4-11-2019

Best Probable Subset: A New Method for Reducing Data Dimensionality in Linear Regression

Elieser Nodarse
Florida International University, enoda006@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

BEST PROBABLE SUBSET: A NEW METHOD FOR REDUCING DATA DIMENSIONALITY IN LINEAR REGRESSION

A thesis submitted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

in

STATISTICS

by

Elieser Nodarse

2019

To: Dean Michael R. Heithaus
College of Arts, Sciences, and Education

This thesis, written by Elieser Nodarse, and entitled Best Probable Subset: A New Method for Reducing Data Dimensionality in Linear Regression, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this thesis and recommend that it be approved.

Sneh Gulati

Golam Kibria

Zhenmin Chen

Hassan Zahedi, Major Professor

Date of Defense: April 11, 2019

The thesis of Elieser Nodarse is approved.

Dean Michael R. Heithaus
College of Arts, Sciences, and Education

Andrés G. Gil
Vice President for Research and Economic Development
And Dean of the University Graduate School

Florida International University, 2019

DEDICATION

I dedicate this thesis to my family: Maria T. del Cristo, Orlando Nodarse, Orlando Nodarse Jr., Ernesto Carnota, and Shadow Nodarse. This thesis would not have been possible without your love and support. I love all of you, always.

ACKNOWLEDGMENTS

I would like to thank my Major Professor, Dr. Hassan Zahedi, for guiding me throughout my professional career during my years at FIU. I would also like to thank the rest of my committee members and other professors who helped throughout the process of my thesis, Dr. Sneh Gulati, Dr. Golam Kibria, Dr. Zhenmin Chen, Dr. Florence George, and Professor Sergio Perez. Thank you for your guidance and support. Finally, I am grateful to the Department of Mathematics and Statistics and Florida International University for giving me the opportunity to study and excel in what I love.

ABSTRACT OF THE THESIS

BEST PROBABLE SUBSET: A NEW METHOD FOR REDUCING DATA DIMENSIONALITY IN LINEAR REGRESSION

by

Elieser Nodarse

Florida International University, 2019

Miami, Florida,

Professor Hassan Zahedi, Major Professor

Regression is a statistical technique for modeling the relationship between a dependent variable Y and two or more predictor variables, also known as regressors [1]. In the broad field of regression, there exists a special case in which the relationship between the dependent variable and the regressor(s) is linear. This is known as linear regression.

The purpose of this paper is to create a useful method that effectively selects a subset of regressors when dealing with high dimensional data and/or collinearity in linear regression. As the name depicts, high dimensional data occurs when the number of predictor variables is far too large to use commonly known methods. Collinearity, on the other hand, occurs when there exists a linear relationship amongst one or more pairs of independent variables. This paper is divided into three main section: an introduction, which reviews key concepts that are needed for a full understanding of the paper; the methodology, which guides the reader, step-by-step, through the process of the newly devised method; results, which thoroughly explain and analyze any findings and propose further ideas to be studied.

TABLE OF CONTENTS

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

FIGURE PAGE

INTRODUCTION

RESEARCH PURPOSE

The principal purpose of this paper is to present a new method that deals with high dimensional data and/or collinearity in linear regression.

GENERAL INFORMATION

Regression Analysis is a statistical technique used for modeling and analyzing the relationship between a response variable Y and one or more predictor variables X_1, $X_2, \dots, X_k$ [1]. When the relationship between Y and each of the predictors $X_1,\ X_2, \dots, X_k$ is known or assumed to be linear, we use a linear approach, commonly known as Linear Regression. Some of the main purposes of linear regression analysis include data collection as well as fitting and estimating the regression parameters of a linear regression model, which can provide us with useful information about the relationship between the response variable and one or more predictor variables. Data collection is very essential, as "any regression analysis is only as good as the data on which it is based" [1].

A very popular method for estimating the regression parameters of a linear model is known as *Least Squares Estimation* (LSE) [2]. However, the LSE method cannot be used when the number of regressors equals or exceeds the number of observations in the data, a phenomenon known as *high dimensionality* [3]. Using the LSE can overfit the data when the number of predictors equals the number of observations, which means that the data will follow the error too closely. There are some existing solutions to data having high dimensionality, including Principal Component Analysis, which transforms the data, deriving a low-dimensional set of orthogonal variables [3].

In the present paper we focus on methods for reducing the dimensions of data sets without having to transform the data set. Most of the methods that we will discuss belong to the Subset Selection class, which select a smaller subset of regressors [3]. After thoroughly exploring these methods, we define and propose a new subset selection method, which we call *Best Probable Subset* (BPS) in hopes of fitting linear regression models containing only those predictor variables that contribute to the model while keeping unnecessary variables out of the model.

CHAPTER I

LITERATURE REVIEW

A Statistical Model is a quantity in which we relate a set of input variables to an output variable, along with some variability [4] (Figure 1). We begin this paper by reviewing a specific type of statistical models – Linear Regression model – and reviewing some of the most widely used techniques to estimate and analyze these models.

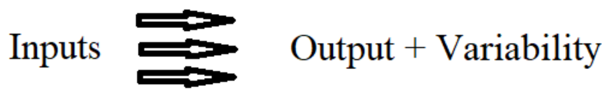


Figure 1: *Schematic for Statistical Models. (Reproduced and updated from Methods and Applications of Linear Models, Hocking, 2nd ed.)*

Regression

Regression is a statistical technique for modeling the relationship between a response – sometimes called dependent– variable Y and one or more predictor variables, or regressors. *Regression Analysis* deals with finding and modeling the best relationship between Y and the regressor(s), quantifying the strength of said relationship, and predicting future response values for a given set of values of the regressors [6]. Amongst all regression models, a special case exists in which the relationship between the dependent variable Y and the regressors is linear; this is known as *Linear Regression* [1]. Linear regression can be further divided into two commonly known cases. The first case involves a response variable Y and only one predictor variable X. This is known as *Simple Linear Regression* [1] and it has the general model:

$$Y = \beta_0 + \beta X + \varepsilon \qquad (1)$$

where β_0 and β are unknown parameters known as *regression coefficients*. Specifically, β_0 is known as the Y-Intercept –the value that Y assumes when X is zero, if it can be zero–

and β is the slope, which determines how Y changes with every unit change in X [1]. One of the purposes of linear regression analysis is to estimate those parameters. Lastly, ε (epsilon), distributed as $(0, \sigma^2)$, is a random variable that accounts for the error, or difference, between the linear function of X –$\beta_0 + \beta X$– and the observed values of the variable Y[1]. Also, it is assumed that σ^2 is unknown [5]. Here we can see that linear regression models can be viewed as having a predictable part, $\beta_0 + \beta X$, and an unpredictable part, ε [7]. There are some assumptions that must be met for us to perform any linear regression analysis on the regression model. These assumptions are formally listed here for the convenience of the reader:

1. The relationship between the predictor variable(s) and the response variable must be linear [7].
2. The error term ε is distributed as $N(0, \sigma^2)$ [7]
3. The errors of the response variable are independent of each other [7]
4. Predictor variables are independent of each other (i.e. no collinearity) [7]

For each value of x, there is a population of possible values, $Y|x$, for the response variable, which follows the model:

$$Y|x = \beta_0 + \beta x + \varepsilon \qquad (2)$$

We use the notation $Y|x$ to indicate the population of values for the variable Y given a value of x, and y to indicate an observed value of $Y|x$. Figure 2 shows a visual representation of the model $Y|x = \beta_0 + \beta x + \varepsilon$. Note that there is a distribution for the population of possible values in $Y|x$, when X assumes a value x. The $Y|x$ population has a mean equal to the linear model at a given value of x and a constant variance σ^2. This occurs because the value x is fixed and $\varepsilon \sim N(0, \sigma^2)$, so the mean of $Y|x$ is a linear function of the x [2], and the variance equals the variance of ε:

$$E(Y|x) = \beta_0 + \beta x \; and \; V(Y|x) = \sigma^2$$

The other case of linear regression involves a response variable and two or more predictor variables, which is known as *Multiple Linear Regression* [1] and the multiple linear regression model follows the format:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \varepsilon \qquad (3)$$

In (3), Y is the response variable while $X_1, X_2, \ldots, X_k$ are the predictor variables. Parameter β_0 is known as the Y-intercept –the value of Y when all regressors are assumed to have a value of zero, if it makes sense for them to be zero– and parameters $\beta_1, \beta_2, \ldots, \beta_k$ are the regression coefficients, which determine how regressors $X_1, X_2, \ldots, X_k$, respectively, influence Y. Like simple linear regression, ε is the random variable accounting for the difference between the model and the observed values of the variable Y, and it is distributed as $N(0, \sigma^2)$ [1]. In equation (3), the response variable depends on variables $X_1, X_2, \ldots, X_k$. Similar to simple linear regression, for a set

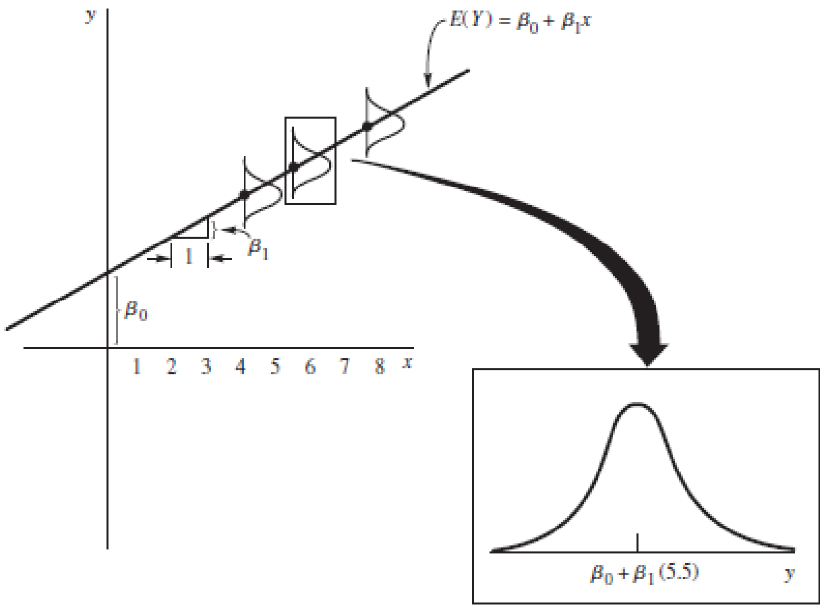


Figure 2: *A visual representation of a Linear Model. (Figure taken from Mathematical Statistics with Applications, Wackerly, 7th ed.)*

of fixed values $x_1,\ x_2, \dots, x_k$, the response variable $Y|x_1, x_2, \dots, x_k$ has a population of possible values. Essentially, the response variable $Y|x_1, x_2, \dots, x_k$ follows the model:

$$Y|x_1, x_2, \dots, x_k\ =\ \beta_0 +\ \beta_1 x_1\ +\ \beta_2 x_2 + \cdots + \beta_k x_k +\ \varepsilon \qquad (4)$$

Since the set of values $x_1,\ x_2, \dots, x_k$ are fixed and $\varepsilon \sim N(0, \sigma^2)$, the mean of $Y|x_1, x_2, \dots, x_k$ is a linear function of the regressors, and the variance equals the variance of ε [1]. In other words:

$$E(Y|x_1, x_2, \dots, x_k) =\ \beta_0 +\ \beta_1 x_1\ +\ \beta_2 x_2 + \cdots + \beta_k x_k\ and\ V(Y|x_1, x_2, \dots, x_k)\ =\ \sigma^2$$

In order to model the relationship between the response variable Y and the predictor variable(s), we must first estimate the regression parameters $\beta_0, \beta_1, \dots, \beta_k$, which brings us to the next section.

Least Squares Estimation

The Method of Least Squares (LS) is used to estimate the regression coefficients in both simple and multiple linear regression. For simplicity purposes, let us consider this method in simple linear regression first and then in multiple linear regression.

