	5	7	35	120	10	120

It is evident from Table 4.4 that the proposed method outperformed its competitors under the condition of contamination except for sensitivity score in the sample ($n = 200, p = 40$). This performance was better at *bdp* value of 25%. Also, the proposed method was good at classifying the observation as shown in the confusion matrix in Table 4.3 for the large sample ($n = 800, p = 200$). Resulting performance of the methods for this sample as displayed in 4.4 also reveal the proposed method as a better estimator among them. It only recorded a slightly lower sensitivity than the other two methods while achieving the highest accuracy with least MSE. This result improved with the larger *bdp* value of 25%.

Table 4.4: Statistics from the confusion matrix under contamination (single sample).

		LAD-Lasso	`ncvreg`	Proposed method at $bdp = 10\%$	Proposed method at $bdp = 25\%$
	Accuracy	0.675	0.375	0.775	0.825
$n = 200$	*Sensitivity*	0.400	1.000	0.400	0.533
$p = 40$	*Specificity*	0.840	0.000	1.000	1.000
	$MSE(\hat{\beta})$	79.0730	26.3718	0.0105	0.0100
	Accuracy	0.575	0.410	0.825	0.950
$n = 800$	*Sensitivity*	0.100	0.938	0.563	0.875
$p = 200$	*Specificity*	0.892	0.058	1.000	1.000
	$MSE(\hat{\beta})$	149.8613	28.0336	0.0022	0.0006

4.3.3 Average Performance of the Methods

The average performance of the methods were studied for 100 replications of the respective models under the data conditions with results presented in Table 4.5.

Table 4.5: Average performance of the methods for 100 replications.

		Clean				Contaminated			
		Lad-Lasso	`ncvreg`	RPLTS1	RPLTS2	Lad-Lasso	`ncvreg`	RPLTS1	RPLTS2
	Accuracy	0.995	0.940	0.638	0.659	0.578	0.423	0.646	0.681
$n = 200$	*Sensitivity*	0.993	0.850	0.094	0.146	0.164	0.928	0.115	0.201
$p = 40$	*Specificity*	0.993	0.851	0.102	0.155	0.170	0.918	0.124	0.209
	$MSE(\hat{\beta})$	1.438	0.00016	0.00258	0.00258	103.330	28.661	0.00246	0.00239
	Accuracy	0.991	0.985	0.614	0.613	0.579	0.420	0.826	0.927
$n = 800$	*Sensitivity*	0.982	0.962	0.036	0.033	0.137	0.971	0.564	0.818
$p = 200$	*Specificity*	0.982	0.963	0.045	0.043	0.144	0.963	0.568	0.820
	$MSE(\hat{\beta})$	1.165	0.00022	0.00014	0.00013	146.245	25.624	0.00431	0.00517

In the above table, RPLTS1 and RPLTS2 represent the proposed method at 10% and 25% respectively. The proposed method did not perform well with the clean data but showed most efficiency at *bdp* of 25% among the competitors with large p and n.

Chapter 5

Illustrative Example for a Real Dataset

To give a practical illustration, further analyses were performed on a real-world data. The data was collected on acute aquatic toxicity towards the fish *Pimephales promelas* (fathead minnow) on a set of 908 chemicals as reported in Cassotti et al. (2014) and available on the UCI data platform. The dataset contains 9 variables and 546 observations. Thus, 8 attributes (molecular descriptors) of 546 chemicals were used to predict quantitative acute aquatic toxicity towards *LC50* (Daphnia Magna). The variable description indicates LC50 as the response variable which shows the concentration that causes death in 50% of test Daphnia magna over a test duration of 48 hours. The eight molecular descriptors which formed the predictors were; *TPSA(Tot)* (Molecular properties), *SAacc* (Molecular properties), *H-050* (Atom-centred fragments), *MLOGP* (Molecular properties), *RDCHI* (Connectivity indices), *GATS1p* (2D autocorrelations), *nN* (Constitutional indices) and *C-040* (Atom-centred fragments). All values were measured on a continuous scale which meets the basic requirement for use as linear regression problem.

5.1 Exploratory Analyses

First, the data was explored as often done in most statistical analyses to get a fair idea of the nature of the data. Here, the check for missing values was inevitable as the presence of one or more has to be computed by appropriate technique. The result, however, revealed no missing value in the data which was highly desirable for the analyses.

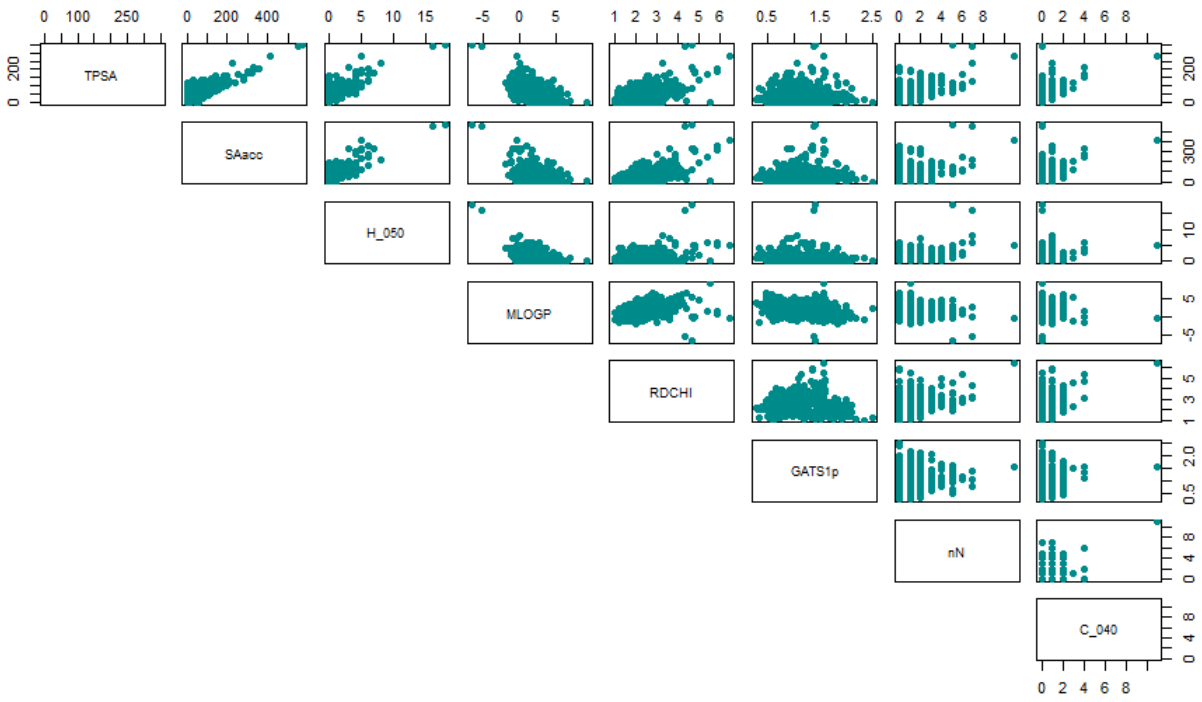

Figure 5.1: Scatterplot matrix among the variables.

Figure 5.1 above shows the scatterplot matrix between the variables in the data providing a visualization on how the data points are distributed in each pair of variables. A bivariate correlation was performed on the predictors to check how the variables are associated in a linear fashion.

