Abstract

The shrinkage methods such as Lasso and Relaxed Lasso introduce some bias in order to reduce the variance of the regression coefficients in multiple linear regression models. One way to reduce bias after shrinkage of the coefficients would be to apply ordinary least squares to the subset of predictors selected by the shrinkage method used. We extensively studied this idea in this work and developed a new variable selection algorithm. We named this technique OLSAVS (Ordinary Least Squares After Variable Selection). We have implemented the OLSAVS algorithms in R. Simulations were used to illustrate that the new method is able to produce better predictions with less bias for various error distributions. We compare the OLSAVS method with a few widely used shrinkage methods in terms of their achieved test root mean square error and bias.
Acknowledgments

My deepest gratitude goes to the Appalachian State University mathematics department for providing me with the knowledge to accomplish a milestone such as this. I wish to extend my gratitude to my thesis director, Dr. Hasthika Rupasinghe, for his continuous support and guidance throughout this thesis. Without his help, this would not be possible. An additional sincere thanks to my second reader, Dr. Lasanthis Watagoda, for taking the time out of her schedule to analyze this paper. Furthermore, I wish to acknowledge my honors director, Dr. William J. Cook, for providing me with this opportunity to express my knowledge.

Finally, I would like to extend my sincere gratitude to all of my family and friends for all the support received throughout this process.
1 Introduction

Suppose we have a response predictor $Y_i$ and $p$ distinct predictors, then our multiple linear regression model (MLR) takes the form of

$$Y_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \epsilon$$

(1)

where $X_j$ represents each predictor and $\beta_j$ represents the corresponding relationship between the variable and response, with $\beta_0$ representing the intercept. MLR is an important tool for this study and helps lay a foundation for more complex models.

1.1 MLR model in matrix notation

MLR introduced in equation (1) can oftentimes become lengthy depending on the number of predictors. Therefore, using matrix notation, we can write the multiple linear regression model in the form of $Y = X\beta + \epsilon$ according to [8]. Analyzing this form,

$$Y_{n\times1} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}$$

(2)

Where $Y$ is a $n \times 1$ vector, and $n$ is the sample size. The matrix $X$ below,

$$X_{n\times p} = \begin{bmatrix} 1 & X_{11} & X_{12} & \ldots & X_{1,p-1} \\ 1 & X_{21} & X_{22} & \ldots & X_{2,p-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & X_{n1} & X_{n2} & \ldots & X_{n,p-1} \end{bmatrix}$$

(3)

is a $n \times p$ matrix of predictors. Then, $\beta$,
is an unknown $p \times 1$ vector of coefficients, and $\epsilon$,

$$
\beta_{p \times 1} = \begin{bmatrix}
\beta_0 \\
\beta_1 \\
\vdots \\
\beta_{p-1}
\end{bmatrix}
$$

is an unknown $p \times 1$ vector of coefficients, and $\epsilon$,

$$
\epsilon_{n \times 1} = \begin{bmatrix}
\epsilon_1 \\
\epsilon_2 \\
\vdots \\
\epsilon_n
\end{bmatrix}
$$

is $n \times 1$ vector representing the random error in each predictor.

Note that $Y$ and $\epsilon$ are the same as for simple linear regression. The $\beta$ vector contains additional regression parameters, and the $X$ matrix contains a column of 1’s as well as a column of $n$ observations for each $p - 1$ explanatory variable in the regression model. Additionally, the row subscript for each element in $X_{ik}$ in the matrix, $X$ represents the trial/case, and the column subscript identifies the $X$ variables. See [1].

### 1.2 Estimation of Regression Coefficients

This section closely follows [4] and covers the theory for estimating simple linear and MLR regression model coefficients.

#### 1.2.1 Estimating simple linear regression model coefficients

Consider the simple linear regression model where there is only one predictor variable and this function is linear, we can describe this model below,

$$
Y_i = \beta_0 + \beta_1 X_i + \epsilon_i
$$
Where,

\( Y_i \) is the value we obtain of the response variable in the \( i \)th trial

\( \beta_0 \) and \( \beta_1 \) are parameters that are defined in section 1.2 through the least squares method

\( X_i \) is a known constant that represents the value of the predictor variable in the \( i \)th trial.

\( \epsilon_i \) represents the random error in the mean with mean 0 and variance \( \sigma^2 \)

Additionally we can expand equation (6) below,

\[
\begin{align*}
Y_{n \times 1} &= \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} \\
&= \begin{bmatrix} \beta_0 + \beta_1 X_1 \\ \vdots \\ \beta_0 + \beta_1 X_n \end{bmatrix}
\end{align*}
\]  
(7)

Where \( Y \) is a \( n \times 1 \) vector with \( n \) being the number of response variables. Next, we look at the matrix \( X \),

\[
X_{n \times 2} = \begin{bmatrix} 1 & X_1 \\ 1 & X_2 \\ \vdots & \vdots \\ 1 & X_n \end{bmatrix}
\]  
(8)

Following this we have vectors for \( \beta \) and \( \epsilon \) respectively,

\[
\begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}
\]  
(9)

\[
\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}
\]  
(10)
In order to perform any kind of statistical testing, we must first understand and describe the regression coefficients. These coefficients are parameters in the regression model and are typically described as $\beta_0$ and $\beta_1$. The $\beta_1$ represents the slope of the regression line and $\beta_0$ represents the y-intercept in the line.

However, in many cases, the values for these two parameters are not known. Therefore, we must estimate these values. We will estimate these values through either observational data or experimental data.

In order to estimate values for $\beta_0$ and $\beta_1$, we will use the least squares method. By using observations in a data set $(X_i, Y_i)$, we can build a formula that considers the deviation of $Y_i$ from its expected value,

$$Y_i - (\beta_0 + \beta_1 X_i)$$

Now, we can build off equation (11) and consider the sum of $n$ squared deviations. We will use $Q$ to define this,

$$Q = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_i)^2$$

In equation (12), we will estimate $\beta_0$ and $\beta_1$ with $b_0$ and $b_1$, which are values that help minimize $Q$ for $n$ sample observations.

The idea for least squares is to find the estimate values, $b_0$ and $b_1$, for the parameters that help minimize $Q$. We can find these estimates in two ways, through numerical search procedures or analytical procedures. By using the analytical procedure, we can show that $b_0$ and $b_1$ that minimize $Q$ are given by the following,

$$\sum Y_i = nb_0 + b_1 \sum X_i$$

$$\sum X_i Y_i = b_0 \sum X_i + b_1 \sum X_i^2$$

where these two equations are known as normal equations and $b_0$ and $b_1$ are known as the point estimators for $\beta_0$ and $\beta_1$. Now, we can solve equations (13) and (14) for $b_0$ and $b_1$. 
\[ b_1 = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sum (X_i - \bar{X})^2} \]  
(15)

\[ b_0 = \frac{1}{n} (\sum Y_i - b_1 \sum X_i) = \bar{Y} - b_1 \bar{X} \]  
(16)

Here, \( \bar{X} \) and \( \bar{Y} \) are the means of the \( X_i \) and \( Y_i \) observations.

### 1.2.2 Estimating MLR model coefficients

Extending equation (12) using the MLR model in equation (1) we get;

\[ Q = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_{i1} - \ldots - \beta_{p-1} X_{i,p-1})^2 \]  
(17)

In this equation, the least squares estimators are \( \beta_0, \beta_1, \ldots, \beta_{p-1} \), which help minimize \( Q \). However, the least squares estimated regression coefficients \( b_0, b_1, \ldots, b_{p-1} \), where \( b, \)

\[ b_{p \times 1} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{p-1} \end{bmatrix} \]  
(18)

where the least squares normal equations for the general linear regression model are,

\[ X'Xb = X'Y \]  
(19)

and the least squares estimators are,

\[ b_{p \times 1} = (X'X)^{-1}X'Y \]  
(20)

### 1.2.3 Fitted Values and Residuals

Let \( \hat{Y} \) be a vector of fitted values and the vector of residuals \( e_i = Y_i - \hat{Y}_i \) be represented by \( e \):
Where the fitted values are represented by,

$$\hat{Y}_{n \times 1} = Xb$$  \hspace{1cm} (23)$$

since,

\[
\begin{bmatrix}
\hat{Y}_1 \\
\hat{Y}_2 \\
\vdots \\
\hat{Y}_n
\end{bmatrix} =
\begin{bmatrix}
1 & X_1 \\
1 & X_2 \\
\vdots & \vdots \\
1 & X_n
\end{bmatrix}
\begin{bmatrix}
\beta_0 \\
\beta_1 \\
\vdots \\
\beta_n
\end{bmatrix} =
\begin{bmatrix}
b_0 + b_1 X_1 \\
b_0 + b_1 X_2 \\
\vdots \\
b_0 + b_1 X_n
\end{bmatrix}
\hspace{1cm} (24)$$

and the residual terms by,

$$e_{n \times 1} = Y - \hat{Y} = Y - Xb$$  \hspace{1cm} (25)$$

The residual value represents the difference between the observed response value and the value predicted by the specific model being used. Now we can define the residual sum of squares (RSS) as follows,
\[ \text{RSS} = e_1^2 + e_2^2 + \ldots + e_n^2 \]  
\[(26)\]

or expanded as,

\[ \text{RSS} = (y_1 - \hat{\beta}_0 + \hat{\beta}_1 X_1)^2 + (y_2 - \hat{\beta}_0 + \hat{\beta}_1 X_2)^2 + \ldots + (y_n - \hat{\beta}_0 + \hat{\beta}_1 X_n)^2 \]  
\[(27)\]

where the least squares method mentioned previously helps determine \( \beta_0 \) and \( \beta_1 \) to minimize the RSS.