Suppose we have n pairs of data $(x_1, y_1),\ (x_2, y_2), \dots, (x_n, y_n)$. Note that here we only have one predictor variable and one response variable. The subscripts from 1 to n simply define the different data sets. The Method of Least Squares estimates the parameters $\beta_0\ and\ \beta_1$ and generates a fitted linear regression model $\hat{Y} =\ b_0 +\ b_1 x$, where $b_0\ and\ \ b_1$ are unbiased Least Squares Estimators (LSE) of $\beta_0\ and\ \beta_1$, respectively, making $\hat{Y}$ an unbiased estimator of the model, so $E(Y|x) =\ \beta_0 +\ \beta x$ [2]. For each given value x_1, $x_2, \dots, x_n$, the fitted model generates a corresponding fitted value $\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n$ [1]. Without diving into the formulas and computations used by the method of least squares, we only need to know that said method provides us with estimates that minimize the sum of squares of the difference between the observations $y_1,\ y_2, \dots, y_n$ and the fitted values $\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n$ [1]. Let us note here that the difference between an observed value y_i and a fitted value $\hat{y}_i$, both corresponding to the same x_1, is known as a *residual* e_i [3].

The LS method follows the same idea for multiple linear regression: It provides us with estimates for $\beta_1, \beta_2, \dots, \beta_k$ which minimize the sum of squares of the residuals $e_i = y_i - \hat{y}_i, for\ 1 \leq i \leq n$ [1]. For the purpose of the current paper, we will review the estimates of the regression parameters in matrix form. Suppose that we have n observations on a response variable $Y|x$ and k predictor variables:

$$(x_{11}, x_{12}, \dots, x_{1k}, y_1), (x_{21}, x_{22}, \dots, x_{2k}, y_2), \dots, (x_{n1}, x_{n2}, \dots, x_{nk}, y_n)$$

We define the following matrices:

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \boldsymbol{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1k} \\ 1 & x_{21} & x_{22} & \dots & x_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{nk} \end{bmatrix}, \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}, \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

The solution for finding the estimates of the correlation coefficients is given by:

$$\boldsymbol{b} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \qquad (5)$$

The proof and specifics for obtaining the least squares estimates $\boldsymbol{b}$ are covered in Douglas C. Montgomery's *Introduction to Linear Regression Analysis* or any Regression book cited in this paper.

Violation of the LS Assumptions

The Least Squares estimates $\boldsymbol{b}$ will have low bias provided that the true relationship between the predictor variable(s) and the response variable Y is linear. Furthermore, if the number of observations n is much larger than the number of regressors k, the least squares estimates tend to have a low variance [3]. In fact, if this and the rest of the assumptions presented earlier in the previous section are met, the Least Squares estimator is known to have the minimum variance amongst all unbiased estimators. Such estimators are known as the *Minimum Variance Unbiased Estimators* (MVUE) [2]. However, these assumptions are not always met and can bring along many problems. Here are some of the violated assumptions along with their respective outcomes:

Non-linearity of the Data

When the data are not from linear, the conclusions that we draw from the fitted model are suspect for inaccuracy. A good way to check if this assumption is met is to plot a Residuals vs. Fitted values plot. If a pattern is visible, the data might not be linear. Figure 3 shows an example of a non-linear dataset. When this assumption is violated, a simple transformation –such as $\sqrt{X}, \log(X), or\ X^2$– could be appropriate for fitting a model [3].

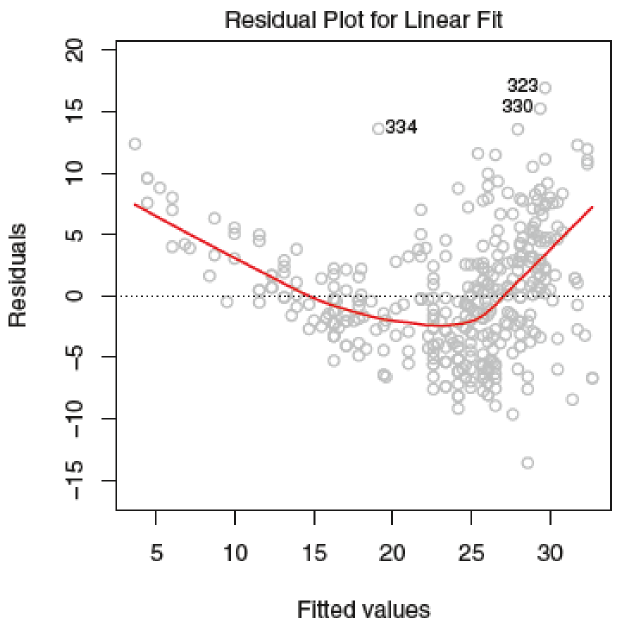


Figure 3: *Linearity assumption not met. (Image taken from Gareth James' An Introduction to Statistical Learning with Applications in R.)*

Correlation of Error Terms

One of the assumptions to linear regression analysis is that the error terms of different observations are uncorrelated to each other. If this assumption is not met, the estimated standard errors of the estimated regression coefficients will underestimate the true standard errors, which will affect any confidence and prediction intervals that are constructed [3].

Non-constant Variance of Error Terms

The assumption that we have a constant variance for all error terms is important when making confidence intervals and hypothesis tests. If the assumption of equal variance is violated, such intervals and tests might be inaccurate. We can check this assumption by plotting the Residuals vs. Fitted values and keeping an eye for any changes in the sparsity of the Residuals vs Fitted Values plot. Figure 4 shows an example of a dataset in which the variance σ^2 is not constant. The non-constant variance problem can usually be solved by applying a transformation to the response variable Y, such as $\log(Y) or \sqrt{Y}$ [3].

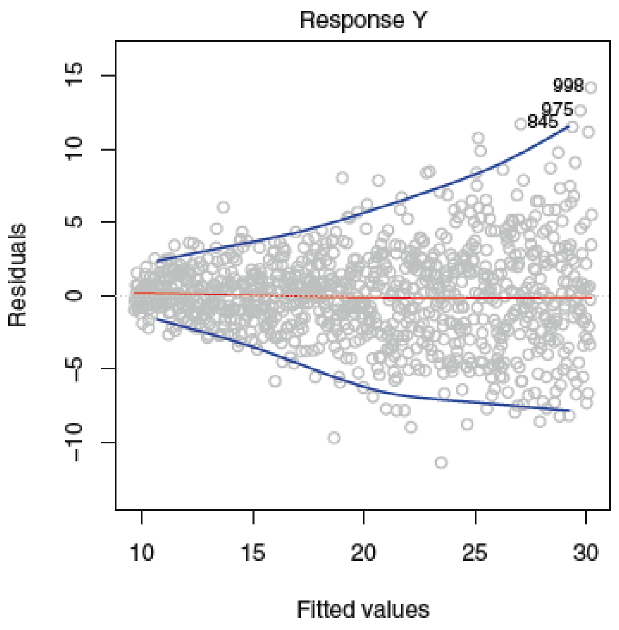


Figure 4: *Constant-variance assumption not met. (Image taken from Gareth James' An Introduction to Statistical Learning with Applications in R.)*

Collinearity

A measure of "closeness" or dependency between two variables, whether dependent or independent, is known as the *correlation* between those variables. *Collinearity* occurs when two predictor variables are closely related to each other [3]. The phenomenon of collinearity reduces the accuracy of the estimates of the regression coefficients and increases their standard error. Consequently, the t-statistics used for testing each individual coefficient is also reduced and the power of the hypothesis tests declines. Collinearity can be detrimental when fitting a model. Some ways to detect it include looking at the correlation matrix of all regressors or computing the *variance inflation factor* (VIF) [3]. When looking at the VIFs, we can drop one of the regressors that yield the highest VIF and check if the collinearity has been reduced or eliminated [4]. The details of VIFs are beyond the scope of this paper, but more information can certainly be found in Hocking's *Methods and Applications of Linear Models*, 3rd Ed.

Quality of Fit of the Model

We have explored the Method of Least Squares when estimating the regression coefficients of a linear model. However, we cannot fit a regression model and hope that it will be a good estimate of the regression parameters. We need a way to measure how well the fitted values do at predicting the observed data. We present some statistics used for measuring the quality of fit of a fitted mode.

Mean Squared Error

The most commonly used measure of fit is known as *Mean Squared Error (MSE)* [3]. Like the name indicates, the MSE measures the average of the square of the errors between the observed data and the predicted values of that same data. The formula for the MSE is provided here:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad (6)$$

where y_i is the ith observation of the data containing n observations and $\hat{y}_i$ is the corresponding ith predicted value. Logically, we expect the value of MSE to be large if the predicted values are far off the observed ones. The quantity $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, without the term $\frac{1}{n}$ being multiplied by it, is known as the *Sum of Squares of the Errors* (SSE). The SSE quantity is a measure of the variability in y after the predictor variable(s) have been considered [1].

Coefficient of Determination: R-Squared

The quantity *R-Squares* (R^2) is known as the coefficient of determination and it is a measure of the variability of the response variable y that has been explained by the predictor variable(s) [1]. When we have a simple regression model – only one predictor variable – the R^2 is known as the squared of the correlation between the predictor variable and the response variable [3]. The measure of variability can be obtained using the following equation:

$$R^2 = \frac{SSR}{SST} = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \qquad (7)$$

where SSR is known as the *Sum of Squares of Regression* (SSR) and SST is the *Total Sum of Squares* (TSS). While SSE is the measure of variability in y after the regressors have been considered, RSS is the variability of y that is explained by the regressors. These two measurements make up the total variability in y, known as TSS. Naturally, these three measures make up the equation:

$$TSS = SSR + SSE \qquad (8)$$

While MSE and R-Squared are both measures of variability in y, they offer different information. A low value in the MSE would be an indication of a good fitted model while a large value would indicate a bad fit. On the other hand, a small value of R^2 indicated a model with poor fit while a large value indicated a good fit [3]. This is because MSE measures the variability in the model not explained by the regressors while R^2 explains the variability explained by the regressors. Naturally, if we add one more predictor variable to a model, the MSE will decrease while the R^2 will decrease, whether slightly or considerably. As a result of the added predictor variable, it is best to use these two measurements of variability when comparing fitted models with the same number of regressors.

So which one should we use? This question cannot be answered directly, but we recommend using the R^2. While the MSE can be as large as we can imagine, R^2 is a fraction that can only take on values between 0 and 1, where a value close to 1 indicates that the fitted model explains a large portion of the variability in the response variable [1]. This allows us to understand how good a fitted model is in general.

Adjusted R-Squared

As mentioned above, the value of R^2 of an already existing fitted model can only increase with the addition of another predictor variable while the MSE can only decrease. However, having a larger R^2 when the model has an extra regressor does not mean that one model is better with such inclusion. In fact, it could be better with or without it. So how do we know if the inclusion of another regressor improves the model? This question can be answered using other measurements. Particularly, we focus on a statistic known as the *Adjust R-*

Squared ($R^2{}_{adj}$) which is a modified version of R^2 that pays a price for the addition of unnecessary regressors in the model [3]. The formula for finding the adjusted R-Squared is provided here:

$$R^2{}_{adj} = 1 - \frac{\frac{SSE}{(n-k-1)}}{\frac{TSS}{(n-1)}} = 1 - \frac{(1-R^2)(n-1)}{(n-k-1)} \qquad (9)$$

where k is the number of regressors in the model. Like the R^2 statistic, $R^2{}_{adj}$ can only take on values less than 1 [3]. However, unlike the non-adjusted version, $R^2{}_{adj}$ can decrease beyond zero with the addition of unnecessary predictor variables. The adjustment provides us with a tool to compare fitted models involving different numbers of regressors. While there are many more statistics used for comparing fitted models, they are beyond the scope of the present paper and we only focus on the ones mentioned above.

High-Dimensional Data

We mentioned earlier that the Least Squares Estimate is known to be the MVUE if all the assumptions are met. However, we may not be able to always find the LSE for all data sets, as it requires the number of observations n to be much larger than the number of predictor variables k. A problem occurs when n is not much larger than k. If n is only slightly larger than k, there can be too much variability in the least squares estimates, leading to possibly overfitting the data and poorly predicting future observations [3]. What is worse, if k is larger than n, the variances of the estimates are infinite, and the least squares method can no longer be used [3]. Data that fall under any one of these two scenarios are known to have *high dimensionality*.