Figure 5.2 shows the linear association between the variables computed using robust statistics from the `covMcd` function at bulk size (`alpha` = 0.8) in the `robustbase` package. Except for TPSA which showed strong positive linear association with SAacc and moderate postive linear association with Nn, the remaining predictors had either weak positive or negative linear relationship among themselves. However, the predictors were standardized to reduce the effect of correlation before fitting the models. A further check on the distribution of the response variable showed no severe outlying values. To validate the performance of the methods under contamination, noise was added to random samples of the response variable

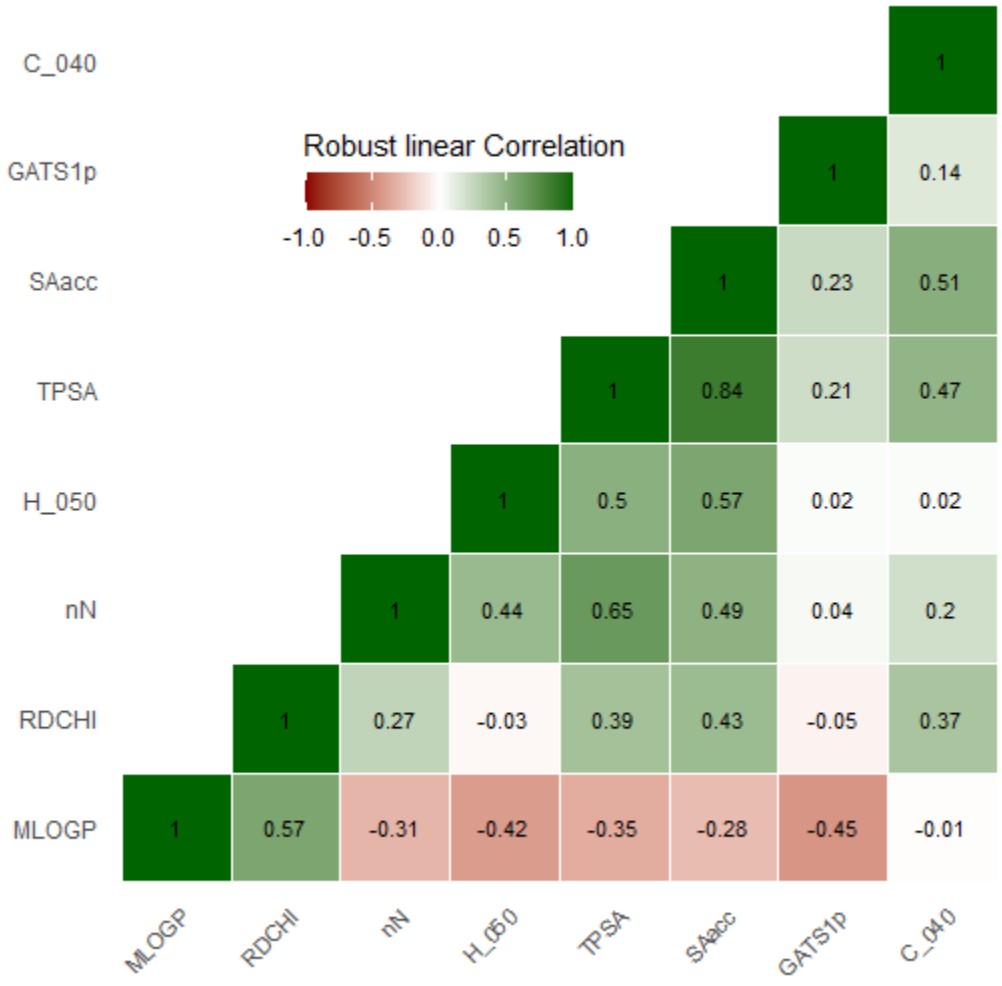

Figure 5.2: Robust linear correlation between the variables.

as did in the simulation.

The next thing was to split the data to constitute the training set and test set. The training set was used to fit the regression model while the test set was used to evaluate the model. The choice of partition for data is not fixed with common choices among researchers like $80:20$, $70:30$ and $60:40$. In such partitions, the training set is chosen larger than the test set. In this study, the data partition was chosen such that the training set constituted

$\frac{2}{3}$ of the data which represent about 67% while the remaining $\frac{1}{3}$ formed the test set.

5.2 Model Fit and Evaluation

To compare the performance of the methods, a regression model was fitted with the training dataset for each method and the sparsity of the models considered. Thus, the number of significant variables selected and those excluded in the model by each method based on the values of the estimated regression coefficients were tabulated. Moreover, the trimmed mean square prediction error was finally used to evaluate how well the model performed. Here 10% of the test data was trimmed based on the response to make the prediction. Summary of the model fits are presented in Table 5.1.

Table 5.1 shows that the `ncvreg` and the proposed method at $bdp = 25\%$ selected six (6) variables each as significant with 5 variables being same for both methods. Accordingly, two predictors were in the linear model for these two methods. On the other hand, the LAD-Lasso selected only two of the eight predictors while excluding the remaining predictors which does not seem convincing. Moreover, the proposed method at $bdp = 10\%$ selected all variables as significant. A sure way to choose an efficient method was through the trimmed MSPE. The method with least trimmed MSPE is considered best and that gives the best model fit for the data. Clearly, the proposed method at bdp =25% had the least trimmed MSPE and thus, the best method for the real data with contamination.

Table 5.1: Summary of model fit by the methods.

	Variables selected	*Variables excluded*	*Trimmed MSPE*
LAD-Lasso	TPSA, MLOGP	SAacc, H-050, RDCHI, GATS1p, nN, C-040	22.36
`ncvreg`	TPSA, H-050, MLOGP, RDCHI, GATS1p, nN	SAacc, C-040	34.36
Proposed method at $bdp = 10\%$	TPSA, SAacc, MLOGP, RDCHI, GATS1p, nN H-050, C-040		10.48
Proposed method at $bdp = 25\%$	TPSA, SAacc, MLOGP, RDCHI, GATS1p, nN	H-050, C-040	6.72

Chapter 6

Discussion and Conclusion

This chapter concludes this work by summarizing the results and providing directions for future study.

6.1 Summary

The attempt to implement robust variable selection methods has been an interesting study with many proposed methods known in literature. This is due to the challenges of analyzing high-dimensional data for estimation and modeling purposes. Classical methods like the OLS have been shown to perform poorly under assumption violations and in the presence of severe outliers. In this study, we proposed a robust variable selection in multiple linear regression via penalized least trimmed squares. The proposed method is a penalization technique which employs varying penalty functions (Lasso, MCP, SCAD). Since the choice of optimal tuning parameter affects the performance of variable selection techniques, a robust BIC criterion was developed and used for the selection. The method has high breakdown point that does not depend on the number of predictors.

In evaluating the performance of the proposed method, it was compared with the `ncvreg` and LAD-Lasso under the conditions of clean and contaminated data. The estimation accuracy of the proposed method was good, though, not optimal under clean data. However, its efficiency increased with increasing sample size, predictors and breakdown point. Also, with contaminated data, the proposed method outperformed its counterparts with very high estimation accuracy and very small MSE. Again, it was observed that the method performed better with increasing sample size and breakdown point with contaminated data. A real

dataset was selected to practically validate the simulation results, now with trimmed MSPE used as the performance metric. With 10% of the test data trimmed, the trimmed MSPE from the model fit for each method was computed and it revealed the proposed method at $bdp = 25\%$ to possess the least value. In fact, the trimmed MSPE of the proposed method at $bdp = 25\%$ was about three times smaller than that of LAD-Lasso and approximately four times smaller than that of the `ncvreg`. This undoubtedly indicates the proposed method as efficient for variable selection especially under severe outliers contamination.