### 1.3 Variable selection

Variable selection is the search for a subset of predictor variables that can be deleted with little loss of information, and so the model with the remaining predictors is useful for prediction. The variable selection methods being looked at in this study are the following, Ridge regression, Lasso, Relaxed Lasso, and Elastic Net models.

#### 1.3.1 Ridge regression

According to [3], similar to least squares, ridge regression finds the best coefficients to make the \( \text{RSS} \) small. Ridge Regression coefficient estimates \( \hat{\beta}^R \), are values that minimize,

\[ \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2 \]  
\[(28)\]

Where \( \lambda \geq 0 \) is a tuning parameter. The second term in the equation,

\[ \lambda \sum_{j=1}^{p} \beta_j^2 \]  
\[(29)\]

is known as the shrinkage penalty. This penalty value is small when \( \beta_1, \ldots, \beta_p \) are close to zero, sending the \( \beta_j \) values to zero but never reaching zero. For this reason, ridge regression includes all predictors \( p \) in the model.
1.3.2 Lasso

Another method for estimating $\beta$ is known as Lasso regression, which contains qualities similar to ridge regression models. Lasso improves on one specific that ridge does not. Lasso sets variables to zero, unlike ridge. See [7]. The Lasso coefficients,

\[ \hat{\beta}^L_\lambda, \text{minimizes the quantity} \]

\[ RSS + \lambda \sum_{j=1}^{p} |\beta_j| \]  \hspace{1cm} (31)

Where RSS is given in equation (26).

1.3.3 Relaxed Lasso

Relaxed Lasso, according to [5] controls model selection and shrinkage estimation by two separate parameters $\lambda$ and $\phi$. The Relaxed Lasso estimator is defined for $\lambda \in [0, \infty)$ and $\phi \in (0, 1]$ as

\[ \hat{\beta}^{\lambda,\phi} = \arg\min_{\beta} n^{-1} \sum_{i=1}^{n} (Y_i - X_i^T \{ \beta \cdot 1_{M_{\lambda}} \})^2 + \phi \lambda |\beta|_1 \]  \hspace{1cm} (32)

Where $M_{\lambda}$ is the indicator function on the set of variables $M \subseteq \{1, \ldots, p\}$ so that for all $k \in \{1, \ldots, p\}$

\[ \beta \cdot 1_{M_{\lambda}} = \begin{cases} 0 & k \notin M_{\lambda} \\ \beta_k & k \in M_{\lambda} \end{cases} \]

Relaxed Lasso is a generalization from Lasso. Unlike Lasso, the Relaxed Lasso takes into account two regularization parameters. This method produces not only Lasso solutions but also a least-squares solution that controls the regularization applied to the selected variables.
1.3.4 Elastic Net

As explained in [10], given data \((y, X)\) and penalty parameters \((\lambda_1, \lambda_2)\) and some augmented data \((y^*, X^*)\) the naive Elastic Net model takes the following form,

\[
\hat{\beta}^* = \arg\min_{\beta^*} |y^* - X^* \beta^*|^2 + \frac{\lambda_1}{\sqrt{1 + \lambda_2}} |\beta|_1 
\]  

(33)

This form of the Elastic Net solves a Lasso-type problem, where the corrected Elastic Net estimates \(\hat{\beta}\) are defined as,

\[
\hat{\beta}\text{(elastic net)} = \sqrt{1 + \lambda_2} \hat{\beta}^*
\]  

(34)

and \(\hat{\beta}\text{(naive elastic net)} = 1/\sqrt{1 + \lambda_2} \hat{\beta}^*\) then,

\[
\hat{\beta}\text{(elastic net)} = (1 + \lambda_2) \hat{\beta}\text{(naive elastic net)}.
\]  

(35)

This then shows that our Elastic Net model takes form from a rescaled naive Elastic Net. Additionally, the scaling factor \((1 + \lambda_2)\) comes from a decomposition of the Ridge operator. However, this is outside the scope of this paper but does give insight into how the Elastic Net model includes Ridge regression as well.

Lastly, Elastic Net (Enet) regression is another method of variable selection. This model takes into account two regularization parameters. These two parameters come for the Lasso and Ridge models. Therefore, Elastic Net does produce variables equal to zero. In many cases, Elastic Net is better to use than Lasso or Ridge, because in this scenario you do not have to choose between the two.

2 Method

2.1 Assessing Model Accuracy

In statistical learning, it is key to determine the model’s accuracy with some value and decide which method works better. This metric is called the Mean Squared Error (MSE), and we will use this measure to represent the extent to which the predicted response value is close to the true response value for a
Figure 1: Left: Data simulated from f, shown in black. We estimate f with the following: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves). Right: Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown on the left.

The MSE is given by the formula,

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$$

(36)

Where $\hat{f}(x_i)$ is the prediction that $\hat{f}$ gives for the $i$th observation. This value will be small if the predicted values are close to the true values. See [1].

Although we would like this value to be small for training data, the key value will come from the test MSE as shown below in figure 1,

Here we see a data set fitted with three different models. Based on the first judgment it appears the green cubic spline fits the data best. However, on the right, we can see how the training and test MSE values compare. Notice that the green square represents the respective cubic line in the left image. Here we can see that although this line has a very low training MSE, the test MSE is noticeably high. This specific scenario is known as over-fitting and happens when we a model fits the data too closely. In this particular case, it is actually the blue line that fits the model the best, and we can look at our test MSE to help prove this.
Figure 2: A display of the validation set approach provided in [1]. A set of n observations are randomly split into a training set (shown in blue, containing observations 7, 22, and 13, among others) and a validation set (shown in beige, and containing observation 91, among others). The statistical learning method is fit on the training set, and its performance is evaluated on the validation set.

2.2 Cross Validation

Thus far, we have learned how to predict model accuracy, and what variable helps us determine this measure. Both the training MSE and test MSE are useful pieces of information, but we determined that the test MSE would be the most beneficial in determining model accuracy. However, in many cases, we are unable to perform a test MSE calculation on a designated test set. Instead, we use methods to help gather an estimate for this error rate by using the training data. In particular, we will hold out a subset of the training observations from the fitting process, and then apply the statistical method to those held-out observations. We will focus on two main forms of cross-validations, “Leave One Out Cross Validation (LOOCV)” and “k-Fold Cross-Validation”.

2.2.1 The Validation Set Approach

When looking for a test error associated with fitting a statistical learning method on a specific set of observations, the validation approach is a simple strategy for this. This approach involves randomly dividing the observations into two sets, a training set and a validation set (hold-out set). We can see how this split is performed in figure 2.

We fit the model on the training set and then use the fitted model to predict the observations in the validation set. The MSE produced from the estimated validation set provides an estimate for the test error rate.

We can also see how the validation set works graphically. By using a data set in R, we can show
Figure 3: The validation set approach was used on a data set to estimate the test error for a response variable using polynomial functions of an explanatory variable. Left: Validation error estimates for a single split into training and validation data sets. Right: The validation method was repeated ten times, each time using a different random split of the observations into a training set and a validation set. This shows the variability in the test MSE. See [1]

how the validation set error rates result from fitting various regression models on the training set. Then, evaluate their performance on the validation set, using MSE as a measure of test set error. In figure 3, on the left, we can see that the validation set error is noticeably smaller for a quadratic fit than a linear fit. However, increasing the fit to the cubic term results in a larger MSE. This showing that adding a higher polynomial term does not result in a better fit. Likewise, the figure on the right shows the same dataset performing ten random splits of the observations into training and test sets. Still, in every split, we can see the drastic drop in test set error from a linear fit to a quadratic fit. Furthermore, showing that higher polynomial fits do not benefit the model much.

Although this approach is beneficial in many ways, there are a couple of drawbacks. These include high variability and over-fitting. Since the method is looking to perform the best on the data set, it may try to include too many variables and overestimate. See [1] for more details.

2.2.2 Leave One Out Cross Validation (LOOCV)

Similar to our previous validation approach, Leave One Out Cross Validation (LOOCV), attempts to fix some of the issues mentioned for the validation set approach. In this approach, we will still split the data into training and test sets. However, instead of making the two sets comparable in size, we will now remove one observation and allow it to be the validation set with the remaining sets making up the
Figure 4: A display of LOOCV provided in [1]. A set of n data points is repeatedly split into a training set (blue) containing all but one observation, and a validation set that contains only that observation (beige). The test error is then estimated by averaging the MSE’s, n times. The first training set contains all but observation 1, the second training set contains all but observation 2, and so forth.

Unlike the previous validation approach and using a single estimate for the test MSE, the LOOCV estimate for the test MSE is the average of the n test error estimates shown in the formula,

\[ CV(n) = \frac{1}{n} \sum_{i=1}^{n} MSE_i \] (37)

According to [1], LOOCV, also has a couple of advantages over the validation set approach. First, this method is less-bias since LOOCV uses \( n - 1 \) training observation sets which nearly cover the entire data set. Secondly, LOOCV tends to not overestimate the test error rate as much as the validation set approach.