Multiple approaches exist when dealing with high dimensional data. Subset selection and Shrinkage are two common classes of methods used when dealing with high dimensionality [3]. Subset selection involves identifying a subset of the k predictor variables that are related to the response Y. Once the subset is identified, we fit a *reduced* model using the Method of Least Squares [8]. Shrinkage, on the other hand, involves using all k original predictors. The k estimated regression coefficients are "shrunken" towards zero, relative to the LSE. Some methods of shrinkage include *Ridge Regression* and the *Lasso* method [3].

While we focus on subset selection as a primary class of methods for dimension reduction in the present paper, more information on the shrinkage class can be found in Montgomery's *Introduction to Linear Regression Analysis*, 5th Ed. and Rawlings' *Applied Regression Analysis: A Research Tool,* 2nd Ed. Another class approach used when dealing with high dimensional data involves *Tree-Based methods*. Tree-based methods are not as common as the other two mentioned above. However, we consider one method called *Random Forest*, which takes on a very different approach by selecting a subset of regressors for each split, or *node*, that occurs in any of the *trees*. We go more in depth into the subset selection class and the random forest method in the next few sections.

Subset Selection

Inside the Subset Selection class there are some useful methods that are commonly used when reducing the dimensions of a dataset. Here we present a summary and the algorithm for some of these methods.

Best Subset Selection

Best subset selection can guarantee that we will find the best possible combination of the k predictor variables [3]. The disadvantage of the best subset selection method is that we would have to fit a LS regression model for each of the 2^k possible combinations of the k predictors, whether the model contains only one predictor or the original k [3]. Best subset selection method can be very useful when the number of regressors is very small, so that we only have a small number of possible fitted models. However, it can be computationally inefficient. When the number of possible regressors is 10, we already have $10^2 = 100$ fitted models to consider; when there are 32 possible regressors, there are over 1000 possible fitted models to consider. The algorithm for this method can be found in from Gareth James' *An Introduction to Statistical Learning with Applications in R*, 7th Printing, but we summarize and present it here for your convenience:

1. Let M_0 represent the null model with no predictors and $M_1, M_2, \dots, M_k$ represent models with $1, 2, \dots, and\ k$ predictor variables.
2. For $l = 1, 2, \dots, k$, fit all possible $\binom{k}{l}$ models that contain exactly l regressors. Pick the best model among each group of $\binom{k}{l}$ and call it M_l.

   Note: The "best" model amongst all models with the same number of predictors l is the one having the largest R^2.

3. Select the single best model among the $k + 1$ models $M_0, M_1, M_2, \dots, M_k$ using any of the $C_p\ (AIC), BIC, or\ adjusted\ R^2$ statistics.[3]

   Note: The $adjusted\ R^2$ statistic is explained in a previous section.

Forward Stepwise Selection

Forward Stepwise Selection is a computationally efficient method compared to subset selection as it does not involve checking every possible fitted model. Rather, the method only considers a maximum of $\frac{k(k+1)}{2}+1$ possible fitted models, where k is the number of regressors in the full model. A major disadvantage of Forward stepwise selection is that it does not consider the possibility that a predictor variable that has already been selected for the fitted model might no longer be significant with the addition of another predictor, so it might not yield the single best possible fitted model.

The method starts with the null model M_0 and considers the regressor that best contributes to the model, if there is any that contributes to it in a significant way. A "significant" addition is one that yields a partial F statistic greater than a pre-selected value F_{IN} for "taking in" a new regressor [7]. A review on the F distribution can be found in Lyman Ott's *An Introduction to Statistical Methods and Data Analysis*, 7th Ed. or any other referenced book that covers elementary Statistics. However, the textbook *Introduction to Statistical Learning with Applications in R*, used as a reference in this paper, presents an algorithm that avoids the calculation of the partial F-statistic. The Forward selection algorithm is summarized here:

1. Begin by fitting the null model, M_0.
2. For $l = 0, 1, \dots, k-1$, consider every $k-l$ fitted model that adds a single predictor variable not already present in M_l. Choose the model that yields the highest R^2 and call it M_{l+1}.

3. Select the single best model among the $M_0, M_1, M_2, \dots, M_k$possible models using any of the $C_p\ (AIC), BIC, or\ adjusted\ R^2$ statistics [3].

Backward Stepwise Elimination

Backward Elimination also has the computational advantage over Best Subset Selection, as it considers a maximum of $\frac{k(k+1)}{2} + 1$ possible fitted models [3].

We begin the procedure with the full model, M_k, instead of the null model, M_0. We then eliminate the least significant regressor in the model, if there is any that is not statistically significant according to a criteria, which is whether the partial F-statistic is smaller than the cut-off point F_{OUT} for "kicking out" an already present regressor [8]. For simplicity purposes, we summarize an algorithm here that does not involve the partial F-statistic. Like the previous subset selection methods, the Backward elimination algorithm was taken from *Introduction to Statistical Learning with Applications in R* and summarized here:

1. Begin by fitting the full model, M_k.
2. For $l = k, k-1, \dots, 1$, consider every of the l possible fitted model that has one less predictor variable than M_l. Choose the model that yields the highest R^2 and call it M_{l-1}.
3. Select the single best model among the $M_0, M_1, M_2, \dots, M_k$ possible models using any of the $C_p\ (AIC), BIC, or\ adjusted\ R^2$ statistics [3].

Stepwise Regression

Stepwise Regression combines both forward selection and backward elimination to create a procedure that keeps the computational advantages of these two methods while mimicking a close approach to the best subset selection method. In this procedure, we have two cut-off F-statistics, F_{IN} *and* F_{OUT}, for "taking in" a new regressor into the model or "kicking out" an already present regressor, respectively. The value of F_{IN} is typically chosen to be larger than F_{OUT} so that it is more difficult to add a new regressor to the model than to delete one [1]. An algorithm for the stepwise regression method is not provided in *Introduction to Statistical Learning with Applications in R*. However, we still summarize the procedure here, as it is explained in *An Introduction to Statistical Methods and Data Analysis*, 7th Ed.

1. Begin by fitting the null model, M_0, with no predictor variables.
2. Add to the model the predictor that yields the largest simple correlation with the response variable y.
3. For $l = 1, \dots, k - 1$:
    a. Consider every $k - l$ fitted model that adds a single predictor variable not already in the M_l model. If any of the $k - l$ models that have a partial F statistic greater than a pre-selected value F_{IN}, select the one that has the highest partial F statistic and name it M_{l+1}. If none of the models have a partial F statistic exceeding F_{IN}, then M_l is the final model.
    b. Once the M_{l+1} model is obtained, check if any of the predictors already in the model yield a partial F statistic lower than F_{OUT}. If so, remove the predictor with the lowest partial F statistic. Continue repeating this process until all irrelevant predictor variables have been removed.
    c. Repeat steps (a) and (b) until no more regressors are added or removed [8].

Disadvantages of Subset Selection Methods in High Dimensional Data

The subset selection methods mentioned above have their own advantages and disadvantages when dealing with low-dimensional data. But what if the number of observations n is larger than the number of predictor variables k? Best subset selection considers *all* possible models, so it cannot be used in high dimensional data because some models would be overfitting the data or the variances of the estimates would be infinite. Backward elimination starts with *all* regressors in the model, so it cannot be used in high dimensional data either. Forward selection, on the other hand, starts with the null model, so we can add predictor variables into the fitted model while n is greater than k. However, as mentioned previously, forward selection has the disadvantages of keeping previously added regressors even if they before irrelevant when other regressors are added because of collinearity [3]. So even though forward selection is our only viable subset selection method when dealing with high dimensionality, it might not be effective at picking the best model.

Regression Trees and Random Forest

As mentioned before, several methods exist for fitting a model of a given data set. We now focus on a method known as *Random Forest*, which can be used when dealing with high dimensional data.

Amongst the multiple methods in the Tree Class, we only focus on the Random Forest method in the present paper. However, more information on other methods, such as *bagging* or *boosting,* can be found in Gareth James' *An Introduction to Statistical Learning* *with Applications in R* or Kotu's *Predictive Analytics and Data Mining*. Before we introduce random forests, it is essential to understand the concept of *regression trees*.

Regression Trees

Regression Trees are a data mining technique that involves stratifying the predictor space into simpler regions than the original predictor space [9]. We then use the mean of the response variable y in each region as the "fitted" value for any observed data point that falls inside that region. The textbook *An Introduction to Statistical Learning with Applications in R* provides a simple algorithm for building regression trees without getting into the complexity of the formulas used in it. The regression tree algorithm is provided here:

1. Divide the predictor space into J different non-overlapping regions $R_1, R_2, \dots, R_J$, known as *leaves* or *terminal nodes*.
    a. Let $\{X\}$ be the predictor space before any splitting has occurred.
2. For every single observation of the response variable that falls into a region J, the "fitted" value of that observation is $\bar{y}_{R_j}$, the mean of all observations in region R_j, where $1 \leq j \leq J$.

When we think of splitting the predictor space, the questions "which regressor do we split first?" and "at what point should a regressor be split?" arise. In simple terms, we select a predictor variable X_j and the cutoff point c such that splitting the predictor space into the regions $\{X|X_j < c\}$ and $\{X|X_j \geq c\}$ yields the greatest reduction in SSE, the sum of squares of the error [3]. The stratifying or splitting of the regressor space continues until the SSE can no longer be reduced or until each leaf has no more than five observations [3]. Figure

5 (left) provides a visual of a regression tree that has been created using two predictor variables while figure 5 (right) is a visual of the generated regions. Note that it looks like an "upside down" tree, hence the name. Although the formulas and algorithm of regression

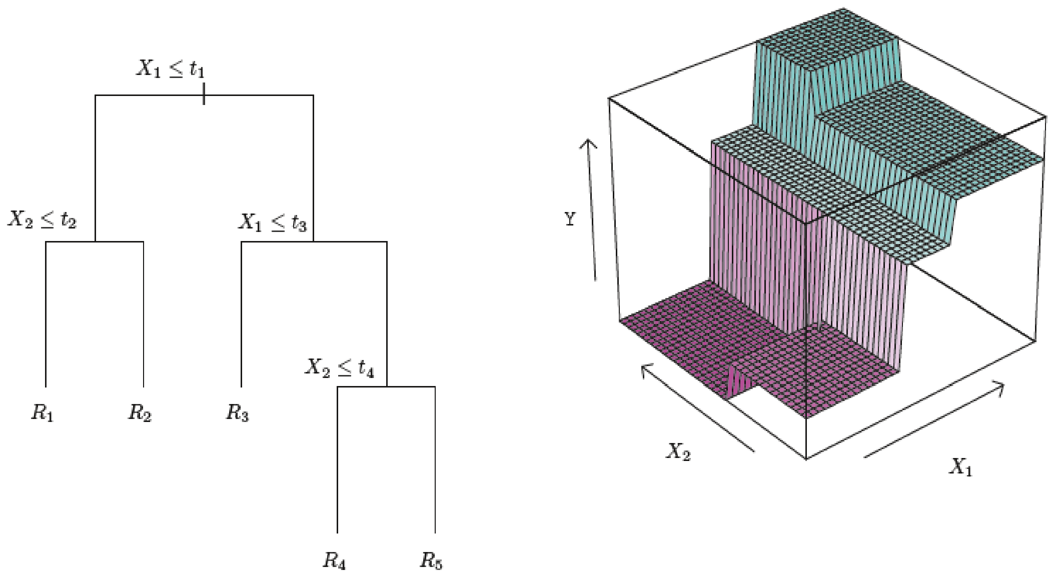


Figure 5: *Splitting of a regressor space (left) and the creation of regions in the response variable (right). (Image taken from An Introduction to Statistical Learning with Applications in R)*

trees are much more complicated, we only provide a general idea in this paper without loss of generality.

Random Forests

Recall that regression trees have a predicted value for each region created by the predictor space. In random forests, we obtain multiple de-correlated regression trees and average them [10].