6.2 Future Work

The study identifies the following directions for future work. Though the robust BIC criterion performed well, a more robust optimal tuning parameter selection criterion can be constructed through bootstrap which will improve the efficiency of the method. Moreover, the proposed method can be extended to the multivariate case since such method is currently not known or implemented.

References

Acuna, E. and Rodriguez, C. (2004). A meta analysis study of outlier detection methods in classification. *Technical paper, Department of Mathematics, University of Puerto Rico at Mayaguez*, pages 1–25.

Alfons, A., Croux, C., Gelper, S., et al. (2013). Sparse least trimmed squares regression for analyzing high-dimensional large data sets. *The Annals of Applied Statistics*, 7(1):226–248.

Algamal, Z. Y. and Lee, M. H. (2015). Penalized logistic regression with the adaptive lasso for gene selection in high-dimensional cancer classification. *Expert Systems with Applications*, 42(23):9326–9332.

Barnett, V. and Lewis, T. (1984). Outliers in statistical data. *Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics*.

Basu, A., Harris, I. R., Hjort, N. L., and Jones, M. (1998). Robust and efficient estimation by minimising a density power divergence. *Biometrika*, 85(3):549–559.

Beatty, W., Webb, E., Kesler, D., Raedeke, A., Naylor, L., Humburg, D., and Kesler, D. (2018). Akaike, h. 1973. information theory and an extension of the maximum likelihood principle. *Comparison of Spring Migration Ecology of American Black Ducks (Anas rubripes) and Mallards (Anas platyrhynchos) in the Montezuma Wetlands Complex*, 147:60.

Ben-Gal, I. (2005). Outlier detection. In *Data Mining and Knowledge Discovery Handbook*, pages 131–146. Springer.

Botchkarev, A. (2018a). Evaluating hospital case cost prediction models using azure machine learning studio. *arXiv preprint arXiv:1804.01825*.

Botchkarev, A. (2018b). Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. *arXiv preprint arXiv:1809.03006*.

Botchkarev, A. (2019). A new typology design of performance metrics to measure errors in machine learning regression algorithms. *Interdisciplinary Journal of Information, Knowledge & Management*, 14.

Brown, B., Fearn, T., and Vannucci, M. (1999). The choice of variables in multivariate regression: a non-conjugate Bayesian decision theory approach. *Biometrika*, 86(3):635–648.

Carbone, R. and Armstrong, J. S. (1982). Note. evaluation of extrapolative forecasting methods: results of a survey of academicians and practitioners. *Journal of Forecasting*, 1(2):215–217.

Cassotti, M., Ballabio, D., Consonni, V., Mauri, A., Tetko, I. V., and Todeschini, R. (2014). Prediction of acute aquatic toxicity toward daphnia magna by using the ga-k nn method. *Alternatives to Laboratory Animals*, 42(1):31–41.

Caussinus, H. and Ruiz, A. (1990). Interesting projections of multidimensional data by means of generalized principal component analyses. In *Compstat*, pages 121–126. Springer.

Charbonnier, P., Blanc-Féraud, L., Aubert, G., and Barlaud, M. (1997). Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on Image Processing*, 6(2):298–311.

Chatterjee, S. and Hadi, A. S. (2009). *Sensitivity analysis in linear regression*, volume 327. John Wiley & Sons.

Chen, Z., Tang, M.-L., Gao, W., and Shi, N.-Z. (2014). New robust variable selection methods for linear regression models. *Scandinavian Journal of Statistics*, 41(3):725–741.

Coakley, C. W. and Hettmansperger, T. P. (1993). A bounded influence, high breakdown, efficient regression estimator. *Journal of the American Statistical Association*, 88(423):872–880.

Croux, C. and Haesbroeck, G. (1999). Influence function and efficiency of the minimum covariance determinant scatter matrix estimator. *Journal of Multivariate Analysis*, 71(2):161–190.

Davies, L. and Gather, U. (1993). The identification of multiple outliers. *Journal of the American Statistical Association*, 88(423):782–792.

De Gooijer, J. G. and Hyndman, R. J. (2006). 25 years of time series forecasting. *International Journal of Forecasting*, 22(3):443–473.

Donoho, D. L. and Huber, P. J. (1983). The notion of breakdown point. *A Festschrift for Erich L. Lehmann*, 157184.

Enderlein, G. (1987). Hawkins, DM: Identification of outliers. Chapman and Hall, London–New York 1980, 188.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360.

Fan, J. and Lv, J. (2011). Nonconcave penalized likelihood with np-dimensionality. *IEEE Transactions on Information Theory*, 57(8):5467–5484.

Frank, L. E. and Friedman, J. H. (1993). A statistical view of some chemometrics regression tools. *Technometrics*, 35(2):109–135.

Gao, X. and Feng, Y. (2018). Penalized weighted least absolute deviation regression. *Statistics and Its Interface*, 11(1):79–89.

Gao, X. and Huang, J. (2010). Asymptotic analysis of high-dimensional lad regression with lasso. *Statistica Sinica*, pages 1485–1506.

Giraud, C. (2014). *Introduction to High-Dimensional Statistics*, volume 138. CRC Press.

Gladitz, J. (1988). Barnett, v. & lewis, t.: Outliers in statistical data, john wiley & sons, Chichester–New York–Brisbane–Toronto–Singapore, 1984.

Golub, G. H., Heath, M., and Wahba, G. (1979). Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223.

Gruber, M. H. (1998). Improving efficiency by shrinkage, Statistics: Textbooks and Monographs, vol. 156.

Gürünlü Alma, Ö., Kurt, S., and Uğur, A. (2011). Genetic algorithms for outlier detection in multiple regression with different information criteria. *Journal of Statistical Computation and Simulation*, 81(1):29–47.

Hadi, A. S. (1992). Identifying multiple outliers in multivariate data. *Journal of the Royal Statistical Society: Series B (Methodological)*, 54(3):761–771.

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

Hampel, F. R. (1974). The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393.

Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., and Stahel, W. A. (2011). *Robust Statistics: the Approach Based on Influence Functions*, volume 196. John Wiley & Sons.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media.

Hawkins, D. M. (1980). *Identification of outliers*, volume 11. Springer.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Huber, P. J. (1992). Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer.

Hurvich, C. M. and Tsai, C.-L. (1990). Model selection for least absolute deviations regression in small samples. *Statistics & Probability Letters*, 9(3):259–265.

Iglewicz, B. and Martinez, J. (1982). Outlier detection using robust measures of scale. *Journal of Statistical Computation and Simulation*, 15(4):285–293.

Imon, A. R. and Hadi, A. S. (2013). Identification of multiple high leverage points in logistic regression. *Journal of Applied Statistics*, 40(12):2601–2616.

Jaeckel, L. A. (1972). Estimating regression coefficients by minimizing the dispersion of the residuals. *The Annals of Mathematical Statistics*, pages 1449–1458.

Jiang, L. and Liao, H. (2020). Mixed fuzzy least absolute regression analysis with quantitative and probabilistic linguistic information. *Fuzzy Sets and Systems*, 387:35–48.