### 2.2.3 k-Fold Cross-Validation

The last validation approach we have is *k-fold CV*. This approach is still similar to the other by forming training and validation sets, but now these sets are split into \( k \) groups, or folds of the same size. The first fold will be treated as the validation set, where the remaining \( k-1 \) folds fit with our method. Figure 5 shows an example of a 5-fold cross-validation,
To test the accuracy for the $k$-1 approach, we compute the MSE for each held-out observation. Since we have $k$ folds, this operation will be produced $k$ times. The $k$-1 fold estimate is computed by averaging these values,

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^{k} MSE_i$$ (38)

Typically, we will only use $k = 5$ folds or $k = 10$ folds. This is due to the time it takes to compute each of these estimates. We can see how $k$ fold validation compares to simulated data through the figure 6.

Here we can see how both methods compare to the test MSE. Including one case where both underestimate the true MSE. Although this underestimate is shown, it is beneficial to see, because in some cases locating the minimal point for the estimated curve is used to compare with other methods to determine which is best.

### 2.3 Bias of a model

The bias for a statistical machine learning model can be described as systematic error that occurs in the model when the model draws incorrect assumptions of the data in the modeling process according
Figure 6: True and estimated test MSE for simulated data sets provided in [1]. The true test MSE is shown in blue, the LOOCV estimate is shown as a black dashed line, and the 10-fold CV estimate is shown in orange. The crosses indicate the minimum of each of the MSE curves.

to [9]. Bias is often referred to as the error that is introduced by approximating a real-life scenario. Additionally, bias in a model can also describe how closely our model follows our training data. Higher bias resembles a low-fitting model, low bias resembles a close-fitting model.

2.4 Variance of a model

Variance in a model is best described as a value in which the model changes when introduced to different values in the training set. Running the model on different training sets will result in different values for the variance, however, this value should not change too much between sets. Variance tends to correlate directly to complexity in a model. A model with high variability relates to a more complex model, whereas a less complex model tends to have low variability. See [1]

2.5 Bias-Variance Trade off

Referring back to the test MSE mentioned in 2.1, we can form an equation using the bias and variance to measure the expected test MSE. This equation is shown below,

\[
E(y_0 - \hat{f}(x_0))^2 = Var(\hat{f}(x_0)) + [Bias(\hat{f}(x_0))]^2 + Var(\epsilon)
\]  

(39)

Where \(E(y_0 - \hat{f}(x_0))^2\) defines the expected test MSE at \(X_0\). Equation (39) shows that in order to
make the test MSE small, two things are required. The model requires both low variance and low bias in order to obtain a low MSE.

When computing machine learning models, to determine the best model, Bias and Variance are typically two points that are examined. The goal is to get low bias and variance to create the best model for the data. However, one contradicts the other, and this is known as the bias-variance trade-off. When we get a model that has high variability, that model has low bias. In the same manner, a model with low variability has high bias. Determining a point where the model performs well for both bias and variability is one of the main goals for all learning models. The graphs in figure 7 help us understand this trade-off better.

Notice, in each of the three graphs, there is a distinct point where the test MSE is the smallest. At this point, there is a corresponding variance and bias level for each graph, giving us some sort of indication of what type of model is being performed.

### 2.6 A new method for model selection: OLSAVS

The shrinkage methods such as Lasso and Relaxed Lasso introduce some bias in order to reduce the variance of the regression coefficients. As briefly mentioned in [2], one way to reduce bias after shrinkage of the coefficients would be to apply ordinary least squares to the subset of predictors se-
lected by the shrinkage method used. In this work, we extensively study this idea to develop a new variable selection algorithm. We name this technique Ordinary Least Squares After Variable Selection (OLSAVS). We have implemented OLSA VS method in R. The set of functions can be found at https://hasthika.github.io/olsvspack.txt. The algorithm of the OLSA VS method is as follows:

Algorithm: Ordinary Least Squares After Variable Selection (OLSAVS)

Repeat: following steps with a different shrinkage method

1) Apply the first shrinkage method to \((Y_i, x_i)\) for \(i = 1, ..., n\).

2) Obtain the \(k\) non-zero predictors selected by the shrinkage method in 1)

3) Apply Ordinary Least Squares on the subset of \(k\) predictors obtained in 2)

Stop

4) Select a single best model using cross-validated prediction error, \(C_p\), (AIC), BIC, or adjusted \(R^2\)

3 Simulation

3.1 Simulation setup

We used R to generate \((Y_i, x_i)\) for \(i = 1, ..., n\). The regression parameters \(\beta\) were set to \((1, 1, ..., 1, 0, ..., 0)\) with \(k + 1\) ones, \(p-k-1\) zeroes where \(p\) is the total number of predictors, and \(k\) is the number of non-trivial predictors. Then for a given regression method, the regression coefficients, \(\hat{\beta}\) were obtained using the proposed method. This process was repeated 5000 times (runs). For each run, the difference between the regression parameters \(\beta\) and the regression coefficients, \(\hat{\beta}\) were obtained using the Minkowski distance. The average difference (Diff) was calculated by averaging all 5000 runs. The root MSE was also obtained using a test of observations for each run that was averaged of the 5000 runs (TRMSE). We used \(p = n/5, n/2, \) or \(n - 1\) as the total number of predictors and \(k = 1, 19, \) or \(p - 1\) as the number of non-trivial predictors in the model. As per the easiness of coding we used the relation \(\text{cor}(x_i, x_j) = \rho = (2\psi + (p-3)\psi^2)/(1 + (p-2)\psi^2)\), for \(i \neq j\), where, \(x_i, x_j\) are non-trivial predictors. As \(\psi\) increases the correlation between preceptors, \(\rho\) grows. We used \(\psi = 0, 0.3 \) or \(0.9\). We considered five error distributions with zero mean.
1. The first distribution we used is the \( N(0, 1) \) which is commonly used in simulation studies.

2. As the second error distribution, we considered \( t_3 \), one of the heavy-tailed distributions.

3. We also considered not very commonly used in simulations but found in many real-life situations, a non-symmetric error distribution \( EXP(1) - 1 \).

4. We also used \( \text{uniform}(-1, 1) \), and

5. \( 0.9N(0, 1) + 0.1N(0, 100) \).

The simulation study was conducted in \( R \). See [6].

### 3.2 Simulation results

![Figure 8: Box plots for table 1 showing simulation results for OLSAVS vs. Lasso regression with error type 1.](image)

For figure 8, we assembled box plots to compare the TRMSE results in table 1. For each of the simulations tables, we will assemble two plots with \( n = 100 \) or \( n = 200 \) to compare the results. Here in figure 8, we see that in either sample size, OLSAVS has a smaller median TRMSE and lower variation in the results. However, it appears for this simulation, for Lasso with a normal error, that when the sample size increases, so do the variation in the plot.
<table>
<thead>
<tr>
<th>n</th>
<th>p</th>
<th>k</th>
<th>ψ</th>
<th>TRMSE</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OLSAVS</td>
<td>Lasso</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>1.0478</td>
<td>1.021</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>1</td>
<td>0.3</td>
<td>1.0417</td>
<td>1.0199</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>1</td>
<td>0.9</td>
<td>1.0083</td>
<td>1.0057</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>19</td>
<td>0</td>
<td>1.1112</td>
<td>1.1107</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>19</td>
<td>0.3</td>
<td>1.1112</td>
<td>1.1377</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>19</td>
<td>0.9</td>
<td>1.1543</td>
<td>2.3451</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>1.0916</td>
<td>1.0369</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>1</td>
<td>0.3</td>
<td>1.0708</td>
<td>1.0303</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>1</td>
<td>0.9</td>
<td>1.0142</td>
<td>1.0059</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>49</td>
<td>0</td>
<td>1.3973</td>
<td>1.4014</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>49</td>
<td>0.3</td>
<td>1.5522</td>
<td>4.353</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>49</td>
<td>0.9</td>
<td>2.1118</td>
<td>9.4775</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>1</td>
<td>0</td>
<td>1.0386</td>
<td>1.0175</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>1</td>
<td>0.3</td>
<td>1.0316</td>
<td>1.0157</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>1</td>
<td>0.9</td>
<td>1.0108</td>
<td>1.005</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>39</td>
<td>0</td>
<td>1.117</td>
<td>1.119</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>39</td>
<td>0.3</td>
<td>1.117</td>
<td>2.5606</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>39</td>
<td>0.9</td>
<td>1.6985</td>
<td>6.8533</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>1</td>
<td>0</td>
<td>1.0691</td>
<td>1.0261</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>1</td>
<td>0.3</td>
<td>1.0605</td>
<td>1.0251</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>1</td>
<td>0.9</td>
<td>1.0184</td>
<td>1.0045</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>99</td>
<td>0</td>
<td>1.4495</td>
<td>1.4561</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>99</td>
<td>0.3</td>
<td>3.166</td>
<td>11.4553</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>99</td>
<td>0.9</td>
<td>3.7274</td>
<td>27.9646</td>
</tr>
</tbody>
</table>

Table 1: TRMSE and difference values for Lasso for $e_i \sim N(0, 1)$
For table 1 we ran simulations to compare the OLSAVS method to Lasso regression. However, in this particular simulation, we compared OLSAVS to a Lasso regression with a normal distribution error type that has a mean of 0 and a variance of 1. Here, and for the remaining simulations, we will compare the methods for $n = 100, 200$.