Random Forest is a method that uses multiple trees $T_1, T_2, \dots, T_B$ to cast an average "fitted" value $f^B(x)$ to each observation of the response variable Y, where x represents the values of a set of predictors that correspond to an observation y. The fitted value generated by a random forest for an observation y_i is given by:

$$f_{avg}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x, R_j) = \frac{1}{B}\sum_{b=1}^{B} T_{bj}(x)$$

where B is the total number of trees and $T_b(x, R_j)$ represents the fitted value given by the jth region in the bth regression tree for a given set of predictor values x that correspond to observation y. In a simpler notation similar to the one provided to regression trees, we can see $T(x, R_{jb})$ as $\bar{y}_{R_jb}$ for set of predictor values x. An advantage of random forests include the fact that they do not overfit the data, no matter how many trees are used [3]. However, it is suggested using $B = 100$ to achieve good predicted values of the observed data [3]. Also, random forests use trees that randomly select a subset of predictor variables (usually $1/3$ of the original number of regressors), making the trees non-correlated [10]. Even though some regressors may not be optimal to a regression tree, their selection might reveal interaction effects with other predictor variables that would otherwise be ignored if a subset of regressors was not randomly selected [12]. The book *The Elements of Statistical Learning* (Hastie, Tibshirani, Jerome, 2009) provides a simple algorithm for building a random forest, which is summarized here:

1. For $b = 1, 2, \dots, B$, grow a regression tree T_b using l randomly selected predictor variables from the total k. A typical value for l is $k/3$.
    a. Note: Refer to the previous section on regression trees to know when to stop splitting/stratifying.
2. Ensemble/combine the B trees to create new regions.

3. Make predictions using the formula $f_{avg}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x, R_j)$ [10].

Suppose we have a response variable Y and three predictor variables X_1, X_2, X_3 *and* X_4. Suppose also that we decide to make a random forest using two regression trees. The first tree T_1 is built using randomly selected regressors X_1 *and* X_3, while the second tree T_2 is built using randomly selected regressors X_2 *and* X_4. Figure 6 offers a visual of this

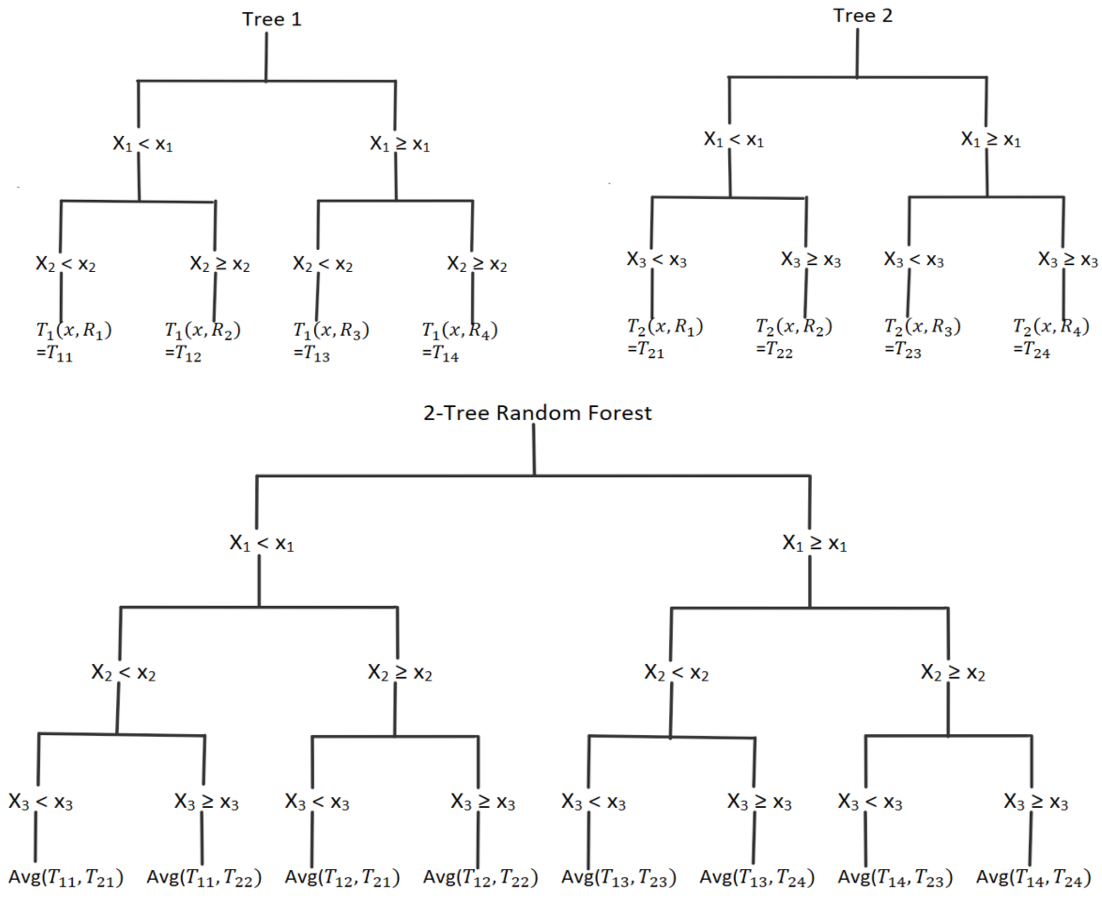


Figure 6: *A visual of a Random Forest using two trees. (Note that the fitted value of each region in the forest is the average of two regions: one from each tree. Visual created using the logic presented in this section.)*

example using the idea of random forests explained in this section. We can see from the diagram in Figure 6 that the "new regions" have a fitted value equal to the average of the corresponding fitted values in the individual trees.

It seems that random forests do a good deal of providing fitted values for observed data. However, they do not provide us with a model, but rather a logic for determining the predicted value of y for a given set of predictor values. Here we explore an example of random forests found in Genuer, Poggi, and Tuleau-Malot's "Variable selection using random forests" (2010) found in the Pattern Recognition Letters journal.

Example

The example stated in the "Variable selection using random forests" article uses the Ozone dataset, located in the $mlbench$ package in statistical program R. The Ozone dataset contains 366 observations, 163 of which contain missing values on at least one variable, so we are left with only 203 observations. The dataset contains a total of 13 variables [13], the fourth of which is used as the response variable and the remaining twelve are the predictor variables. The example used the suggested $\frac{k}{3} = \frac{12}{3} = 4$ predictors per regression tree and grow a total of 2000 trees for the random forest [13], proceeding by creating the random forest and obtaining the regressors that had a positive importance on Y-Daily maximum one-hour-average ozone. The important regressors were X_1-Month, X_5-Pressure height, X_7-Humidity, X_8-Temperature (Sandburg), X_9-Temperature (El Monte), X_{11}-Pressure gradient, and X_{12}-Inversion base temperature [13].

In the next chapter, we provide a new subset selection method that can potentially provide us with a good model, even when we are dealing with high dimensional data.

CHAPTER II

NEW SUBSET SELECTION METHOD

Now that we have a good idea on the concepts of linear regression and some of the many methods that can be used for reducing the dimensions of a data set, we can go over a new method that can potentially provide us with an effective model, whether the data has a low or high dimensionality. We begin with an overview of the new procedure, followed by a detailed explanation of each step along with any comparisons used in each step. The statistical program R was used to build the procedure. Therefore, we include a quick overview of the program code logic after each step plus a copy of the full program code in *Appendix A.*

An Overview

The proposed method consists of selecting various regressors that will make an effective model without having to go through every possible subset like best subset selection; without having to keep previously selected regressors in the model like forward selection; without having to keep previously eliminated regressors out of the model like backward elimination. For naming purposes, we will be using the name *Best Probable Subset (BPS)* for the name of this new method.

Suppose we want to reduce the dimensions of the regressor space by a certain percent, say 60%. That would mean that we want to keep 40% of the total number of regressor k. Let us say that 40% of k equals $0.40 * k = l$ regressors that we want to keep in the model. Previously seen methods, like forward selection, backward elimination, best subset selection, and step-wise regression offer ways to select a good subset of regressors l.

However, as previously discussed, they all run into issues, whether computational ineffectiveness, inability to change previously added/deleted regressors, or inability to deal with high dimensional data. The logic behind the BPS method consists of selecting a subset of l regressors using a probability pool that is calculated from the linear relationships – or squared correlations – between the response variable Y and each of the predictor variables. The probability pool, conveniently named *Linear Relationship Pool (LRP)*, is then used to select a subset of l predictors without replacement to avoid selecting the same predictor twice for the same fitted model. The LRP allows us to have models containing regressors that might not be present in other models at all. This solves the issue that we run into in forward selection, in which previously selected predictors have to stay in the fitted model [3].

Once the l regressors are chosen, we use the original LRP for all regressors to have chance of be picked in the next model, even if they were chosen in the previous model, and repeat the selection process until we have 1000 models. Note that each regressor can be in multiple models but the same regressor cannot be more than once in the same model. We decided to use 1000 as the total number of possible models because it gives each predictor a chance of being picked in at least one model, even if they have a squared correlation as small as $\frac{1}{1000} = 0.001$ with the response variable. We then calculate the coefficient of determination, R^2, for each model of l predictor variables and select the model with the highest R^2 out of the 1000. The selected model is considered to be the "best probable model" with l regressors using the BPS method, which we denote as M_l^{BPS} and the subset of regressors in that model make up the "best probable subset" of size l. When dealing with high dimensional data, this subset selection procedure is much more computationally efficient than the other subset selection methods mentioned in the previous chapter.

Figure 7 summarizes the entire procedure for finding the best probable fitted model with l regressors. The blue path serves as a guide of the steps of the BPS method while the red path are alternative steps used for comparison purposes. The key next to the diagram provides more information about the colors and shapes.

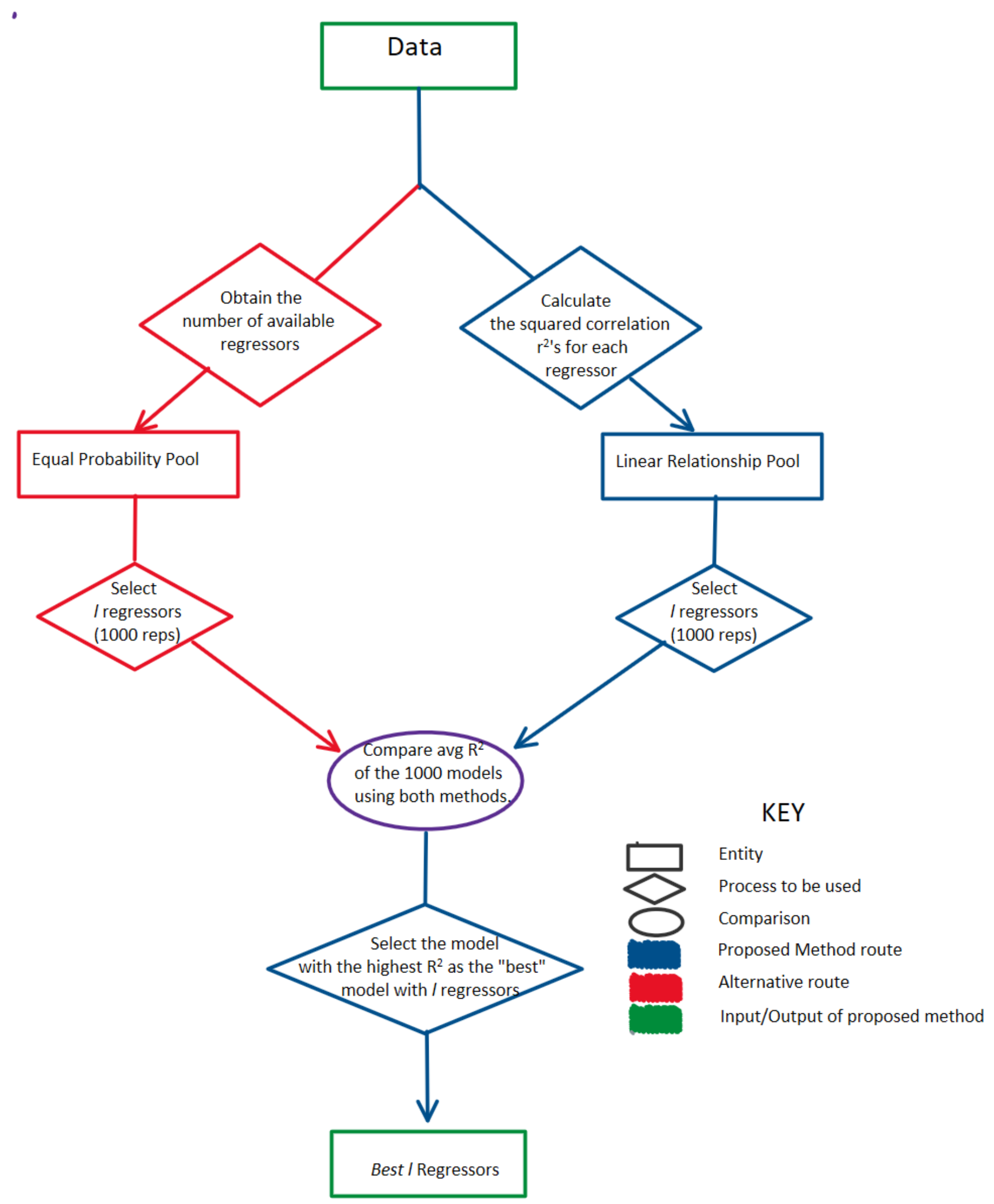
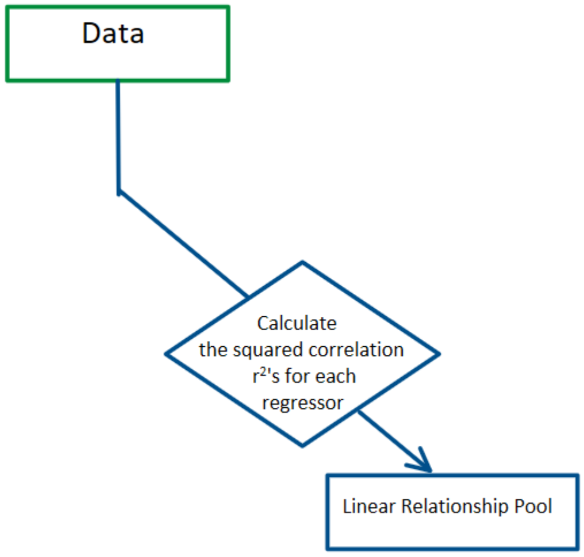