Jiang, Y., Wang, Y., Zhang, J., Xie, B., Liao, J., and Liao, W. (2021). Outlier detection and robust variable selection via the penalized weighted lad-lasso method. *Journal of Applied Statistics*, 48(2):234–246.

Jobe, J. M. and Pokojovy, M. (2015). A cluster-based outlier detection scheme for multivariate data. *Journal of the American Statistical Association*, 110(512):1543–1551.

Johnson, B. A. and Peng, L. (2008). Rank-based variable selection. *Journal of Nonparametric Statistics*, 20(3):241–252.

Jung, K.-M. (2007). Least trimmed squares estimator in the errors-in-variables model. *Journal of Applied Statistics*, 34(3):331–338.

Koenker (2005). *Quantile Regression (Econometric Society Monographs; No. 38)*. Cambridge University Press.

Kuha, J. (2004). Aic and bic: Comparisons of assumptions and performance. *Sociological Methods & Research*, 33(2):188–229.

Lambert-Lacroix, S., Zwald, L., et al. (2011). Robust regression through the Huber's criterion and adaptive lasso penalty. *Electronic Journal of Statistics*, 5:1015–1053.

Lange, K. (1990). Convergence of em image reconstruction algorithms with gibbs smoothing. *IEEE transactions on medical imaging*, 9(4):439–446.

Lehmann, E. L. and Casella, G. (2006). *Theory of Point Estimation*. Springer Science & Business Media.

Leng, C. (2010). Variable selection and coefficient estimation via regularized rank regression. *Statistica Sinica*, pages 167–181.

Leroy, A. M. and Rousseeuw, P. J. (1987). Robust regression and outlier detection. *Wiley Series in Probability and Mathematical Statistics*.

Li, Y. and Zhu, J. (2008). L 1-norm quantile regression. *Journal of Computational and Graphical Statistics*, 17(1):163–185.

Liu, H., Shah, S., and Jiang, W. (2004). On-line outlier detection and data cleaning. *Computers & Chemical Engineering*, 28(9):1635–1647.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2018). Statistical and machine learning forecasting methods: Concerns and ways forward. *PloS one*, 13(3):e0194889.

Mallows, C. (1973). Some remarks of Cp, Technometrics.

Mandal, A. and Ghosh, S. (2019). Robust variable selection criteria for the penalized regression. *arXiv preprint arXiv:1912.12550*.

Maronna, R. A., Martin, R. D., Yohai, V. J., and Salibián-Barrera, M. (2019). *Robust Statistics: Theory and Methods (with R)*. John Wiley & Sons.

Martin, R. D. and Thomson, D. J. (1982). Robust-resistant spectrum estimation. *Proceedings of the IEEE*, 70(9):1097–1115.

Millar, A. M. and Hamilton, D. C. (1999). Modern outlier detection methods and thier effect on subsequent inference. *Journal of Statistical Computation and Simulation*, 64(2):125–150.

Nurunnabi, A., Hadi, A. S., and Imon, A. (2014). Procedures for the identification of multiple influential observations in linear regression. *Journal of Applied Statistics*, 41(6):1315–1331.

Papadimitriou, S., Kitagawa, H., Gibbons, P. B., and Faloutsos, C. (2003). Loci: Fast outlier detection using the local correlation integral. In *Proceedings 19th International Conference on Data Engineering (Cat. No. 03CH37405)*, pages 315–326. IEEE.

Riani, M., Atkinson, A. C., and Cerioli, A. (2009). Finding an unknown number of multivariate outliers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(2):447–466.

Ronchetti, E. (1985). Robust model selection in regression. *Statistics & Probability Letters*, 3(1):21–23.

Ronchetti, E. and Staudte, R. G. (1994). A robust version of mallows's cp. *Journal of the American Statistical Association*, 89(426):550–559.

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

Schmid, C. and Zisserman, A. (2000). The geometry and matching of lines and curves over multiple views. *International Journal of Computer Vision*, 40(3):199–233.

Schwarz, G. et al. (1978). Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464.

Stehman, S. V. (1997). Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, 62(1):77–89.

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

Tukey, J. W. et al. (1977). *Exploratory Data Analysis*, volume 2. Reading, Mass.

Wang, H., Li, G., and Jiang, G. (2007). Robust regression shrinkage and consistent variable selection through the LAD-Lasso. *Journal of Business & Economic Statistics*, 25(3):347–355.

Wang, L. and Li, R. (2009). Weighted wilcoxon-type smoothly clipped absolute deviation method. *Biometrics*, 65(2):564–571.

Wang, X., Jiang, Y., Huang, M., and Zhang, H. (2013). Robust variable selection with exponential squared loss. *Journal of the American Statistical Association*, 108(502):632–643.

Williams, G., Baxter, R., He, H., Hawkins, S., and Gu, L. (2002). A comparative study of rnn for outlier detection in data mining. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 709–712. IEEE.

Wu, C. and Ma, S. (2015). A selective review of robust variable selection with applications in bioinformatics. *Briefings in Bioinformatics*, 16(5):873–883.

Wu, Y. and Liu, Y. (2009). Variable selection in quantile regression. *Statistica Sinica*, pages 801–817.

Yohai, V. J. and Zamar, R. H. (1988). High breakdown-point estimates of regression by means of the minimization of an efficient scale. *Journal of the American statistical association*, 83(402):406–413.

Zhang, T. (2010). Analysis of multi-stage convex relaxation for sparse regularization. *Journal of Machine Learning Research*, 11(3).

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

Zou, H. and Hastie, T. (2003). Regression shrinkage and selection via the elastic net, with applications to microarrays. *Journal of Royal Statistical Society Series B*, 67:301–20.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.

Zou, H. and Yuan, M. (2008). Regularized simultaneous model selection in multiple quantiles regression. *Computational Statistics & Data Analysis*, 52(12):5296–5304.

Zou, H., Yuan, M., et al. (2008). Composite quantile regression and the oracle model selection theory. *The Annals of Statistics*, 36(3):1108–1126.