In table 1 the first trend we can begin to see occurs in the TRMSE for the simulations. In the beginning, Lasso outperforms the method being proposed, and this occurs for the first trial where the non-trivial predictors ($k$) are low. However, as this value increases for both the predictors and the correlation among these predictors respectively, the OLSAVS method will eventually overtake the Lasso regression for the TRMSE value. This trend will continue throughout table 1, but as the sample size and non-trivial predictors increase, the distance between the OLSAVS and Lasso values gets noticeably larger. In comparing the two, the OLSAVS stays consistent throughout for the TRMSE, where Lasso has a lot of variability for this value.

Although the OLSAVS out-performed Lasso for certain cases, we need to analyze the table for the difference (Diff) values computed using Minkowski distance. There is not any noticeable trend unlike the TRMSE, but we can see that for small non-trivial predictors and high correlation values, the Lasso regression either comes close or betters OLSAVS in the difference value here. Other than this certain case, the new method significantly bettered the Lasso in table 1.

Figure 9 shows two simulation plots pulled from table 2. Looking at two cases, each varying in sample size, we see that our method does edge out Lasso regression with the error type being from a t-distribution. However, unlike 8 the variation in the plot now decreases as expected when the error type increases.

Table 2 is a similar simulation approach to the previous table 1, however the error type in table 2 changes for the Lasso regression. The error type for table 2 will be a t-distribution with zero mean.

Analyzing the TRMSE for table 2, a trend begins to form as the number of non-trivial predictors increases. The Lasso regression will outperform OLSAVS every time that the non-trivial predictor $k = 1$. However, once we go from one extreme to another in our simulations, the TRMSE favors our method once the non-trivial predictors are equal to $p - 1$. Additionally, as the correlation between the non-trivial predictors increases, we can see a large distance increase between the two methods.

For the difference in table 2, it is hard to establish any consistent trends throughout. However, it is notable that when $n = 100, p = 50$, and $k = 1, 49$ the Lasso slightly does better than the new method.
Table 2: TRMSE and difference values for Lasso for $e_i \sim t_3$

<table>
<thead>
<tr>
<th>n</th>
<th>p</th>
<th>k</th>
<th>$\psi$</th>
<th>TRMSE OLSAVS</th>
<th>Lasso</th>
<th>Diff OLSAVS</th>
<th>Lasso</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>1.3677</td>
<td>1.3362</td>
<td>0.0260</td>
<td>0.1882</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>1</td>
<td>0.3</td>
<td>1.3551</td>
<td>1.3317</td>
<td>0.1497</td>
<td>0.2031</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>1</td>
<td>0.9</td>
<td>1.3178</td>
<td>1.3169</td>
<td>0.6539</td>
<td>0.7482</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>19</td>
<td>0</td>
<td>1.4441</td>
<td>1.4442</td>
<td>0.0047</td>
<td>0.0130</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>19</td>
<td>0.3</td>
<td>1.4441</td>
<td>1.4590</td>
<td>0.0061</td>
<td>0.0142</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>19</td>
<td>0.9</td>
<td>1.4487</td>
<td>2.4245</td>
<td>1.8395</td>
<td>7.2843</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>1.3704</td>
<td>1.3311</td>
<td>0.2075</td>
<td>0.2549</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>1</td>
<td>0.9</td>
<td>1.3060</td>
<td>1.3015</td>
<td>0.9523</td>
<td>0.6991</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>49</td>
<td>0</td>
<td>1.8360</td>
<td>1.8418</td>
<td>0.0076</td>
<td>0.0076</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>49</td>
<td>0.3</td>
<td>1.9877</td>
<td>4.4636</td>
<td>0.0245</td>
<td>0.0245</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>49</td>
<td>0.9</td>
<td>2.2778</td>
<td>9.6242</td>
<td>8.8102</td>
<td>43.9438</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>1</td>
<td>0</td>
<td>1.3453</td>
<td>1.3165</td>
<td>0.0351</td>
<td>0.1550</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>1</td>
<td>0.3</td>
<td>1.3324</td>
<td>1.3163</td>
<td>0.1351</td>
<td>0.1707</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>1</td>
<td>0.9</td>
<td>1.3075</td>
<td>1.3033</td>
<td>0.8243</td>
<td>0.5671</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>39</td>
<td>0</td>
<td>1.4351</td>
<td>1.4355</td>
<td>0.0036</td>
<td>0.0107</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>39</td>
<td>0.3</td>
<td>1.4351</td>
<td>2.5826</td>
<td>0.0048</td>
<td>0.0376</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>39</td>
<td>0.9</td>
<td>1.8981</td>
<td>6.7108</td>
<td>5.7510</td>
<td>32.8242</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>1</td>
<td>0</td>
<td>1.3837</td>
<td>1.3189</td>
<td>0.0460</td>
<td>0.1847</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>1</td>
<td>0.3</td>
<td>1.3559</td>
<td>1.3153</td>
<td>0.1774</td>
<td>0.2115</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>1</td>
<td>0.9</td>
<td>1.3096</td>
<td>1.2998</td>
<td>1.1920</td>
<td>0.2210</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>99</td>
<td>0</td>
<td>1.8397</td>
<td>1.8487</td>
<td>0.0058</td>
<td>0.0147</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>99</td>
<td>0.3</td>
<td>3.3891</td>
<td>11.4710</td>
<td>0.1501</td>
<td>0.4976</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>99</td>
<td>0.9</td>
<td>3.8424</td>
<td>27.4098</td>
<td>17.2990</td>
<td>93.9071</td>
</tr>
</tbody>
</table>

Table 2: TRMSE and difference values for Lasso for $e_i \sim t_3$
Overall, the new method outperforms Lasso by a large majority.

Figure 10 shows boxplots gathered from two simulations in table 3. In the two plots shown, we see yet again that OLSAVS edges out the Lasso and has a much shorter variation in the box plot. Lasso with an exponential error does compete in the $n = 100$ plot but then has a much larger gap when the sample size is increased.

In table 3 we will compare our method to Lasso again, but this time we will compare to the Lasso regression with the error types being from the exponential distribution. Again, with a mean of zero, will analyze the TRMSE and Diff for both methods.

Upon analyzing the TRMSE for both methods, we can see a very similar trend occurring. Much like our simulations from table 2, as the non-trivial predictors are increased to $p - 1$ we can see our method produces a lower TRMSE than Lasso. Additionally, the same gap occurs when the correlation is increased for the $p - 1$ predictors as well.

The differences in table 3 appear to have no consistent trends. OLSAVS seems to dominate for the majority of the difference values. Still, much like the previous simulations, there is this interesting difference that occurs when the correlation between predictors is equal to 0.9 and $k = 1$. At this particular point, Lasso does better than our method, but only at this point.
<table>
<thead>
<tr>
<th>n</th>
<th>p</th>
<th>k</th>
<th>$\psi$</th>
<th>TRMSE OLSAVS</th>
<th>Lasso</th>
<th>Diff OLSAVS</th>
<th>Lasso</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>1.0589</td>
<td>1.0366</td>
<td>0.0208</td>
<td>0.1416</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>1</td>
<td>0.3</td>
<td>1.0287</td>
<td>1.0099</td>
<td>0.1123</td>
<td>0.1533</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>19</td>
<td>0</td>
<td>0.9913</td>
<td>0.9900</td>
<td>0.6264</td>
<td>0.6519</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>19</td>
<td>0.3</td>
<td>1.1055</td>
<td>1.1063</td>
<td>0.0033</td>
<td>0.0118</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>19</td>
<td>0.9</td>
<td>1.1055</td>
<td>1.1583</td>
<td>0.0046</td>
<td>0.0148</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>19</td>
<td>0.9</td>
<td>1.1335</td>
<td>2.3835</td>
<td>1.5415</td>
<td>7.5833</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>1.0948</td>
<td>1.0456</td>
<td>0.0232</td>
<td>0.1804</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>1</td>
<td>0.3</td>
<td>1.0767</td>
<td>1.0414</td>
<td>0.1689</td>
<td>0.2039</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>1</td>
<td>0.9</td>
<td>1.0263</td>
<td>1.0188</td>
<td>0.9343</td>
<td>0.5814</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>49</td>
<td>0</td>
<td>1.4078</td>
<td>1.4105</td>
<td>0.0045</td>
<td>0.0147</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>49</td>
<td>0.3</td>
<td>1.5744</td>
<td>4.3752</td>
<td>0.0153</td>
<td>0.0520</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>49</td>
<td>0.9</td>
<td>2.1137</td>
<td>9.5345</td>
<td>8.8138</td>
<td>44.0251</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>1</td>
<td>0</td>
<td>1.0298</td>
<td>1.0071</td>
<td>0.0202</td>
<td>0.1201</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>1</td>
<td>0.3</td>
<td>1.0202</td>
<td>1.0037</td>
<td>0.1063</td>
<td>0.1331</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>1</td>
<td>0.9</td>
<td>0.9969</td>
<td>0.9925</td>
<td>0.7664</td>
<td>0.4067</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>39</td>
<td>0</td>
<td>1.1034</td>
<td>1.1048</td>
<td>0.0026</td>
<td>0.0097</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>39</td>
<td>0.3</td>
<td>1.1034</td>
<td>2.4771</td>
<td>0.0035</td>
<td>0.0380</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>39</td>
<td>0.9</td>
<td>1.7076</td>
<td>6.7040</td>
<td>5.8761</td>
<td>32.8648</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>1</td>
<td>0</td>
<td>1.0522</td>
<td>1.0122</td>
<td>0.0371</td>
<td>0.0371</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>1</td>
<td>0.3</td>
<td>1.0356</td>
<td>1.0101</td>
<td>0.1402</td>
<td>0.1657</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>1</td>
<td>0.9</td>
<td>1.0067</td>
<td>0.9953</td>
<td>0.9636</td>
<td>0.0896</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>99</td>
<td>0</td>
<td>1.4138</td>
<td>1.4192</td>
<td>0.0037</td>
<td>0.0131</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>99</td>
<td>0.3</td>
<td>3.1658</td>
<td>11.2754</td>
<td>0.1451</td>
<td>0.4836</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>99</td>
<td>0.9</td>
<td>3.6705</td>
<td>27.4919</td>
<td>17.5606</td>
<td>93.9063</td>
</tr>
</tbody>
</table>

Table 3: TRMSE and difference values for Lasso for $e_i \sim EXP(1) - 1$
Figure 10: Box plots for table 3 showing simulation results for OLSAVS vs. Lasso regression with error type 3.