Data
Obtain the number of available regressors
Calculate the squared correlation r²'s for each regressor
Equal Probability Pool
Linear Relationship Pool
Select *l* regressors (1000 reps)
Select *l* regressors (1000 reps)
Compare avg R² of the 1000 models using both methods.
Select the model with the highest R² as the "best" model with *l* regressors
Best l Regressors
KEY
Entity
Process to be used
Comparison
Proposed Method route
Alternative route
Input/Output of proposed method

Figure 7: *Diagram of the Best Probable Subset (BPS) method.*

Step One: Linear Relationship Pool

The first step on the process is obtaining our sampling plan, the Linear Relationship Pool (LRP), which is created by calculating the squared correlation,r^2, of each predictor variable with the response variable. We then use these squared correlations and divide each of them by the sum of all squared correlations, giving us a probability of selection, p, for each regressor. Each of these probabilities can be calculated using the formula:

$$p_j = P(X_j) = \frac{r_j^2}{\sum_{i=1}^{k} r_i^2}, 1 \leq j \leq k$$



where k is the total number of regressors in the dataset and r_j^2 is the marginal R-squared of the jth regressor with the response variable Y. Once we have calculated every p_j for $1 \leq j \leq k$, we have completed the LRP and are ready to proceed to the next step.

Figure 8: *Step One: Creating the Linear Relationship Pool (LRP)*

Coding Equivalent

In coding, we create the LRP using the following steps:

1. Calculate the squared correlation of each predictor with the response variable.
2. Calculate the probability of selection, p, for each predictor variable.

3. Find the number of decimal places, m, until the first non-zero digit for the smallest probability.
4. Create a vector in which integer j appears $round(p_j * 10^{m+1})$ times, representing the j^{th} regressor. The value m makes sure that the regressor with the smallest probability is represented at least once in the vector.

Example

Suppose we have three regressors, X_1, X_2, X_3 with marginal squared correlations equal to $0.070, 0.800, and\ 0.002$, respectively. Their probabilities of being selected would be $\frac{0.07}{0.07+0.8+0.002} = 0.080, 0.917, and\ 0.003$, respectively. The smallest probability is 0.003, which has two decimal places before the first non-zero digit, so $m = 2$. In R, we would have a vector where "1" (for Regressor X_1) appears $round(0.080 * 10^{2+1}) = 80$ times, "2" (for Regressor X_2) appears $round(0.917 * 10^{2+1}) = 917$ times, and "3" (For Regressor X_3) appears $round(0.003 * 10^{1+1}) = 3$ times. This step completes our linear relationship pool and the vector to be used in R for the selection of l regressors.

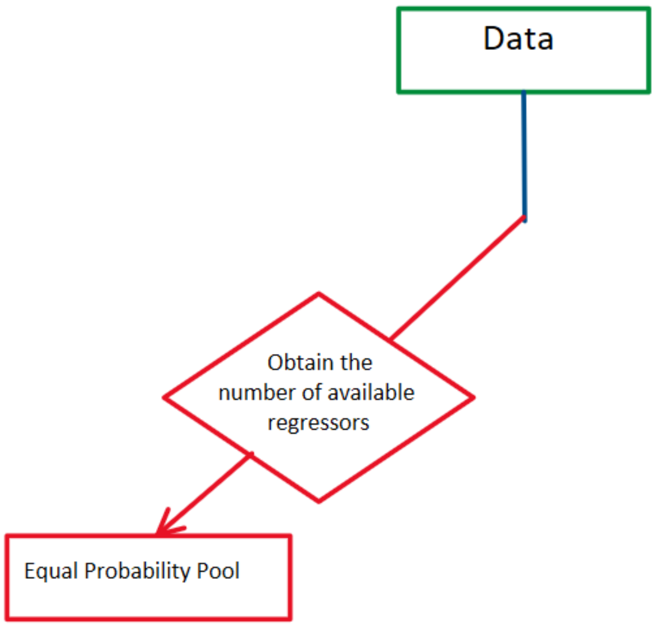


Figure 9: *Step One: Alternative sampling plan.*

Comparison Step: Equal Probability Pool

For comparison purposes, we also use random selection of l regressors without replacement for each of the 1000 fitted models. We use this alternative sampling plan to compare the R^2s generated by the 1000 models with l regressors selected using the LRP and the R^2s generated by the 1000 models with l regressors selected at random. Equal probabilities means that all predictor variables have the same chance of being selected. Therefore, the probabilities for each regressor is:

$$p = p_j = P(X_j) = \frac{1}{k}, 1 \le j \le k$$

Where k is the total number of regressors and p_j is the probability of selecting the jth regressor.

Coding Equivalent

In coding, we create the pool of equal probabilities using the following steps:

1. For $j = 1, 2, \ldots, k$: create a vector in which integer j appears one time, representing the jth predictor variable.
    a. Note: When coding, we can use $seq(1{:}\,k)$ to create this vector.

Example

Suppose we have the same three regressors, X_1, X_2, X_3 as in the previous example with the same squared correlations $0.070, 0.800, and\ 0.002$, respectively. Their probabilities of being selected at random would be $1/3$ for each of them. Therefore, the vector of integers in R contains $"1", "2", and\ "3"$ once each.

Step Two: Selection of the Predictor Variables

Now that we have the linear relationship pool, the next step in the procedure is selecting the l predictor variables using the probabilities. For comparison purposes previously mentioned, we will be selecting a subset of l predictors 1000 times using the LRP and another 1000 times using the equal probabilities pool. Both methods of selection can be done using a computer program. We used computer program R for both random and non-random selections.

Coding Equivalent

Using Statistical Computer Program R, this can be easily done by using the command:

$$sample(Prob, 1, T)$$

where *Prob* is the probability pool being used, whether LRP or the equal probabilities pool for selecting at random. This command selects an integer from a vector of integers and deletes all of its repetitions. The "T" – which represents *True* – deletes any replicates of the selected number. Deleting replicates avoids selecting the same regressor more than once for the same fitted model; it is our way of selecting l regressors without replacement.

Example

Recall our previous example, where the three regressors, $X_1, X_2, and\ X_3$ have a LRP with probabilities $0.080, 0.917, and\ 0.003$, respectively, of being selected using the R-Squared Probability pool. Suppose that we want to select $l = 2$ out of the three total regressors. Let us assume that X_2 is selected first since it has a significantly larger probability of being than the others. Using $sample(LRP, 1, T)$, once the integer "2" is selected – which represents regressor X_2 – the 917 repetitions of the integer "2" will be deleted from the vector of integers, leaving us with 80 repetitions of integer "1" and 3 repetitions of integer "3". Since we are selecting two regressors without replacement, every time we select another regressor for the fitted model the probabilities of those regressors that have not yet been selected are updated. This example is just for the sake of showing the logic of the method. It would be totally unnecessary to produce 1000 models with $l = 2$ out of three regressors.

Comparing the Outcome of Using both Probability Pools

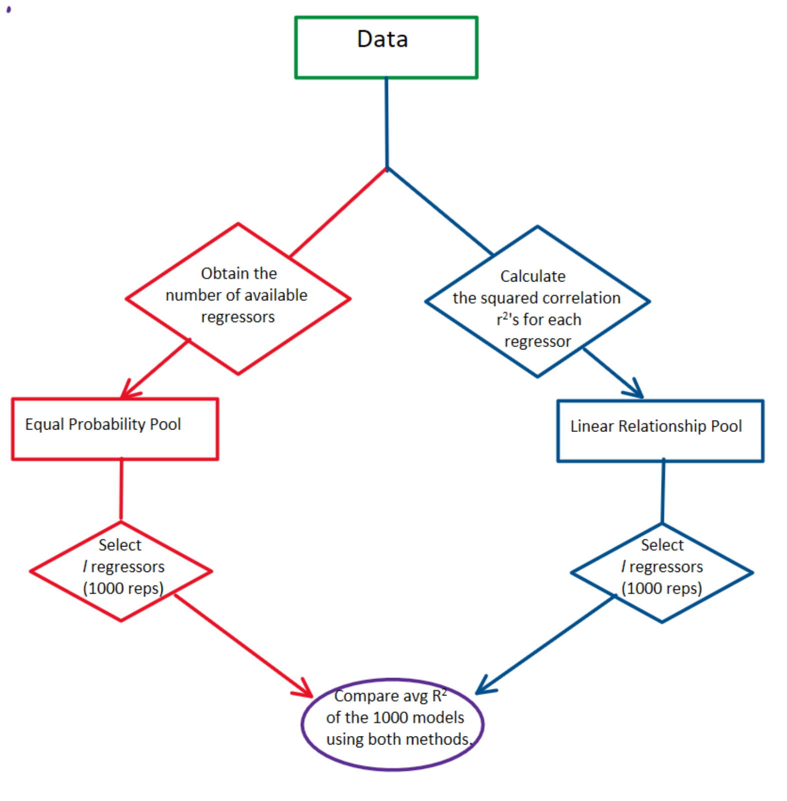

Data
Obtain the number of available regressors
Calculate the squared correlation r^2's for each regressor
Equal Probability Pool
Linear Relationship Pool
Select *l* regressors (1000 reps)
Select *l* regressors (1000 reps)
Compare avg R^2 of the 1000 models using both methods

Figure 10: *Step Two: Selecting a subset of l regressors using the preferred and alternative sampling plans and a comparison between the two.*

We have been using two probability pools –the LRP and the equal probabilities pool– to select l out of k regressors. In order to determine whether which probability pool is indeed more effective, we only need to compare the average R-Squared of the 1000 fitted models produced by selecting l regressors using the LRP and the average R-Squared of the 1000 fitted models produced by selecting a subset of l regressors using the random probability pool. In short, we just need to show that:

$$Avg(R^2_{LRP}) > Avg((R^2_{EPP})$$

Where LRP is means “linear relationship pool” and EPP means “equal probabilities pool” for selecting at random. We should expect to obtain models with higher R^2s when using the LRP since the regressors that have a higher squared correlation with the response variable have a higher chance of being selected in the subset of l regressors. We omit showing this in the examples in Chapter IV, but can be demonstrated using with the code provided in Appendix A.