Appendix

Code for Real Data Illustration

```
setwd("C:/Users/Kojo␣Asiamah/Desktop/attempt/simulation
/Real␣data")
require(robustbase)
library(caTools)
real <- read.csv(file = "qsar_aquatic_toxicity.csv",
                 header = F, stringsAsFactors = F)
dim(real)
names(real)<-c("TPSA", "SAacc", "H_050", "MLOGP",
                  "RDCHI", "GATS1p", "nN", "C_040", "LC50")
anyNA(real)  # checking for missing values
head(real)   # viewing the first 6 observations
cor(real)   # checking the bivariate correlation
# -------------------
# Data partitioning
# -------------------
set.seed(11)
split = sample.split(real, SplitRatio = 2/3)
D1 = subset(real, split == TRUE)           # training set
D2 = subset(real, split == FALSE)          # test set
dim(D1)
D1 <- as.matrix(D1)
y <- D1[ ,  9]          # response variable
X <- D1[ , -9]          # predictors
# Trimming D2
```

```
D2 <- real[order(D2[,9]), ]
D2 <- D2[-c(1: floor(0.05*nrow(D2)), nrow(D2),
            ceiling(0.95*nrow(D2)):nrow(D2)), ]
# ------------------------------------------------
# Checking the distribution of error terms
# ------------------------------------------------
plot(y)
boxplot(y)
# ----------------------------------------
# Generating and adding noise to the response
# ----------------------------------------
noise = rep(quantile(y, 0.55)*10, floor(0.1*length(y)))
y[1:floor(0.1*length(y))] = sample(y,
                          floor(0.1*length(y)),
                           replace = F) + noise
plot(y)  # nature of error terms after adding noise
  # ---------------------------------------
  # Robust regression for the FULL model
  # ---------------------------------------
  ncvreg.robust.full = function(X, y,
                                bdp = 0.25,
                                verbose = F) {
  n = nrow(X)
  p = ncol(X)
  h = robustbase::h.alpha.n(0.5 + bdp, n, 1L)
  #fit = robustbase::lmrob.fit(X, y, control =
  robustbase::lmrob.control())
  fit = robustbase::lmrob.fit(cbind(1, apply(X, 2, scale,
      center= s_Qn(X, mu.too = TRUE)[1],
      scale = Qn(X, finite.corr = FALSE))),
```

```
      y, control = robustbase::lmrob.control())
                  # Intercept is added to X
    beta.hat    = fit$coefficients # Non-sparse beta estimate
  sigma.hat   = fit$scale # Should be used as the
  # error estimate
                            # with ALL predictors included
  df.hat  = sum(abs(beta.hat) >= .Machine$double.eps)
  # Equal to p with probability 1
  epsilon.hat = fit$residuals
  return(list(beta.hat = beta.hat, sigma.hat = sigma.hat,
     df.hat = df.hat, epsilon.hat = epsilon.hat))
}

# ----------------------------
# Penalized robust regression
# ----------------------------
ncvreg.robust = function(X,
   y,
   full.model = NULL, # Provide full model
   # to improve efficiency
   beta.hat.init = NULL,
   penalty = c("MCP", "SCAD", "lasso"),
   gamma = switch(penalty, SCAD = 3.7, 3),
   alpha = 1.0,lambda,eps = 1.0E-06,
   max.iter = 1000L,max.iter.C.step = 500L,
   penalty.factor = rep(1.0, ncol(X)),
   warn = TRUE, bdp = 0.25,
   # breakdown points (between
   # 0 = least robust and 0.5 = most  robust)
   verbose = F){
```

```
n = nrow(X)
p = ncol(X)
h = robustbase::h.alpha.n(0.5 + bdp, n, 1L)
if (is.null(full.model)) {
  full.model = ncvreg.robust.full(X, y, bdp, verbose)
}
if (is.null(beta.hat.init)) {
  beta.hat.init = full.model$beta.hat
}
X = cbind(1, apply(X, 2, scale, center =
s_Qn(X, mu.too = TRUE)[1],
scale = Qn(X, finite.corr = FALSE)))
residuals = as.vector(y - X %*% beta.hat.init)
best = order(residuals, decreasing = FALSE)[1:h]
beta = beta.hat.init
for (iter in 1:max.iter.C.step) {
  fit = ncvreg::ncvfit(X = X[best, ], y = y[best],
  init = beta,
   penalty, gamma, alpha, lambda, eps, max.iter,
    penalty.factor, warn)
  dbeta = max(abs(as.vector(fit$beta) - beta))
  beta = as.vector(fit$beta)
  residuals = (y - X %*% matrix(beta, nrow = p + 1))^2
  best = order(residuals, decreasing = FALSE)[1:h]
  loss = sum(residuals[best])
  if (verbose) {
    cat("C-step iter = ", iter, ": ",
    "loss = ", loss, ",beta = (", paste(beta,
    collapse = ", "), ")\n", sep = "")
  }
```

```
    if ((dbeta < eps) || (iter == max.iter.C.step)) {
      res = fit
      break
    }
  }
  if (warn && (iter == max.iter.C.step)) {
    warning("Maximum number of C-steps achieved.
    #  Consider increasing max.iter.C.step.")
  }

  res$n    = n
  res$best = best
  res$loss = loss
  res$residualss = residuals

  res$beta.hat = res$beta
  res[["beta"]] = NULL
  res$nu.hat = sum(abs(fit$beta[-1])
  >= .Machine$double.eps)
  res$w.hat  = rep(0.0, n)
  res$w.hat[best] = 1.0

  alpha = 1 - (h - res$nu.hat)/(n - res$nu.hat)
  res$sigma.hat = sqrt(res$loss/(h - res$nu.hat)/
                       pchisq(qchisq(1 - alpha, 1), 3))

  # for optimal tuning parameter selection
  res$BIC = res$nu.hat*(log(n)) + n*(log(2*pi*res$sigma.hat)
   + 1  - (res$nu.hat/h))
  res$AIC = 2*res$nu.hat +  n*(log(2*pi*res$sigma.hat)
```

```
  + 1 - (res$nu.hat/h))
  res$BIC = n*(res$sigma.hat/full.model$sigma.hat)^2 -
  n + 2*(res$nu.hat+1)
  return(res)
}

# -----------------
# "Homotopy" path
# -----------------
full.model = ncvreg.robust.full(X = apply(X, 2, scale,
center =s_Qn(X, mu.too = TRUE)[1], scale = Qn(X,
   finite.corr = FALSE)), y = y)
lambda.grid = exp(seq(from =
log(0.0001), to = log(0.99), length.out = 100L))
BIC.grid = rep(0.0, 100L)
nu.hat.grid = rep(0.0, 100L)
beta.grid <- rep(0.0, 100L)
beta.hat = full.model$beta.hat

BIC.opt = Inf
lambda.opt = NULL
beta.hat.opt = NULL
nu.hat.opt = NULL

for (i in 1:length(lambda.grid)) {
  lambda = lambda.grid[i]
  fit = ncvreg.robust(X = apply(X, 2, scale,
  center =   s_Qn(X, mu.too = TRUE)[1], scale =
  Qn(X, finite.corr = FALSE)),
 y = y, full.model = full.model, beta.hat.
```

```
 init = beta.hat,  penalty = "lasso",
 lambda = lambda, verbose = FALSE)
  beta.grid = fit$beta.hat[i]
  beta.hat = fit$beta.hat
  BIC.grid[i] = fit$BIC
  nu.hat.grid[i] = fit$nu.hat

  if (fit$BIC < BIC.opt) {
    BIC.opt = fit$BIC
    lambda.opt = lambda
    beta.hat.opt = fit$beta.hat
    nu.hat.opt = fit$nu.hat
  }
}
plot(log(lambda.grid), BIC.grid, type = "l",
     xlab = expression(paste("log␣(",lambda ,")")),
     , ylab = "BIC")
abline(v = log(lambda.opt), col = "red")
plot(log(lambda.grid), nu.hat.grid, type = "l",
     xlab = expression(paste("log␣(",lambda ,")")),
     ylab = "nu-hat")
points(log(lambda.grid), nu.hat.grid)
abline(v = log(lambda.opt), col = "red")
cat("\n\n\n")
cat("lambda␣opt␣=␣", lambda.opt, "\n", sep = "")
cat("beta-hat␣opt␣=␣(", paste(beta.hat.opt,
 collapse = ",␣"), ")\n", sep = "")
cat("nu-hat␣opt␣=␣", nu.hat.opt, "\n", sep = "")
```

```
# ----------------------------------
# Fitting model with optimal lambda
# ----------------------------------
fit.rob = ncvreg.robust(X = 2, scale,
  center =   s_Qn(X, mu.too = TRUE)[1], scale =
  Qn(X, finite.corr = FALSE)),
 , y = y, penalty = "lasso", lambda =  lambda.opt)
cat("robust beta-hat w/ outliers =
(", paste(fit.rob$beta, collapse = ", ")
, ")\n", sep = "")
# ---------------------------------------
# Computing the MSPE for proposed method
# ---------------------------------------
yhat.prop <- as.matrix(cbind(1,D2[ ,-9]))
 %*% as.matrix(fit.rob$beta.hat)
yobs <- as.matrix(D2[, 9])
MSPE.prop <- mean((yobs-yhat.prop)^2)
MSPE.prop
# ------------------------------------------------
# Using the LAD-lasso package to fit same data
# ------------------------------------------------
 library("flare")
mod1 = slim(X=X, Y=y, nlambda = 5, lambda = NULL,
             lambda.min.ratio=0.3,
             method ="lasso",        # Lasso shrinkage
             q=1)                    # LAD loss
# ------------------------------
# Computing the MSPE for LAD-lasso
# ------------------------------
```

```
bhat<-c(mod1$intercept[ , 2], mod1$beta[ , 2]); bhat
yhat.lad <- as.matrix(cbind(1,D2[ ,-9]))
%*% as.matrix(bhat)
yobs <- as.matrix(D2[, 9])
MSPE.lad <- mean((yobs-yhat.lad)^2)
MSPE.lad


# ----------------------------------------
# Using the ncvreg package to fit same data
# ----------------------------------------

require(ncvreg)
newfit<- ncvreg(X, y, family = "gaussian",
penalty = "lasso")
cvfit <- cv.ncvreg(X, y, family = "gaussian",
penalty = "lasso")

summary(newfit , cvfit$lambda.min)
# ------------------------------
# Computing the MSPE for ncvreg
# ------------------------------

yhat.ncv <-predict(newfit, as.matrix(D2[ ,-9]),
 type="response", lambda = cvfit$lambda.min)
yobs <- as.matrix(D2[, 9])
MSPE.ncv <- mean((yobs-yhat.ncv)^2)
MSPE.ncv
```

```
# ------------------------------------------------
# Alternative way to compute the MSPE for ncvreg
# ------------------------------------------------
bh <- coef(newfit, cvfit$lambda.min)
yhatp <- as.matrix(cbind(1,D2[ ,-9])) %*% as.matrix(bh)
yobs <- as.matrix(D2[, 9])
MSPE.prop <- mean((yobs-yhatp)^2)
MSPE.prop
```