Figure 11: Box plots for table 4 showing simulation results for OLSAVS vs. Lasso regression with error type 4.
<table>
<thead>
<tr>
<th>n</th>
<th>p</th>
<th>k</th>
<th>ψ</th>
<th>TRMSE OLSAVS</th>
<th>TRMSE Lasso</th>
<th>Diff OLSAVS</th>
<th>Diff Lasso</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>0.6087</td>
<td>0.5952</td>
<td>0.0112</td>
<td>0.0843</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>1</td>
<td>0.3</td>
<td>0.6024</td>
<td>0.5945</td>
<td>0.0662</td>
<td>0.0900</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>1</td>
<td>0.9</td>
<td>0.5905</td>
<td>0.5879</td>
<td>0.5773</td>
<td>0.4067</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>19</td>
<td>0</td>
<td>0.6360</td>
<td>0.6369</td>
<td>0.0023</td>
<td>0.0103</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>19</td>
<td>0.3</td>
<td>0.6360</td>
<td>0.9626</td>
<td>0.0031</td>
<td>0.0308</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>19</td>
<td>0.9</td>
<td>0.7599</td>
<td>2.3075</td>
<td>1.3762</td>
<td>7.8005</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>0.6265</td>
<td>0.5999</td>
<td>0.0177</td>
<td>0.1043</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>1</td>
<td>0.3</td>
<td>0.6169</td>
<td>0.5955</td>
<td>0.0943</td>
<td>0.1160</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>1</td>
<td>0.9</td>
<td>0.5917</td>
<td>0.5826</td>
<td>0.7318</td>
<td>0.2120</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>49</td>
<td>0</td>
<td>0.8221</td>
<td>0.8299</td>
<td>0.0028</td>
<td>0.0127</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>49</td>
<td>0.3</td>
<td>1.0467</td>
<td>4.3592</td>
<td>0.0121</td>
<td>0.0573</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>49</td>
<td>0.9</td>
<td>1.9089</td>
<td>9.4600</td>
<td>9.0251</td>
<td>44.0993</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>1</td>
<td>0</td>
<td>0.6078</td>
<td>0.5907</td>
<td>0.0127</td>
<td>0.0697</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>1</td>
<td>0.3</td>
<td>0.6016</td>
<td>0.5904</td>
<td>0.0604</td>
<td>0.0760</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>1</td>
<td>0.9</td>
<td>0.5886</td>
<td>0.5836</td>
<td>0.4935</td>
<td>0.1028</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>39</td>
<td>0</td>
<td>0.6506</td>
<td>0.6506</td>
<td>0.0013</td>
<td>0.0088</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>39</td>
<td>0.3</td>
<td>0.6506</td>
<td>2.5639</td>
<td>0.0018</td>
<td>0.0388</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>39</td>
<td>0.9</td>
<td>1.4729</td>
<td>6.8174</td>
<td>5.9630</td>
<td>33.0055</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>1</td>
<td>0</td>
<td>0.6130</td>
<td>0.5903</td>
<td>0.0123</td>
<td>0.0831</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>1</td>
<td>0.3</td>
<td>0.5967</td>
<td>0.5967</td>
<td>0.0898</td>
<td>0.0973</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>1</td>
<td>0.9</td>
<td>0.5902</td>
<td>0.5830</td>
<td>0.5646</td>
<td>0.0216</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>99</td>
<td>0</td>
<td>0.8125</td>
<td>0.8243</td>
<td>0.0021</td>
<td>0.0119</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>99</td>
<td>0.3</td>
<td>2.9615</td>
<td>11.3001</td>
<td>0.1460</td>
<td>0.4744</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>99</td>
<td>0.9</td>
<td>3.5458</td>
<td>27.3187</td>
<td>17.4181</td>
<td>93.9087</td>
</tr>
</tbody>
</table>

Table 4: TRMSE and difference values for Lasso for $e_i \sim \text{uniform}(-1, 1)$
For table 4 in our Lasso simulations, we look at a Lasso regression with a uniform distribution error type. Here in figure 11, the variation for each method is large, however, OLSAVS still has the smaller average TRMSE. Once the sample size increases, both the variation and the average TRMSE shrinks for each method, but the OLSAVS still maintains the advantage in each.

The last table we will analyze for Lasso regression is table 4. This table keeps the same ideas, but again our distribution for the errors is changed. We will look at a uniform distribution this time with zero mean.

Once again for our TRMSE simulations, the same trend appears between the OLSAVS and Lasso estimates as before. However, this time we get an output where the two functions are equal for observation 16. Compared to the other simulations, this Lasso regression stands out since the other simulations for this observation were larger than the OLSAVS value.

The difference for the simulations remains mostly the same as for our previous simulations for Lasso with an exponential distribution. Here our method outperforms Lasso again, with the same occurrence of only one certain scenario when the correlation between predictors is equal to 0.9 and $k = 1$.

Figure 12 shows our first boxplots for the Relaxed Lasso simulations with error type 1. Here in
Table 5: TRMSE and difference values for Relaxed Lasso for $e_i \sim N(0, 1)$

<table>
<thead>
<tr>
<th>n</th>
<th>p</th>
<th>k</th>
<th>$\psi$</th>
<th>TRMSE</th>
<th>Diff</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OLSAVS</td>
<td>Lasso</td>
<td>OLSAVS</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>1.05004</td>
<td>1.03955</td>
<td>0.01180</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>1</td>
<td>0.3</td>
<td>1.04307</td>
<td>1.03993</td>
<td>0.11183</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>1</td>
<td>0.9</td>
<td>1.00809</td>
<td>1.00586</td>
<td>0.63942</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>19</td>
<td>0</td>
<td>1.11518</td>
<td>1.11527</td>
<td>0.00315</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>19</td>
<td>0.3</td>
<td>1.11518</td>
<td>1.12323</td>
<td>0.00384</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>19</td>
<td>0.9</td>
<td>1.14518</td>
<td>1.44021</td>
<td>1.54564</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>1.10563</td>
<td>1.08454</td>
<td>0.02597</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>1</td>
<td>0.3</td>
<td>1.09096</td>
<td>1.08971</td>
<td>0.16601</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>1</td>
<td>0.9</td>
<td>1.03093</td>
<td>1.02483</td>
<td>0.95682</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>49</td>
<td>0</td>
<td>1.42151</td>
<td>1.42127</td>
<td>0.00415</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>49</td>
<td>0.3</td>
<td>1.57851</td>
<td>3.89696</td>
<td>0.01567</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>49</td>
<td>0.9</td>
<td>2.12296</td>
<td>4.80282</td>
<td>8.87674</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>1</td>
<td>0</td>
<td>1.03855</td>
<td>1.03304</td>
<td>0.02126</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>1</td>
<td>0.3</td>
<td>1.03295</td>
<td>1.03285</td>
<td>0.10517</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>1</td>
<td>0.9</td>
<td>1.01005</td>
<td>1.00655</td>
<td>0.78436</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>39</td>
<td>0</td>
<td>1.12531</td>
<td>1.12531</td>
<td>0.00267</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>39</td>
<td>0.3</td>
<td>1.12531</td>
<td>1.80026</td>
<td>0.00386</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>39</td>
<td>0.9</td>
<td>1.70105</td>
<td>3.57716</td>
<td>5.88467</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>1</td>
<td>0</td>
<td>1.06191</td>
<td>1.05162</td>
<td>0.02599</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>1</td>
<td>0.3</td>
<td>1.04883</td>
<td>1.04896</td>
<td>0.13610</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>1</td>
<td>0.9</td>
<td>1.01643</td>
<td>1.00291</td>
<td>0.96788</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>99</td>
<td>0</td>
<td>1.40641</td>
<td>1.40518</td>
<td>1.40518</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>99</td>
<td>0.3</td>
<td>3.15561</td>
<td>10.08956</td>
<td>0.14074</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>99</td>
<td>0.9</td>
<td>3.71833</td>
<td>9.86930</td>
<td>17.35998</td>
</tr>
</tbody>
</table>
these plots, we can see immediately that OLSAVS has the lower average TRMSE and variation. If the sample size is increased, the same comparison results hold, while making the variation for OLSAVS even smaller.