It should be clear that using the LRP is more effective at creating 1000 fitted models with l regressors having a higher average R^2 than the 1000 fitted models with l regressors chosen at random. We now continue to the next step in the BPS method assuming that only the LRP will be used for the selection of l regressors.

Step Three: Choosing the Model with the Best l Regressors

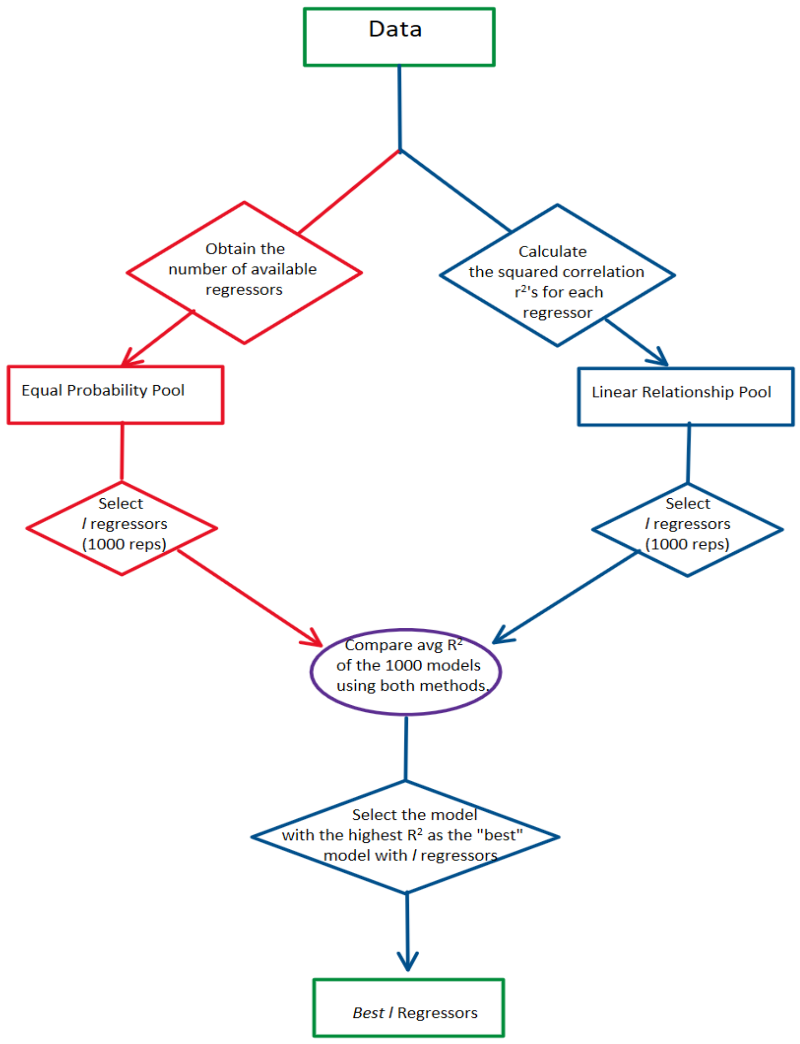

Data
Obtain the number of available regressors
Calculate the squared correlation r²'s for each regressor
Equal Probability Pool
Linear Relationship Pool
Select l regressors (1000 reps)
Select l regressors (1000 reps)
Compare avg R² of the 1000 models using both methods.
Select the model with the highest R² as the "best" model with l regressors
Best l Regressors

Figure 11: *Step Three: Selecting the model with the best subset of l.*

After creating the linear relationship pool and using it to select a subset of l regressors when fitting the 1000 models, the next step is to determine which of the 1000 models has the best subset of l out of the k original regressors. To choose the best model, we must first decide what criteria would make one model to be the best out of the 1000.

Criterion for Selecting the Best Subset of Size l

We covered in Chapter I that some measures of fit include *Mean Squared Error (MSE)*, the *coefficient of determination* R^2, and theR^2_{adj}. We also mentioned that R^2 and adjusted R^2 have the advantage of having a range between 0 and 1 and less than 1, respectively, making them easier to understand that MSE [3]. Lastly, we explained that R^2 continues to increase when we keep adding more regressors to the model, making it more useful for comparing fitted models with the same number of regressors.

The dilemma we currently have is determining which model(s) out of the generated 1000 is the best probable model, M_l^{BPS}, containing the best subset of l predictors. Since all of them have the same number of predictors l, we can use the regular R^2 to measure their fit. Since a large R^2 indicates a better fit for equally sized models, we just need to select the model with l regressors that yields the highest R^2.

Coding Equivalent

Now that we have chosen a measure of fit as the criteria for choosing the best fitted model with l predictor variables, we only need to find out which model has the highest R^2. In order to do this using R, we simply calculated the R^2 for each of the 1000 models having l predictors and sorted them from largest to smallest using the command:

$$sort(RSquares, decreasing = TRUE)$$

where RSquares is a vector containing the 1000 R^2 and the subsection “decreasing = TRUE” tells R to sort the numbers from largest to smallest. We select the highest R^2 and the unique model that yields it.

Step Four: Determining the Best Subset Size l

Up to now we have assumed that the reduction amount that we want to perform on a data is known, making it clear what the subset size l is. But what happens if we do not know what the reduction amount is? Or what if we want to select the best subset, no matter what the size of it is? Assuming that the sample size n is much greater than the number of regressors k, we are left with $k + 1$ possible best probable models – $M_0^{BPS}, M_1^{BPS}, \dots, M_k^{BPS}$ each of them being the best probable subset of size $0, 1, \dots, k$, respectively – and no direction on where to go next. We proceed by determining a criteria for selecting the single best probable subset without having a pre-set subset size.

General Criterion for Selecting the Best Subset of Any Size

In the previous section, we used the R-Squares to determine the best fitted linear model out of the 1000 models generated using the Linear Relationship Pool. Assuming that we fit a model with the best probable subset for all possible subset sizes, we are left with $M_0^{BPS}, M_1^{BPS}, \dots, M_k^{BPS}$, a total of $k + 1$ plausible models, where M_0^{BPS} is the null model and M_k^{BPS} is the full model containing all regressors. Since the $k + 1$ best probable models contain different number of regressors, we need other criteria other than the R^2s to determine the best one out of them.

As mentioned in Chapter I, the ${R^2}_{adj}$ statistic offers flexibility when comparing subsets of different sizes because, unlike the non-adjusted R^2, it is penalized for the addition of unnecessary regressors [3]. Furthermore, ${R^2}_{adj}$ is simple to understand as it goes up to 1. Because of these reasons, we decided to use this statistic as the criterion for choosing the best possible subset out of the $k + 1$ subsets to consider.

The ${R^2}_{adj}$ increases when an added regressor contributes to the fit of the model, so the best out of the $k + 1$ plausible fitted models $M_0^{BPS}, M_1^{BPS}, \dots, M_k^{BPS}$ should be the one yielding the highest ${R^2}_{adj}$. We will use the fitted model having the highest ${R^2}_{adj}$ as the general criteria for choosing the best probable subset. We denote the linear model containing the best probable subset as M^{BPS}, without a subscript.

Coding Equivalent

Similarly to the coding logic in the previous step, we simply calculate the ${R^2}_{adj}$ for each of the $k+1$ models, each having the best probable subset of size $0, 1, \dots, k$ and sort the adjusted R^2s using the command:

$$sort(AdjustedRSquares, decreasing = TRUE)$$

where AdjustedRSquares is a vector containing the $k+1$ ${R^2}_{adj}$ and the subsection "decreasing = TRUE" tells R to sort the numbers from largest to smallest. We select the highest adjusted R^2 and the unique model that yields it, M^{BPS}.

High Dimensional Data and Stopping Rules

Recall from Chapter I that data having a large number of predictors, k, that equals or exceeds the number of observations, n, are known to have high dimensionality, and that the least squares estimation does not work with such data [3]. Recall also that forward selection is the only alternative subset selection methods when we are dealing with such data. Here we explain how the best probable subset method deals with high dimensional data while keeping the number of fitted models to a minimum by creating some stopping rules.

Stopping Rules

We begin by creating a stopping rule so that the number of predictors in the model never exceeds, or even approaches, the number of observations. For this paper, we want the number of observations to be at least 5 times larger than the number of regressors in the fitted mode, limiting l to be less than or equal to $n/5$. The rule will allow us to have five observations per regressor and let us always use the BPS method while dealing with low or high dimensional data. Note that the $n/5$ limit on l can be changed to any other number less than k, and there is no statistical proof that this limit will always be the best; the limit is specified for the purpose of this paper can be changed accordingly as long as it is less than the number of observation n.

Using the stopping rule above, we would still need to consider various best probable models of different sizes, $M_0^{BPS}, M_1^{BPS}, \dots, M_{n/5}^{BPS}$. If $n/5$ were to equal 100, we would still be considering $100 + 1$ best possible models of different sizes, each of which was selected from 1000 models, with the exception of the null model M_0^{BPS}. The total number of fitted models would then equal $100 * 1000 + 1 = 100{,}001$, which can be computationally inefficient. To improve the situation, we created a second stopping rule, which states that we will continue to find the best possible subset of the next size until the adjusted R-squared ceases to increase by more than 1%. In short, we begin by finding M_0^{BPS}, followed by M_1^{BPS}, followed by M_2^{BPS}, etc. until M_i^{BPS} yields a lower ${R^2}_{adj}$ than M_{i-1}^{BPS}, where $1 \leq i \leq k$. At that point M_{i-1}^{BPS} is the best probable model. Note that the 1% was selected for the present paper but there could be other stopping rules that are just as reliable.

When we combine the stopping rules described above, we obtain the following stopping criteria:

1. Begin by finding the null model M_0^{BPS}.
2. For $1 \leq i \leq \min(k, \frac{n}{5})$, continue finding the model with the best possible subset size i until M_i^{BPS} yields an ${R^2}_{adj}$ than is less than 1% greater than the ${R^2}_{adj}$ of M_{i-1}^{BPS}.

Once we have reached the stopping rule, we are left with $i + 1$ best possible models of different sizes. Since the adjusted R-squared continues to increase by more than 1% until M_{i-1}^{BPS}, the best probable model is M_{i-1}^{BPS}, so $M^{BPS} = M_{i-1}^{BPS}$.

Best Probable Method Algorithm

The *Best Probable Subset (BPS)* method is summarized here:

1. Create a probability pool, called *Linear Relationship Pool (LRP)*, for selecting a subset of regressors by using the following formula for each regressor of the k regressors.

   $$p_j = P(X_j) = \frac{r_j^2}{\sum_{i=1}^{k} r_i^2}, 1 \leq j \leq k$$

   Where p_j is the probability of choosing the jth predictor for the reduced model and r_j^2represents the squared correlation between the jth predictor and the response variable.

2. Begin by finding the null model M_0^{BPS}. For $1 \leq i \leq \min(k, \frac{n}{5})$:

   a. Select a subset of size i without replacement from the predictor by using the LRP. Repeat 1000 times and select the model with the highest R^2. Call it M_i^{BPS}.

   b. Continue finding the model with the best probable subset size i until M_{i+1}^{BPS} yields an ${R^2}_{adj}$ than is less than 1% greater than the ${R^2}_{adj}$ of M_i^{BPS}.

3. Since the adjusted R-squared continues to increase by more than 1% until M_{i-1}^{BPS}, the best probable model is $M^{BPS} = M_i^{BPS}$.

CHAPTER III

DATA SIMULATION

In order to validate the BPS method, we simulated some linear regression models and generated data from them. In Chapter III we cover how we generated both the models and the data.