Code for Simulation (one sample)

```
# ----------------
# Code for RPWLTS
# install.packages("ncvreg")
# install.packages("robustbase")


# ----------------------------
# Penalized robust regression
# ----------------------------
ncvreg.robust = function(X,
   y,
   full.model = NULL, # Provide full model
   # to improve efficiency
   beta.hat.init = NULL,
   penalty = c("MCP", "SCAD", "lasso"),
   gamma = switch(penalty, SCAD = 3.7, 3),
   alpha = 1.0,lambda,eps = 1.0E-06,
```

```
 max.iter = 1000L,max.iter.C.step = 500L,
 penalty.factor = rep(1.0, ncol(X)),
 warn = TRUE, bdp = 0.25,
 # breakdown points (between
 # 0 = least robust and 0.5 = most  robust)
 verbose = F){
n = nrow(X)
p = ncol(X)
h = robustbase::h.alpha.n(0.5 + bdp, n, 1L)
if (is.null(full.model)) {
  full.model = ncvreg.robust.full(X, y, bdp, verbose)
}
if (is.null(beta.hat.init)) {
  beta.hat.init = full.model$beta.hat
}
X = cbind(1, apply(X, 2, scale, center =
s_Qn(X, mu.too = TRUE)[1],
scale = Qn(X, finite.corr = FALSE)))
residuals = as.vector(y - X %*% beta.hat.init)
best = order(residuals, decreasing = FALSE)[1:h]
beta = beta.hat.init
for (iter in 1:max.iter.C.step) {
  fit = ncvreg::ncvfit(X = X[best, ], y = y[best], init = beta,
   penalty, gamma, alpha, lambda, eps, max.iter,
    penalty.factor, warn)
  dbeta = max(abs(as.vector(fit$beta) - beta))
  beta = as.vector(fit$beta)
  residuals = (y - X %*% matrix(beta, nrow = p + 1))^2
  best = order(residuals, decreasing = FALSE)[1:h]
  loss = sum(residuals[best])
```

```
    if (verbose) {
      cat("C-step␣iter␣=␣", iter, ":␣",
      "loss␣=␣", loss, ",beta␣=␣(", paste(beta,
      collapse = ",␣"), ")\n", sep = "")
    }
    if ((dbeta < eps) || (iter == max.iter.C.step)) {
      res = fit
      break
    }
  }
  if (warn && (iter == max.iter.C.step)) {
    warning("Maximum␣number␣of␣C-steps␣achieved.
␣␣␣␣#␣Consider␣increasing␣max.iter.C.step.")
  }

  res$n    = n
  res$best = best
  res$loss = loss
  res$residualss = residuals

  res$beta.hat = res$beta
  res[["beta"]] = NULL
  res$nu.hat = sum(abs(fit$beta[-1])
  >= .Machine$double.eps)
  res$w.hat  = rep(0.0, n)
  res$w.hat[best] = 1.0

  alpha = 1 - (h - res$nu.hat)/(n - res$nu.hat)
  res$sigma.hat = sqrt(res$loss/(h - res$nu.hat)/
                       pchisq(qchisq(1 - alpha, 1), 3))
```

```
  # for optimal tuning parameter selection
  res$BIC = res$nu.hat*(log(n)) + n*(log(2*pi*res$sigma.hat)
   + 1  - (res$nu.hat/h))
  res$AIC = 2*res$nu.hat +  n*(log(2*pi*res$sigma.hat)
  + 1 - (res$nu.hat/h))
  res$BIC = n*(res$sigma.hat/full.model$sigma.hat)^2 -
  n + 2*(res$nu.hat+1)
  return(res)
}

# ------------------------------
# Data Generation - One Sample
# ------------------------------
set.seed(1)
p = 200
n = 800
X = matrix(rnorm(n*p), nrow = n, ncol = p)
 # Design Matrix
beta = c(runif(120, 2, 10), rep(0, 80))    # beta
eps = 0.1*matrix(rnorm(n), nrow = n, ncol = 1)
# random error
y = X %*% matrix(beta, nrow = p, ncol = 1) + eps
# response variable
X1 <- scale(X, center = T, scale = T)
# ----------
# Adding noise
# ----------
noise = rep(quantile(y, 0.95)*10, floor(0.1*length(y)))
y[1:floor(0.1*length(y))] = sample(y, floor(0.1*length(y))
```

```
                          , replace = F) + noise
# ----------
# LAD-lasso
# ----------
library("flare")
mod1 = slim(X=X1, Y=y, nlambda = 3, lambda = NULL,
            lambda.min.ratio=0.3,
            method ="lasso",        # Lasso shrinkage
            q=1)                    # LAD loss
beta.hat.lad <- as.vector(c(mod1$intercept[ , 2],
mod1$beta[ , 2]))
sum((beta==0) & (mod1$beta[ , 2] == 0))
# MSE for lad-lasso
mse_lad = mean((beta - beta.hat.lad[-1])^2)
mse_lad
# Generating the confusion matrix for LAD-lasso
TP <- sum((beta==0) & (mod1$beta[ , 2] == 0)) # TP
FP <- sum((beta!=0) & (mod1$beta[ , 2] == 0)) # FP
FN <- sum((beta==0) & (mod1$beta[ , 2] != 0)) # FN
TN <- sum((beta!=0) & (mod1$beta[ , 2] != 0)) # TN
conf_mat <- matrix(c(TP, FP, FN, TN),
ncol = 2, byrow = T)
conf_mat <- as.table(conf_mat)
rownames(conf_mat)<-c("TRUE", "FALSE")
colnames(conf_mat)<-c("TRUE", "FALSE")
# computing the metrics from confusion matrix
require(caret)
require(robustbase)
confusionMatrix(conf_mat)
require(ncvreg)
```

```
nfi <- ncvreg(X, y, family = "gaussian",
penalty = "lasso")
cvfit <- cv.ncvreg(X, y, family = "gaussian",
penalty = "lasso")
nfi2 <- ncvreg(X, y, family = "gaussian",
penalty = "lasso")
nc_nonrob<-coef(nfi2, cvfit$lambda.min)
mse_nc = mean((beta - nc_nonrob[-1])^2)
mse_nc
# Generating the confusion matrix
TP <- sum((beta==0) & (nc_nonrob[-1]==0)) # TP
FP <- sum((beta!=0) & (nc_nonrob[-1]==0)) # FP
FN <- sum((beta ==0) & (nc_nonrob[-1]!=0)) # FN
TN <- sum((beta!=0) & (nc_nonrob[-1]!=0)) # TN
conf_mat <- matrix(c(TP, FP, FN, TN), ncol = 2, byrow = T)
conf_mat <- as.table(conf_mat)
rownames(conf_mat)<-c("TRUE", "FALSE")
colnames(conf_mat)<-c("TRUE", "FALSE")
# computing the metrics from confusion matrix
confusionMatrix(conf_mat)

# -----------------
# "Homotopy" path
# -----------------
full.model = ncvreg.robust.full(X = apply(X, 2, scale,
center =s_Qn(X, mu.too = TRUE)[1], scale = Qn(X,
   finite.corr = FALSE)), y = y)
lambda.grid = exp(seq(from =
log(0.0001), to = log(0.99), length.out = 100L))
BIC.grid = rep(0.0, 100L)
```

```
nu.hat.grid = rep(0.0, 100L)
beta.grid <- rep(0.0, 100L)
beta.hat = full.model$beta.hat

BIC.opt = Inf
lambda.opt = NULL
beta.hat.opt = NULL
nu.hat.opt = NULL

for (i in 1:length(lambda.grid)) {
  lambda = lambda.grid[i]
  fit = ncvreg.robust(X = apply(X, 2, scale,
  center =   s_Qn(X, mu.too = TRUE)[1], scale =
  Qn(X, finite.corr = FALSE)),
 y = y, full.model = full.model, beta.hat.
 init = beta.hat,  penalty = "lasso",
 lambda = lambda, verbose = FALSE)
  beta.grid = fit$beta.hat[i]
  beta.hat = fit$beta.hat
  BIC.grid[i] = fit$BIC
  nu.hat.grid[i] = fit$nu.hat

  if (fit$BIC < BIC.opt) {
    BIC.opt = fit$BIC
    lambda.opt = lambda
    beta.hat.opt = fit$beta.hat
    nu.hat.opt = fit$nu.hat
  }
}
plot(log(lambda.grid), BIC.grid, type = "l",
```

```
      xlab = expression(paste("log␣(",lambda ,")"))
      , ylab = "BIC")
abline(v = log(lambda.opt), col = "red")
plot(log(lambda.grid), nu.hat.grid, type = "l",
      xlab = expression(paste("log␣(",lambda ,")")),
      ylab = "nu-hat")
points(log(lambda.grid), nu.hat.grid)
abline(v = log(lambda.opt), col = "red")
cat("\n\n\n")
cat("lambda␣opt␣=␣", lambda.opt, "\n", sep = "")
cat("beta-hat␣opt␣=␣(", paste(beta.hat.opt,
 collapse = ",␣"), ")\n", sep = "")
cat("nu-hat␣opt␣=␣", nu.hat.opt, "\n", sep = "")

# Generating the confusion matrix
TP <- sum((beta==0) & (beta.hat.opt[-1]==0)) # TP
FP <- sum((beta!=0) & (beta.hat.opt[-1]==0)) # FP
FN <- sum((beta ==0) & (beta.hat.opt[-1]!=0)) # FN
TN <- sum((beta!=0) & (beta.hat.opt[-1]!=0)) # TN
conf_mat <- matrix(c(TP, FP, FN, TN),
                   ncol = 2, byrow = T)
conf_mat <- as.table(conf_mat)
rownames(conf_mat)<-c("TRUE", "FALSE")
colnames(conf_mat)<-c("TRUE", "FALSE")

# computing the metrics from confusion matrix
mse_prop_method = mean((beta -beta.hat.opt[-1])^2)
mse_prop_method
confusionMatrix(conf_mat)
```

```
# ----------------------------
# Robust with outliers
# ----------------------------
fit.rob = ncvreg.robust(X = X, y = y,
penalty = "lasso", lambda = lambda.opt )
cat("robust beta-hat w/ outliers =
 (", paste(fit.rob$beta, collapse = ", "), ")
 \n", sep = "")
sum(fit.rob$beta[-1]==0)

mse_p2 = mean((beta - fit.rob$beta[-1])^2)
mse_p2


# Generating the confusion matrix
TP <- sum((beta==0) & (fit.rob$beta[-1]==0)) # TP
FP <- sum((beta!=0) & (fit.rob$beta[-1]==0)) # FP
FN <- sum((beta ==0) & (fit.rob$beta[-1]!=0)) # FN
TN <- sum((beta!=0) & (fit.rob$beta[-1]!=0)) # TN

conf_mat <- matrix(c(TP, FP, FN, TN), ncol = 2, byrow = T)
conf_mat <- as.table(conf_mat)
rownames(conf_mat)<-c("TRUE", "FALSE")
colnames(conf_mat)<-c("TRUE", "FALSE")

# computing the metrics from confusion matrix
confusionMatrix(conf_mat)
```