Table 5 introduces simulations using the Relaxed Lasso method. In this method comparison, we will keep the same variables and simulation values while only changing the method in which we’re applying. For the Relaxed Lasso simulations, we also will look at the same error types as the Lasso regression.

For table 5, we compare our method with a Relaxed Lasso regression with a normal distribution error type. Again, our mean will remain zero and we compare the simulations results for TRMSE and the average difference. Much like our results for Lasso regression, we see that Relaxed Lasso appears to have the advantage when \( k = 1 \). However, once \( k \) is increased to \( p - 1 \) predictors, OLSAVS begins to have the smaller TRMSE. Still, Relaxed Lasso is more competitive than Lasso for a normal distribution error.

The differences for table 5 do not have much of a trend occurring. However, our method does perform better than Relaxed Lasso for the majority of the simulations. Yet again, Relaxed Lasso does compete well with the error type coming from a normal distribution. Our method does get beat in various errors in the table, but no trends seem to appear in these spots.

Figure 13 shows simulation results between OLSAVS and Relaxed Lasso with error type 2. Figure 13 follows closely to the results from figure 12, but here both methods in figure 13 seem to contain larger variation for error type 2. However, OLSAVS still maintains the lower quantities for the TRMSE.

By looking at the results in table 6, we are going to get almost the exact same trend for TRMSE values as our values in table 5. Again, we see that Relaxed Lasso has the advantage in the TRMSE when \( k = 1 \). However, once \( k \) is increased to \( p - 1 \) predictors, OLSAVS has the smaller test error value.

Lastly, our differences in table 6 show our method performing well once again for the bias values. We can see that our method does compete with relaxed Lasso with error type 2, but judging on the majority of simulations, our method wins outright. Additionally, we can see OLSAVS also provides more consistent differences overall.
<table>
<thead>
<tr>
<th>$n$</th>
<th>$p$</th>
<th>$k$</th>
<th>$\psi$</th>
<th>TRMSE OLSAVS</th>
<th>TRMSE Lasso</th>
<th>OLSAVS Diff</th>
<th>R Lasso Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>1.36487</td>
<td>1.34437</td>
<td>0.02908</td>
<td>0.08346</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>1</td>
<td>0.3</td>
<td>1.34847</td>
<td>1.34209</td>
<td>1.34209</td>
<td>0.15530</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>1</td>
<td>0.9</td>
<td>1.30332</td>
<td>1.30211</td>
<td>0.62132</td>
<td>0.74412</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>19</td>
<td>0</td>
<td>1.44316</td>
<td>1.44326</td>
<td>0.00553</td>
<td>0.00605</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>19</td>
<td>0.3</td>
<td>1.44316</td>
<td>1.44720</td>
<td>0.00764</td>
<td>0.03972</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>19</td>
<td>0.9</td>
<td>1.43871</td>
<td>1.65548</td>
<td>1.83681</td>
<td>7.26831</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>1.44446</td>
<td>1.40636</td>
<td>0.03604</td>
<td>0.10807</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>1</td>
<td>0.3</td>
<td>1.41096</td>
<td>1.40477</td>
<td>0.20648</td>
<td>0.21225</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>1</td>
<td>0.9</td>
<td>1.34176</td>
<td>1.33864</td>
<td>0.95980</td>
<td>0.71164</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>49</td>
<td>0</td>
<td>1.85165</td>
<td>1.85164</td>
<td>0.00554</td>
<td>0.00698</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>49</td>
<td>0.3</td>
<td>2.00740</td>
<td>4.14320</td>
<td>4.14320</td>
<td>4.14320</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>49</td>
<td>0.9</td>
<td>2.31574</td>
<td>4.91246</td>
<td>8.71224</td>
<td>45.59930</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>1</td>
<td>0</td>
<td>1.33118</td>
<td>1.31585</td>
<td>0.03086</td>
<td>0.05388</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>1</td>
<td>0.3</td>
<td>1.31769</td>
<td>1.31472</td>
<td>0.13591</td>
<td>0.13865</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>1</td>
<td>0.9</td>
<td>1.29278</td>
<td>1.28860</td>
<td>0.84014</td>
<td>0.58889</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>39</td>
<td>0</td>
<td>1.44431</td>
<td>1.44436</td>
<td>0.00272</td>
<td>0.00272</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>39</td>
<td>0.3</td>
<td>1.44431</td>
<td>2.02409</td>
<td>0.00388</td>
<td>0.13289</td>
</tr>
<tr>
<td>200</td>
<td>40</td>
<td>39</td>
<td>0.9</td>
<td>1.91426</td>
<td>3.72937</td>
<td>5.77462</td>
<td>33.17816</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>1</td>
<td>0</td>
<td>1.35272</td>
<td>1.32707</td>
<td>0.04958</td>
<td>0.07102</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>1</td>
<td>0.3</td>
<td>1.33008</td>
<td>1.32910</td>
<td>0.18092</td>
<td>0.18116</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>1</td>
<td>0.9</td>
<td>1.28972</td>
<td>1.27810</td>
<td>1.19952</td>
<td>0.21588</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>99</td>
<td>0</td>
<td>1.80251</td>
<td>1.80167</td>
<td>0.00513</td>
<td>0.00464</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>99</td>
<td>0.3</td>
<td>3.31817</td>
<td>10.32727</td>
<td>0.14145</td>
<td>0.92488</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>99</td>
<td>0.9</td>
<td>3.76114</td>
<td>9.86319</td>
<td>17.36192</td>
<td>97.95763</td>
</tr>
</tbody>
</table>

Table 6: TRMSE and difference values for Relaxed Lasso for $e_i \sim EXP(1) - 1$
For this method, we will consider a Wisconsin nursing home data set provided by the Wisconsin Department of Health and Family Services. The goal of this data set is to utilize nursing home capacity. In the set we will look at the years of 2000 and 2001, with 362 and 355 facilities respectively. However, 10 observations were removed for containing missing values. Our data set contains 12 variables, with total patient years (TPY) being our response variable.

To determine how our methods perform, we will split our data into testing and training sets. In the sets, we will allocate 60% of the data to be in the training set and 40% to be in the testing set. We will compute the TRMSE from this data set and compare the values of the OLSAVS method to those of Lasso, Relaxed Lasso, and Elastic Net.

Below in table 7, we can see the results of using this real-world example. We can compare each of these side-by-side results and see that the OLSAVS method edges out each of these for this real-world example.

By analyzing table 7, it appears that OLSAVS does a nice job against the other common methods. If we look at each individual comparison, it turns out that OLSAVS has the lower test error for any method.
Table 7: TRMSE comparison for OLSAVS vs. common methods using real data

<table>
<thead>
<tr>
<th>Method</th>
<th>OLSAVS</th>
<th>ENET</th>
<th>OLSAVS</th>
<th>Lasso</th>
<th>OLSAVS</th>
<th>Relaxed Lasso</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRMSE</td>
<td>7.956877</td>
<td>8.071371</td>
<td>8.007678</td>
<td>8.044224</td>
<td>8.007678</td>
<td>8.013652</td>
</tr>
</tbody>
</table>

we choose. This showing that OLSAVS does have some practicality as it is able to perform in real-world scenarios.

5 R Package

In this section, we will briefly cover the R functions required to compute simulation results and plots shown in section 3. With each of these functions, our goal was to compare the OLSAVS with a few most common variable selection methods. Below, we will describe briefly what each of these functions do. To load this package: use `source("https://hasthika.github.io/olsvspack.txt")`

5.1 Function descriptions

`OLSAVS(x, y, method)`

Where `x` is a matrix of predictors, `y` is the response variable we are searching for, and the method determines what method we are applying OLSAVS with. Here the methods vary between Lasso, Ridge, Relaxed Lasso, and Elastic Net models. After a preferred method is selected, we will apply ordinary least squares to this model and gather a new model which will be the OLSAVS model.

`OLSAVSPredict(model, newx)`

This function will give us a vector of predictions ($\hat{y}$) for an OLSAVS model. ‘OLSAVSPredict’ will use the OLSAVS model and an argument `newx`, where `newx` is a matrix `x` values.

`olsafterlrsim(n, p, k, nruns, psi, type, alpha, pitype)`

This function was used to create Tables 1 through 6. This function requires the following arguments, `n` representing our sample size for the simulation, `p`, the number of predictors in the model, `k` as the number of non-trivial predictors in the model, `nruns` represents the number of simulations we run the model on, `ψ` as the correlation between non-trivial predictors, `type` defines the error distribution our model and rival model will use, and lastly, `pitype` represents which model we will compare our method
to. This could fall under Lasso, Ridge, Relaxed Lasso, or Elastic Net. This function results in four different outputs. Function outputs the TRMSE value for OLSAVS and the method selected, but also the difference between the estimated and the actual regression coefficients computed by using the Minkowski distance.

\[ \text{olsafterlrsimplots}(n, p, k, \text{nruns}, \text{psi}, \text{type}, \text{alpha}, \text{pitype}) \]

This function is very similar to olsafterlrsim(), however, this code now includes an output that produces plots. These plots produced show a side-by-side box plot of the TRMSE values for both methods defined in the code. The same variables are used here as in olsafterlrsim().