Simulating Multiple Regression Linear Models

Our next goal is to prove that the BPS method can pick up a subset of regressors that make an effective fitted model for observed data. To do so, we create multiple regression linear models and generate "observed" data from them. Without loss of generality, we decided to create simulate three linear models, each containing a different total number of regressors – $k = 10, 25, and\ 50$. To create data sets containing significant and insignificant regressors, we made the response variable using only the first 20% of the regressors. The data for each regressor, whether significant or insignificant to the model, were generated using one randomly chosen distribution from this list:

- Chi-Squared (DF is a random integer from 1 to 10)
- Gamma (α is a random integer from 1 to 10; β is a random integer from 1 to 10)
- Normal (μ is a random integer from 1 to 10; σ is a random integer from 1 to 10)

Once the data for the first 20% of the regressors are generated, the observations for the response variable were created using the model:

$$Y = \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_l X_l + \varepsilon$$

where X_1, $X_2, \dots, X_l$ are the first 20% and only significant regressors and the regression parameters are randomly generated integers from -10 to 10, excluding 0. The error term ε is distributed as $N(0, \sigma^2)$, where σ^2 is specified in each dataset. Once the observations of the response variable are generated using the above model, we simulate data for the other 80% of the regressors, each having a regression parameter equal to zero and a distribution chosen at random from the list:

- Uniform (where x is between 0 and 1)
- Poisson (λ is a random integer from 1 to 10)
- Binomial (p is a random number between 0 and 1; n is a random integer form 1 to 10)
- Beta (α is a random integer from 1 to 100; β is a random integer from 1 to 10)
- Chi-Squared (DF is a random integer from 1 to 10)
- Exponential (β is a random integer from 1 to 10)
- Gamma (α is a random integer from 1 to 100; β is a random integer from 1 to 10)
- Normal (μ is a random integer from 1 to 100; σ is a random integer from 1 to 10)

To compare the BPS method with already existing methods, we need the number of observations to be well above the number of regressors. The three data sets mentioned in this section will have a sample size equal to five times the total number of regressors, k.

A Data Set with High Dimensionality

We created a 4th data set, also having a total of 100 regressors, which have 80 observations and only the first 20 regressors will be significant to the model. We will use this high dimensional data set to compare it to forward selection and random forests, both of which can be used in data with high dimensionality.

Lastly, we reference back to the random forest example provided at the end of Chapter I, and it to the BPS method by looking at the MSE that each of the two methods produced using the Ozone dataset.

DATA COLLECTION

The section walks the reader through some existing data sets used in this paper. The data set, named "U.S. News and World Report's College Data," was collected from the "ISLR" package in R, and belong to their respective publishers. The data set, along with many others, was made public and can be used freely. Without further delay, here is a summary of the data:

"College" – U.S. News and World Report's College Data

The College data set offers statistics for a large number of Colleges, including graduation rates, enrollments, tuition, and more. The data was collected from the 1995 US News and World Reports[11]. The following information is a summary from the dataset description in R:

R Package

"ISLR"

Description

The College data has 777 observations and 18 variables.

Format

This data frame contains the following columns:

“Private	Indicates ‘Yes’ or ‘No’ as a factor
Apps	Number of applications received
Accept	Number of applications accepted
Enroll	Number of new students enrolled
Top10perc	Percent of new students from the top 10% of their H.S class
Top25perc	Percent of new students from the top 25% of their H.S class
F.Undergrad	Number of full time undergraduates
P.Undergrad	Number of part time undergraduates
Outstate	Out-of-state tuition
Room.Board	Costs for room and board
Books	Estimated cost of books
Personal	Estimated amount of personal spending
PhD	Percent of faculty with Ph.D.’s
Terminal	Percent of faculty with terminal degrees
S.F.Ratio	Student/faculty ratio
Perc.alumni	Percent of alumni who donate to the College
Expend	Instructional expenditure per student
Grad.Rate	Graduation rate” [11].

Source

Dataset was taken from the Statistics library, StatLib, maintained at Carnegie Mellon University. The dataset was also used in the 1995 Data

Analysis Exposition, sponsored by the ASA Statistical Graphics Section [11].

For more information on this dataset, refer to the ISLR package documentation.

CHAPTER IV

COMPARISONS AND RESULTS

In Chapter IV we apply the BPS and other existing methods to each data set to provide some insight on the method's functionality. In the first section, we use simulated data sets that follow the linear regression assumptions in order to validate the new method and contain low to moderate correlation between regressors. After, we use a data set that is not as perfect and compare the results with some existing methods.

Validation of the BPS Method

We begin the validation of the BPS method by using a simulated data sets with only ten predictor variables. For each data set we use, we provide a list of the contributing regressors, which are those that were used to build the observations of the response variable Y. Since we are simulating the data using linear regression models with the error term distributed as $N(0, \sigma^2 = 1)$, there is no need to check the model assumptions. We do provide the correlation matrix to the first data set. After that, the correlation matrices are too large, so we only provide the highest correlation coefficient between two different regressors to show that there is no collinearity amongst regressors.

Dataset # 1:

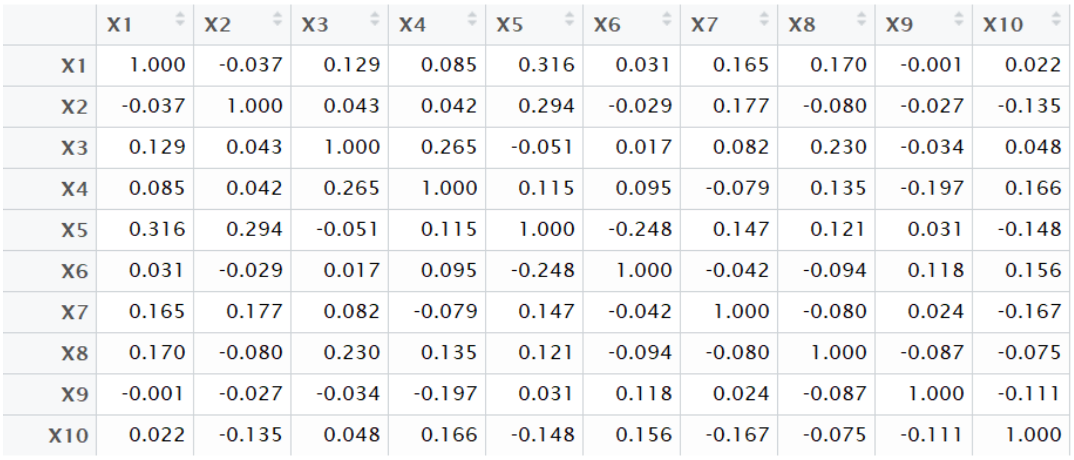
Number of Predictors: 10

Contributing Predictors: X_1, X_2

Sample Size: 50

Error Term: $\varepsilon \sim N(0, \sigma^2 = 2)$

Correlation Matrix:

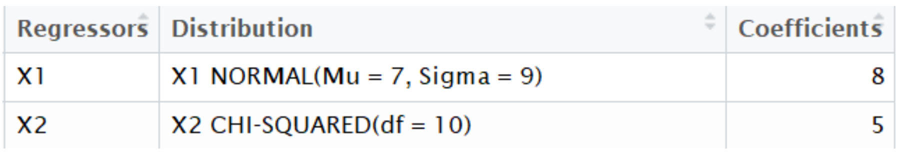


	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
X1	1.000	-0.037	0.129	0.085	0.316	0.031	0.165	0.170	-0.001	0.022
X2	-0.037	1.000	0.043	0.042	0.294	-0.029	0.177	-0.080	-0.027	-0.135
X3	0.129	0.043	1.000	0.265	-0.051	0.017	0.082	0.230	-0.034	0.048
X4	0.085	0.042	0.265	1.000	0.115	0.095	-0.079	0.135	-0.197	0.166
X5	0.316	0.294	-0.051	0.115	1.000	-0.248	0.147	0.121	0.031	-0.148
X6	0.031	-0.029	0.017	0.095	-0.248	1.000	-0.042	-0.094	0.118	0.156
X7	0.165	0.177	0.082	-0.079	0.147	-0.042	1.000	-0.080	0.024	-0.167
X8	0.170	-0.080	0.230	0.135	0.121	-0.094	-0.080	1.000	-0.087	-0.075
X9	-0.001	-0.027	-0.034	-0.197	0.031	0.118	0.024	-0.087	1.000	-0.111
X10	0.022	-0.135	0.048	0.166	-0.148	0.156	-0.167	-0.075	-0.111	1.000

Table 1: *Correlation Matrix – Dataset # 1*

Non-Zero Correlation Coefficients:

The following table provides the distribution and true regression coefficients for the regressors with non-zero regression coefficients in Dataset # 1. All other regressors have a true regression coefficient of zero.



Regressors	Distribution	Coefficients
X1	X1 NORMAL(Mu = 7, Sigma = 9)	8
X2	X2 CHI-SQUARED(df = 10)	5

Table 2: *Regression Coefficients – Dataset # 1*

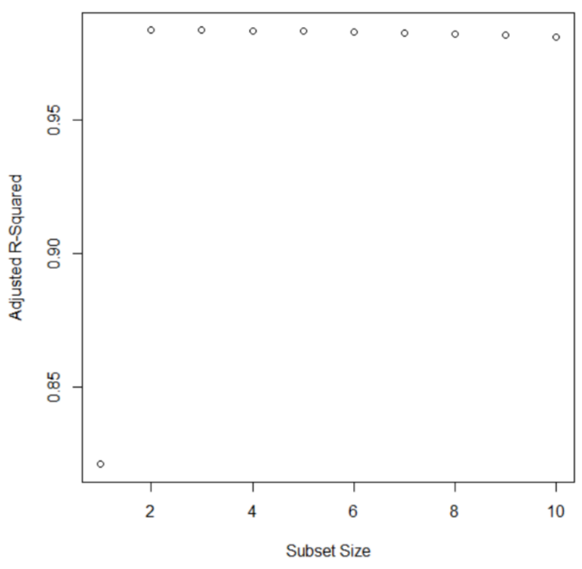
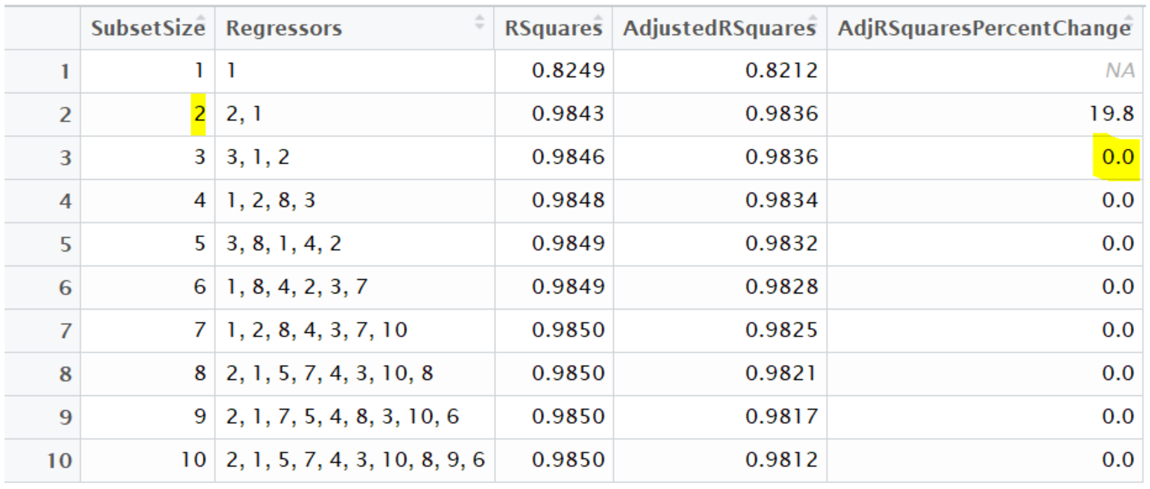


Figure 12: *Adjusted R-Squares vs. Subset Size Plot – Dataset # 1*

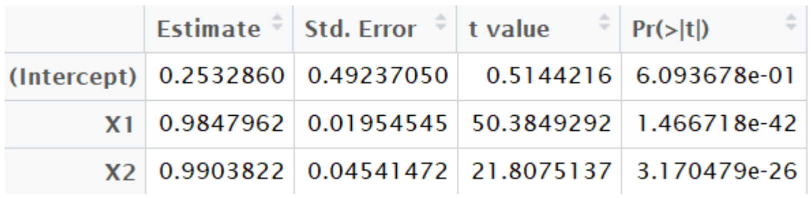
Using the statistical program R, we generated 1000 models for each subset size i, where $1 \leq i \leq 10$. From each group of 1000 models, the best model of size i, denoted as M_i^{BPS}, is selected. The following plot and table show the adjusted R-squares for each M_i^{BPS} model.