Code for Simulation (100 replications)

```
# ----------------------------
# Penalized robust regression
# ----------------------------
ncvreg.robust = function(X,
   y,
   full.model = NULL, # Provide full model
   # to improve efficiency
   beta.hat.init = NULL,
   penalty = c("MCP", "SCAD", "lasso"),
   gamma = switch(penalty, SCAD = 3.7, 3),
   alpha = 1.0,lambda,eps = 1.0E-06,
   max.iter = 1000L,max.iter.C.step = 500L,
   penalty.factor = rep(1.0, ncol(X)),
   warn = TRUE, bdp = 0.25,
   # breakdown points (between
   # 0 = least robust and 0.5 = most  robust)
   verbose = F){
  n = nrow(X)
  p = ncol(X)
  h = robustbase::h.alpha.n(0.5 + bdp, n, 1L)
  if (is.null(full.model)) {
    full.model = ncvreg.robust.full(X, y, bdp, verbose)
  }
  if (is.null(beta.hat.init)) {
    beta.hat.init = full.model$beta.hat
  }
  X = cbind(1, apply(X, 2, scale, center =
```

```
  s_Qn(X, mu.too = TRUE)[1],
  scale = Qn(X, finite.corr = FALSE)))
  residuals = as.vector(y - X %*% beta.hat.init)
  best = order(residuals, decreasing = FALSE)[1:h]
  beta = beta.hat.init
  for (iter in 1:max.iter.C.step) {
    fit = ncvreg::ncvfit(X = X[best, ], y = y[best], init = beta,
     penalty, gamma, alpha, lambda, eps, max.iter,
      penalty.factor, warn)
    dbeta = max(abs(as.vector(fit$beta) - beta))
    beta = as.vector(fit$beta)
    residuals = (y - X %*% matrix(beta, nrow = p + 1))^2
    best = order(residuals, decreasing = FALSE)[1:h]
    loss = sum(residuals[best])
    if (verbose) {
      cat("C-step iter = ", iter, ": ",
      "loss = ", loss, ",beta = (", paste(beta,
      collapse = ", "), ")\n", sep = "")
    }
    if ((dbeta < eps) || (iter == max.iter.C.step)) {
      res = fit
      break
    }
  }
  if (warn && (iter == max.iter.C.step)) {
    warning("Maximum number of C-steps achieved.
    #Consider increasing max.iter.C.step.")
  }

  res$n    = n
```

```
  res$best = best
  res$loss = loss
  res$residualss = residuals

  res$beta.hat = res$beta
  res[["beta"]] = NULL
  res$nu.hat = sum(abs(fit$beta[-1])
  >= .Machine$double.eps)
  res$w.hat  = rep(0.0, n)
  res$w.hat[best] = 1.0

  alpha = 1 - (h - res$nu.hat)/(n - res$nu.hat)
  res$sigma.hat = sqrt(res$loss/(h - res$nu.hat)/
                       pchisq(qchisq(1 - alpha, 1), 3))

  # for optimal tuning parameter selection
  res$BIC = res$nu.hat*(log(n)) + n*(log(2*pi*res$sigma.hat)
   + 1  - (res$nu.hat/h))
  res$AIC = 2*res$nu.hat +  n*(log(2*pi*res$sigma.hat)
  + 1 - (res$nu.hat/h))
  res$BIC = n*(res$sigma.hat/full.model$sigma.hat)^2 -
  n + 2*(res$nu.hat+1)
  return(res)
}

set.seed(1)
p = 40
n = 200

nrep = 100
```

```
for (ii in 1:length(p)) {
  for (jj in 1:length(n)) {
    Accuracy <- c()
    Sensitivity <- c()
    Specificity <- c()
    mse_rob = c()
    for (rep in 1:nrep) {
    np <- n[jj]*p[ii]
    X = matrix(rnorm( np, 0, 1),
    nrow = n[jj], ncol = p[ii])
    # Design Matrix
    # beta = c(runif(120, 2, 10),
    rep(0, 80))   # beta
    beta = c(runif(floor(.6*p[ii]), 2, 10),
    rep(0, ceiling(0.4*p[ii])))
    # beta
    eps = 0.1*matrix(rnorm(n[jj], 0, 1),
    nrow = n[jj], ncol = 1)
    # random error
    y = X %*% matrix(beta, nrow = p[ii],
    ncol = 1) + eps    # response variable

    noise = rep(quantile(y, 0.95)*10, floor(0.1*length(y)))
    y[1:floor(0.1*length(y))] = sample(y,
     floor(0.1*length(y)), replace = F) + noise


# -----------------
# "Homotopy" path
# -----------------
```

```
full.model = ncvreg.robust.full(X = apply(X, 2, scale,
center =s_Qn(X, mu.too = TRUE)[1], scale = Qn(X,
   finite.corr = FALSE)), y = y)
lambda.grid = exp(seq(from =
log(0.0001), to = log(0.99), length.out = 100L))
BIC.grid = rep(0.0, 100L)
nu.hat.grid = rep(0.0, 100L)
beta.grid <- rep(0.0, 100L)
beta.hat = full.model$beta.hat

BIC.opt = Inf
lambda.opt = NULL
beta.hat.opt = NULL
nu.hat.opt = NULL

for (i in 1:length(lambda.grid)) {
  lambda = lambda.grid[i]
  fit = ncvreg.robust(X = apply(X, 2, scale,
  center =   s_Qn(X, mu.too = TRUE)[1], scale =
  Qn(X, finite.corr = FALSE)),
 y = y, full.model = full.model, beta.hat.
 init = beta.hat,  penalty = "lasso",
 lambda = lambda, verbose = FALSE)
  beta.grid = fit$beta.hat[i]
  beta.hat = fit$beta.hat
  BIC.grid[i] = fit$BIC
  nu.hat.grid[i] = fit$nu.hat

  if (fit$BIC < BIC.opt) {
    BIC.opt = fit$BIC
```

```
    lambda.opt = lambda
    beta.hat.opt = fit$beta.hat
    nu.hat.opt = fit$nu.hat
  }
}

TP <- sum((beta==0) & (beta.hat.opt[-1]==0)) # TP
  FP <- sum((beta!=0) & (beta.hat.opt[-1]==0)) # FP
  FN <- sum((beta ==0) & (beta.hat.opt[-1]!=0)) # FN
  TN <- sum((beta!=0) & (beta.hat.opt[-1]!=0)) # TN

  conf_mat <- matrix(c(TP, FP, FN, TN),
                     ncol = 2, byrow = T)
  conf_mat <- as.table(conf_mat)
  rownames(conf_mat)<-c("TRUE", "FALSE")
  colnames(conf_mat)<-c("TRUE", "FALSE")

  # computing the metrics from confusion matrix
  Accuracy <- append(Accuracy, (TP+TN)/(TP+TN +FP+FN))
  Sensitivity <- append(Sensitivity, TP/(TP+FN) )
  Specificity <- append(Sensitivity, TN/(TN+FP) )
  mse_rob <- append(mse_rob, (beta - beta.hat.opt[-1])^2)
    }
  }

  matrix(1:4, nrow=2, byrow = T)
  cat("\n\n")

  print(mean(Accuracy))
  print(mean(Sensitivity))
```

```
  print(mean(Specificity))

}

mean(mse_rob)
```