# Example

```r
library(ISLR)
library(lars)
library(readr)

Wisc <- read_csv("WiscNursingHome (1).csv")
Wisc <- na.omit(Wisc)
set.seed(1008)

## Splitting and preparing data
train = sample(c(TRUE, FALSE), size = 0.6*nrow(Wisc), replace = TRUE)
test <- (!train)

xtrain <- model.matrix(TPY ~., Wisc)[train,]
xtest <- model.matrix(TPY ~., Wisc)[test,]

ytrain <- Wisc$TPY[train]
ytest <- Wisc$TPY[test]
```

32
y <- ytrain
x <- xtrain

#################################################################### Applying OLSAVS ####################################################################
# Method - Lasso, RR, Enet or Relaxed Lasso (method = 1, method = 0, # method = 0.5 or method = 2)
mod <- OLSAVS(x, y, method = 0.5) # Fitting the OLSAVS model
summary(mod)
# Predictions
# OLSAVSPredict: function for predictions for OLSAVS models
preds <- OLSAVSPredict(mod, xtest)
# Test root mean square error for OLSAVS
yhatTest <- preds$yhat
RMSE_test_OLSAVS <- sqrt(mean((ytest - yhatTest)^2))
RMSE_test_OLSAVS

#################################################################### Lasso, Ridge, relax, enet ####################################################################
# Test root mean square error Lasso, Ridge, enet or relax (alpha = 1, # 0 or 0.5 respectively. relax Lasso: see below)
# for relax Lasso: out <- cv.glmnet(x, y, relax = TRUE)
out <- cv.glmnet(x, y, relax = TRUE)
lam <- out$lambda.min
out_coef <- predict(out, s = lam, type = "coefficients")
6 Conclusions

The new method for variable selection, OLSAVS involves applying ordinary least squares to a subset of predictors selected from a specific variable selection method such as Lasso, relax Lasso, or Elastic Net. The proposed method is able to maintain the ability of consistency through the low variation applied from shrinkage methods, while also reducing bias by applying OLS to the predictors selected. We expected the OLSAVS to reduce the bias in the regression coefficients introduced by the shrinkage method and lead to the model being close-fitting while keeping the consistency of the reduced variance from the shrinkage method. Simulation results show that the OLSAVS method not only reduced the bias of the regression coefficient but also further reduced the variance of the estimates.

Furthermore, the OLSAVS method performs well in terms of predictions error as well. As discussed in section 3.2, the test root mean square errors when using OLSAVS for all error types studied are either significantly low or equal to the competing shrinkage method. It is interesting to notice that the prediction accuracy drastically decreases as the correlation between the predictors increases when using commonly used shrinkage methods. Prediction accuracy decreases further with number of non-trivial predictors. OLSAVS method outperformed the other shrinkage studied in both of above mentioned scenarios and produced much lower test root mean square error values. Lastly, when applied to the real world example of the Wisconsin nursing home data set, the method shows its usefulness by resulting in the lowest observed model accuracy values. Hence, this concludes that OLSAVS does have practically in variable selection techniques and is reliable for multiple error distributions.
References


Appendices

A R Package

#########################################################################
# To load this library: use source("https://hasthika.github.io/olsvspack.txt") #
# Need library(ggplot2), library(leaps) #
#########################################################################

olsafterlrsim <- function(n = 10, p = 4, k = 2, nruns = 100, eps = 0.1,
  shift = 9, psi = 0.0, type = 1, alpha = 0.05,
  pitype = 1){

  # Needs library(glmnet). Does Lasso, RR or Relaxed Lasso.
  # slow 10 fold CV,
  # Simulates RMSE for lasso if pitype = 1, Ridge regression if
  # pitype = 0, raaxed lasso for pitype = 2.
  # 1 <= k <= p-1, k is the number of nonnoise variables
  # Uses five iid error distributions:
  # type = 1 for N(0,1) errors, 2 for t5 errors, 3 for exp(1) - 1 errors
  # 4 for uniform(-1,1) errors, 5 for (1-eps) N(0,1) + eps N(0,(1+shift)^2)
  # errors.
  # constant = 1 so there are p = q+1 coefficients
  # need p > 1, beta = (1, 1, ..., 1, 0, ..., 0) with k+1 ones, p k-1 zeroes
  # Multiply x by A: for MVN data this results

  #
# in a covariance matrix with eigenvector \(c(1, \ldots, 1)^T\)

# corresponding to the largest eigenvalue. As \(\psi\) gets

# close to 1, the data clusters about the line in the

# direction of \((1, \ldots, 1)^T\). See Maronna and Zamar (2002).

# \(\text{cor}(X_i,X_j) = \frac{2 \psi + (q-2)\psi^2}{1 + (q-1)\psi^2}\), \(i \neq j\)

# when the correlation exists.

# set.seed(974) # need \(p > 2\)

```r

e_lr <- rep(NA, nruns)
e_ols <- rep(NA, nruns)
RMSE_lr <- 0
RMSE_ols <- 0
ols_coef2 <- rep(0, p)
ols_coef_sum <- rep(0, p)
lr_coef_sum <- rep(0, p)
ols_coef_avg <- rep(0, p)
lr_coef_avg <- rep(0, p)

q <- p-1
rho <- (2*psi + (q-2)*psi^2)/(1 + (q-1)*psi^2)
A <- matrix(psi, nrow=q, ncol=q)
diag(A) <- 1
b <- 0 * 1:q
```

37

indx <- 1:n

for(i in 1:nruns) {
  x <- matrix(rnorm(n * q), nrow = n, ncol = q)
  x <- x %*% A
  xf <- rnorm(q) %*% A
  if(type == 1) {
    y <- 1 + x %*% b + rnorm(n)
    yf <- 1 + xf %*% b + rnorm(1)
  }
  if(type == 2) {
    y <- 1 + x %*% b + rt(n, df = 5)
    yf <- 1 + xf %*% b + rt(1, df = 5)
  }
  if(type == 3) {
    y <- 1 + x %*% b + rexp(n) - 1
    yf <- 1 + xf %*% b + rexp(1) - 1
  }
  if(type == 4) {
    y <- 1 + x %*% b + runif(n, min = -1, max = 1)
    yf <- 1 + xf %*% b + runif(1, min = -1, max = 1)
  }
}
if (type == 5) {
  err <- rnorm(n, sd = 1 + rbinom(n, 1, eps) * shift)
  y <- 1 + x %*% b + err
  ef <- rnorm(1, sd = 1 + rbinom(1, 1, eps) * shift)
  yf <- 1 + xf %*% b + ef
}

# make an MLR data set

# find the 10 fold CV lasso or Ridge regression model when
# pitype == 1 || pitype == 0
if (pitype == 1 || pitype == 0) {
  out <- cv.glmnet(x, y, alpha = pitype)
  lam <- out$lambda.min
  out_coef <- predict(out, s = lam, type = "coefficients")
  yfhat <- predict(out, s = lam, newx = xf)
}

# find the 10 fold CV Relaxed lasso model when pitype == 2
if (pitype == 2) {
  rel_fit <- cv.glmnet(x, y, relax = TRUE)
  lam <- rel_fit$lambda.min
  out_coef <- predict(rel_fit, s = lam, type = "coefficients")
  yfhat <- predict(rel_fit, s = lam, newx = xf)
# Filter out the non-zero coefficients
non_zero_indx <- which(out_coef[-1] != 0)
x_ols <- x[, non_zero_indx]

ols_mod <- lm(y ~ x_ols)  # Apply OLS on the selected predictors
ols_coef <- ols_mod$coefficients

# For comparisons of coefficients
non_zero_coef_indx <- which(out_coef != 0)
ols_coef2[non_zero_coef_indx] <- ols_coef

xf_ols <- xf[, non_zero_indx]
xf_ols <- append(xf_ols, 1, after = 0)  # modified xf so it will work with ols
yfhat_ols <- xf_ols %*% ols_coef

e_lr[i] <- (yf - yfhat)^2
e_ols[i] <- (yf - yfhat_ols)^2

# Compare coefficients
ols_coef_sum <- ols_coef2 + ols_coef_sum
lr_coef_sum <- out_coef + lr_coef_sum

# TRMSEs
RMSE_lr <- sqrt(sum(e_lr)/nruns)
RMSE_ols <- sqrt(sum(e_ols)/nruns)

# Coef avgs
ols_coef_avg <- ols_coef_sum/nruns
lr_coef_avg <- lr_coef_sum/nruns

# Minkowski distance
diff_ols <- (sum( (abs(append(b, 1, after = 0)
ols_coef_avg))^p ))^(1/p) # Sum of the absolute differences
between b and new ols coefficients estimates
diff_lr <- (sum( (abs(append(b, 1, after = 0) - lr_coef_avg))^p
))^(1/p) # Sum of the absolute differences between b and l or r
coefficients estimates

list(RMSE_ols = RMSE_ols, RMSE_lr = RMSE_lr, diff_ols=diff_ols,
diff_lr=diff_lr)
olsafterlrsimplots <- function(n = 50, p = 4, k = 2, nruns = 100, eps = 0.1,
shift = 9, psi = 0.0, type = 1, alpha = 0.05,
pitype = 1){
  # Needs library(glmnet). Does lasso, RR or Relaxed lasso
  simulates and produces a sid-by-side boxplot.
  ##slow 10 fold CV,
  # Simulates RMSE for lasso if pitype = 1, Ridge regression if
  pitype = 0, ralxed lasso for pitype = 2.
  # 1 <= k <= p-1, k is the number of nonnoise variables
  # Uses five iid error distributions:
  # type = 1 for N(0,1) errors, 2 for t3 errors, 3 for exp(1) - 1 errors
  # 4 for uniform(-1,1) errors, 5 for (1-eps) N(0,1) + eps N(0,(1+shift)^2)
  # errors.
  # constant = 1 so there are p = q+1 coefficients
  # need p > 1, beta = (1, 1, ..., 1, 0, ..., 0) with k+1 ones, p-k-1 zeroes
  # Multiply x by A: for MVN data this results
  # in a covariance matrix with eigenvector c(1, ..., 1)^T
  # corresponding to the largest eigenvalue. As psi gets
  # close to 1, the data clusters about the line in the
  # direction of (1, ..., 1)^T. See Maronna and Zamar (2002).
\[
\text{cor}(X_i,X_j) = \frac{2\psi + (q-2)\psi^2}{1 + (q-1)\psi^2}, \text{ i not = j}
\]

# when the correlation exists.

# set.seed(974) ##need p>2

e_lr <- rep(NA, nruns)
e_ols <- rep(NA, nruns)
RMSE_lr <- 0
RMSE_ols <- 0
ols_coef2 <- rep(0, p)
ols_coef_sum <- rep(0, p)
lr_coef_sum <- rep(0, p)
ols_coef_avg <- rep(0, p)
lr_coef_avg <- rep(0, p)

q <- p-1
rho <- (2*psi + (q-2)*psi^2)/(1 + (q-1)*psi^2)
A <- matrix(psi, nrow=q, ncol=q)
diag(A) <- 1
b <- 0 * 1:q
indx <- 1:n
for(i in 1:nruns) {
  x <- matrix(rnorm(n * q), nrow = n, ncol = q)
  x <- x %*% A
  xf <- rnorm(q) %*% A
  if(type == 1) {
    y <- 1 + x %*% b + rnorm(n)
    yf <- 1 + xf %*% b + rnorm(1)
  }
  if(type == 2) {
    y <- 1 + x %*% b + rt(n, df = 3)
    yf <- 1 + xf %*% b + rt(1, df = 3)
  }
  if(type == 3) {
    y <- 1 + x %*% b + rexp(n) - 1
    yf <- 1 + xf %*% b + rexp(1) - 1
  }
  if(type == 4) {
    y <- 1 + x %*% b + runif(n, min = -1, max = 1)
    yf <- 1 + xf %*% b + runif(1, min = -1, max = 1)
  }
  if(type == 5) {
    err <- rnorm(n, sd = 1 + rbinom(n, 1, eps) * shift)
    y <- 1 + x %*% b + err
  }
ef <- rnorm(1, sd = 1 + rbinom(1, 1, eps) * shift)
yf <- 1 + xf %*% b + ef
}

# make an MLR data set

# find the 10 fold CV lasso or ridge regression model when pitype == 1 || pitype == 0
if(pitype == 1 || pitype == 0){
  out<-cv.glmnet(x, y, alpha = pitype)
  lam <- out$lambda.min
  out_coef <- predict(out, s = lam, type = "coefficients")
  yfhat <- predict(out, s = lam, newx = xf)
}

# find the 10 fold CV Relaxed lasso model when pitype == 2
if(pitype == 2){
  rel_fit <- cv.glmnet(x, y, relax = TRUE)
  lam <- out$lambda.min
  out_coef <- predict(rel_fit, s = lam, type = "coefficients")
  yfhat <- predict(rel_fit, s = lam, newx = xf)
}

# Filter out the non-zero coefficients
non_zero_index <- which(out_coef[-1] != 0)
x_ols <- x[, non_zero_index]

ols_mod <- lm(y ~ x_ols) # Apply OLS on the selected predictors
ols_coef <- ols_mod$coefficients

# For comparisons of coefficients
non_zero_coef_index <- which(out_coef != 0)
ols_coef2[non_zero_coef_index] <- ols_coef

xf_ols <- xf[, non_zero_index]
xf_ols <- append(xf_ols, 1, after = 0) # modified xf so it will work with ols
yfhat_ols <- xf_ols %*% ols_coef

e_lr[i] <- (yf - yfhat)^2
e_ols[i] <- (yf - yfhat_ols)^2

# Compare coefficients
ols_coef_sum <- ols_coef2 + ols_coef_sum
lr_coef_sum <- out_coef + lr_coef_sum
# Plot TRMSEs
rmse <- data.frame(TRMSE = c(e_lr, e_ols), Method = c(rep("Other", nruns), rep("OLSAVS", nruns)))

data = rmse, aes(x= Method, y = TRMSE, fill = Method)) + geom_boxplot() + theme_bw()

# TRMSEs
RMSE_lr <- sqrt(sum(e_lr)/nruns)
RMSE_ols <- sqrt(sum(e_ols)/nruns)

# Coef avgs
ols_coef_avg <- ols_coef_sum/nruns
lr_coef_avg <- lr_coef_sum/nruns

# Minkowski distance
diff_ols <- (sum( (abs(append(b, 1, after = 0) - ols_coef_avg))^p ))^(1/p) # Sum of the absolute differences between b and new ols coefficients estimates
diff_lr <- (sum( (abs(append(b, 1, after = 0) - lr_coef_avg))^p ))^(1/p) # Sum of the absolute differences between b and 1 or r
coefficients estimates

list(RMSE_ols = RMSE_ols, RMSE_lr = RMSE_lr, diff_ols=diff_ols, diff_lr=diff_lr, rmsePlot = rmsePlot)
}

OLSAVS <- function(x, y, method){
    # OLSAVS des the OLSAVS method
    # Needs library(glmnet).
    # need x as a model.matrix
    # y - matrix of predictors
    # y - response
    # Method - lasso, RR, Enet or Relaxed lasso (method = 1, method = 0, method = 0.5 or method = 2)
    # 10 fold CV

    # find the 10 fold CV lasso or ridge regression or enet model
    when method == 1 || method == 0 || method == 0.5
    if(method == 1 || method == 0 || method == 0.5){
        out<-cv.glmnet(x, y, alpha = method)
        lam <- out$lambda.min
        out_coef <- predict(out,s = lam, type = "coefficients")
    }
#yfhat <- predict(out ,s = lam, newx = xf)

if(method == 2){
  rel_fit <- cv.glmnet(x, y, relax = TRUE)
  lam <- rel_fit$lambda.min
  out_coef <- predict(rel_fit, s = lam, type = "coefficients")
  #yfhat <- predict(rel_fit, s = lam, newx = xf)
}

# Filter out the non-zero coefficients
non_zero_indx <- which(out_coef[-1] != 0)

x_ols <- x[, non_zero_indx]

ols_mod <- lm(y ~ x_ols) # Apply OLS on the selected predictors
ols_coef <- ols_mod$coefficients

outPut <- list(ols_mod = ols_mod, ols_coef = ols_coef, out_coef = out_coef)
outPut$ols_mod
OLSAVSPredict <- function(model, newx) {
    # Does predictions for OLSAVS

    xtest <- newx
    mod <- model

    names(mod$coefficients) = gsub(pattern = "x_ols", replacement = ", x = names(mod$coefficients))

    xtest1 <- xtest[, names(mod$coefficients[-1])] # good
    newxtest <- cbind(rep(1, nrow(xtest1)), xtest1) # Added a column
    of ones to make xtest a design matrix
    yhatTest <- newxtest %*% as.matrix(mod$coefficients)

    pred <- list(yhat = yhatTest)
    pred
}

Example
library(ISLR)
library(lars)
data(Auto)

#set.seed(1111)

############################ Splitting and preparing data ############################

train = sample(c(TRUE, FALSE), size = 0.7*nrow(Auto), replace = TRUE)
test <- (!train)

xtrain <- model.matrix(mpg˜., Auto)[train,]
xtest <- model.matrix(mpg˜., Auto)[test,]

ytrain <- Auto$mpg[train]
ytest <- Auto$mpg[test]

y <- ytrain
x <- xtrain

############################ Applying OLSAVS ########################################

#Method - lasso, RR, Enet or Relaxed lasso (method = 1, method =
mod <- OLSAVS(x, y, method = 0.5) # Fitting the OLSAVS model
mod
summary(mod)

# Predictions
preds <- OLSAVSPredict(mod, xtest) # OLSAVSPredict: separate function for predictions

# Test root mean square error for OLSAVS
yhatTest <- preds$yhat
RMSE_test_OLSAVS <- sqrt(mean((ytest - yhatTest)^2))
RMSE_test_OLSAVS

# lasso, ridge, relax, enet

# Test root mean square error lasso, ridge, enet or relax (alpha = 1, 0, 0.5, relax: see below)
# for relax lasso: cv.glmnet(x, y, relax = TRUE)
out<-cv.glmnet(x, y, alpha = 0.5)
lam <- out$lambda.min
out_coef <- predict(out, s = lam, type = "coefficients")
yfhat <- predict(out, s = lam, newx = xtest)
RMSE_test_l <- sqrt(mean((ytest - yfhat)^2))
RMSE_test_l

########################## End Example ##########################