Curriculum Vitae

Reagan Kesseku was born on March 23, 1992; the fourth child of Nicholas and Gladys Asiamah. He graduated from Potsin Ahmadiyya Senior High School in the Central of Ghana in 2011 where he studied General science and won five academic awards including the overall best academic student. He then entered the University of Ghana in 2012, where he obtained his Bachelor of Science degree in Statistics with mathematics and graduated in 2016 with First Class Honors. He was selected as a Teaching Assistant at the Department of Statistics of the University of Ghana where he served from 2016 - 2017. During that period, Reagan actively taught and assisted in Elementary Statistical Methods and Actuarial Statistics.
He later joined a team in 2018 in the position of a script writer and editor to develop an e-learning platform (Nikasemo) for Numerical methods in Ghana. He enrolled in Graduate School of The University of Texas at El Paso (UTEP) in 2019 where he pursued his Masters degree in Statistics. While pursuing his Masters, he worked in the Department of Mathematical Sciences of UTEP as a Graduate Teaching Assistant. He undertook his thesis work on the topic *Robust Variable Selection in Multiple Linear Regression via Penalized Least Trimmed Squares* under the supervision of Dr. Michael Pokojovy. Reagan was selected as the Graduate Student Marshall of Students for the College of Science during the 2021 Spring commencement of UTEP due to his outstanding academic performance. He was then admitted to the Ph.D program in Data Science with a Doctoral Excellence Fellowship award for Fall 2021 at UTEP.

Email address: bigcogent@gmail.com