	SubsetSize	Regressors	RSquares	AdjustedRSquares	AdjRSquaresPercentChange
1	1	1	0.8249	0.8212	NA
2	2	2, 1	0.9843	0.9836	19.8
3	3	3, 1, 2	0.9846	0.9836	0.0
4	4	1, 2, 8, 3	0.9848	0.9834	0.0
5	5	3, 8, 1, 4, 2	0.9849	0.9832	0.0
6	6	1, 8, 4, 2, 3, 7	0.9849	0.9828	0.0
7	7	1, 2, 8, 4, 3, 7, 10	0.9850	0.9825	0.0
8	8	2, 1, 5, 7, 4, 3, 10, 8	0.9850	0.9821	0.0
9	9	2, 1, 7, 5, 4, 8, 3, 10, 6	0.9850	0.9817	0.0
10	10	2, 1, 5, 7, 4, 3, 10, 8, 9, 6	0.9850	0.9812	0.0

Table 3: *Best Probable Subset of Size l – Dataset # 1*

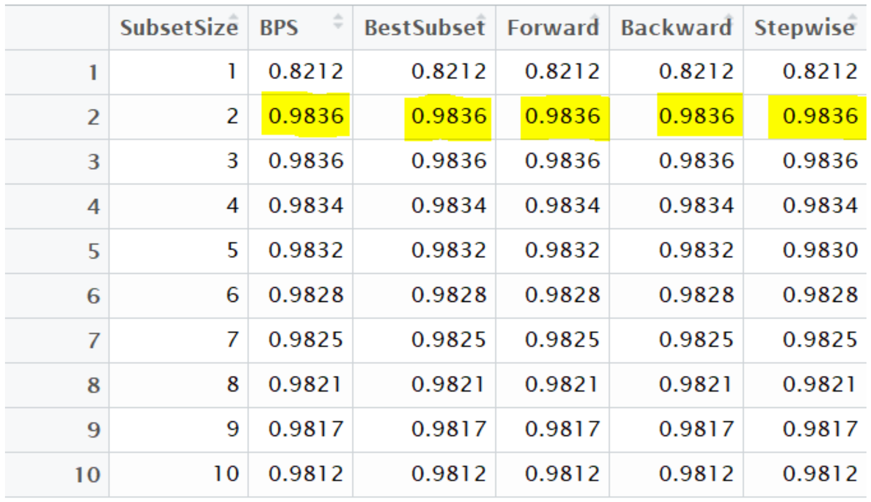
Note that the AdjRSquaresPercentChange contains the percent changes of the R_{adj}^2. According to our stopping rule, we should select the best probable subset of two regressors, X_1 *and* X_2, which yield the following LS estimates:



	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.2532860	0.49237050	0.5144216	6.093678e-01
X1	0.9847962	0.01954545	50.3849292	1.466718e-42
X2	0.9903822	0.04541472	21.8075137	3.170479e-26

Table 4: *Best Probable Subset LS Estimates – Dataset # 1*

When we compare the fitted model to the population linear regression model, we can see that we obtained the same regressors, but very different estimates for the linear regression coefficients. The following R_{adj}^2 table contains useful information for comparing the different subset selection methods. The table provides the R_{adj}^2 for each subset size using each subset selection method. The cells highlighted in yellow indicates the R_{adj}^2 value for the best model selected by each method.



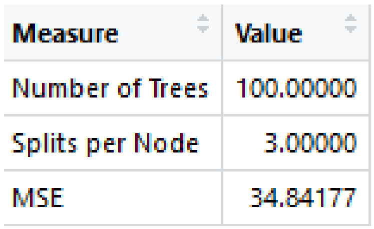
	SubsetSize	BPS	BestSubset	Forward	Backward	Stepwise
1	1	0.8212	0.8212	0.8212	0.8212	0.8212
2	2	0.9836	0.9836	0.9836	0.9836	0.9836
3	3	0.9836	0.9836	0.9836	0.9836	0.9836
4	4	0.9834	0.9834	0.9834	0.9834	0.9834
5	5	0.9832	0.9832	0.9832	0.9832	0.9830
6	6	0.9828	0.9828	0.9828	0.9828	0.9828
7	7	0.9825	0.9825	0.9825	0.9825	0.9825
8	8	0.9821	0.9821	0.9821	0.9821	0.9821
9	9	0.9817	0.9817	0.9817	0.9817	0.9817
10	10	0.9812	0.9812	0.9812	0.9812	0.9812

Table 5: *Subset Selection Methods R_{adj}^2 Comparisons – Dataset # 1*

At first glance, we can observe that the BPS method has the same R_{adj}^2 values as most subset selection methods for all subset sizes. When we look at the table, we can see that the largest R_{adj}^2 for all methods occurs when the subset size equals 2. According to the subset selection criteria explained in chapter 1, all pre-existing subset selection methods should choose $l = 2$ as the best subset size. The BPS method does not take any more regressors once the adjusted R-Squared value increases by less than 1%. In this case, the R^2_{adj} increased $\frac{0.9846-0.9843}{0.9843} * 100\% = 0.03\%$, so we stay at $l = 2$ regressors in the best probable model.

Lastly, we compare the Mean Square Error (MSE) of the best probable model with the MSE that we get by building a random forest, which we denote as MSE_{BPS} and MSE_{RF}, respectively. Recall from chapter 1 that the suggested number of trees for random forests is 100 and that the number of regressors that should be considered at each node is approximately $n/3$ for regression random forests. We provide a summary table of the random forest we obtained using dataset # 1:

The MSE_{BPS} we obtained for dataset # 1 equals 1.2208, a much lower number than



Measure	Value
Number of Trees	100.00000
Splits per Node	3.00000
MSE	34.84177

Table 6: *Random Forest Summary – Dataset # 1*

$MSE_{RF} = 34.8418$. Note that the BPS method is more effective than Random Forests in this data set. This does not mean or imply, however, that the BPS will always be more effective at predicting the values of the response variable.

Dataset # 2:

Number of Predictors: 25

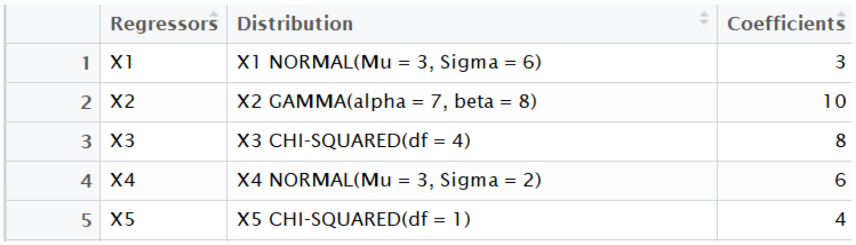
Contributing Predictors: $X_1, X_2, \dots, X_5$

Sample Size: 125

Error Term: $\varepsilon \sim N(0, \sigma^2 = 5)$

Maximum Pair-Wise Correlation Value: 0.273

Non-Zero Regression Coefficients:



	Regressors	Distribution	Coefficients
1	X1	X1 NORMAL(Mu = 3, Sigma = 6)	3
2	X2	X2 GAMMA(alpha = 7, beta = 8)	10
3	X3	X3 CHI-SQUARED(df = 4)	8
4	X4	X4 NORMAL(Mu = 3, Sigma = 2)	6
5	X5	X5 CHI-SQUARED(df = 1)	4

Table 7: *Regression Coefficients – Dataset # 2*

Similar to the first data set, we generated 1000 models for each subset size i, where $1 \leq i \leq 25$, using R. From each group of 1000 models, the best model of size i, denoted as M_i^{BPS}, is selected. The following plot and table show the adjusted R-squares for each M_i^{BPS}.

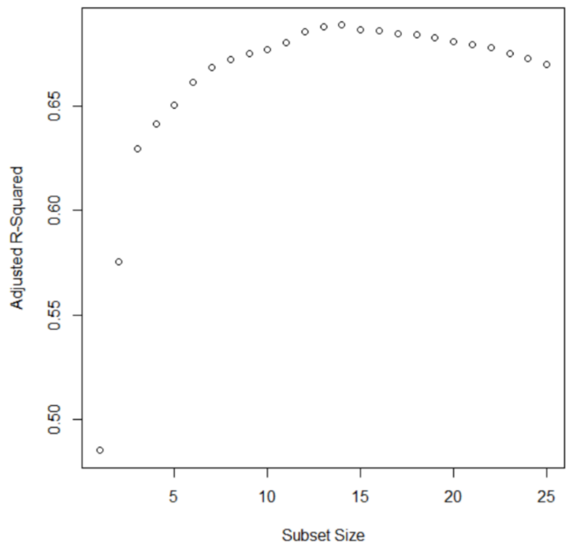
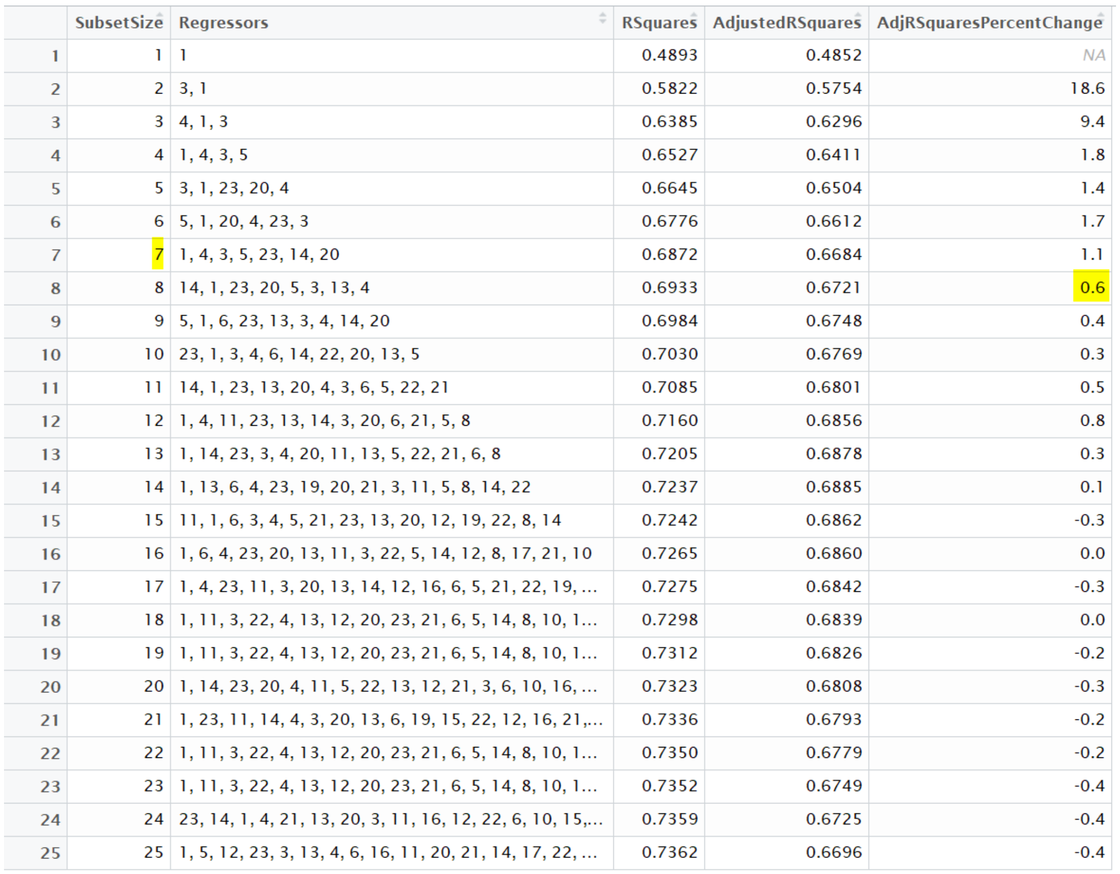


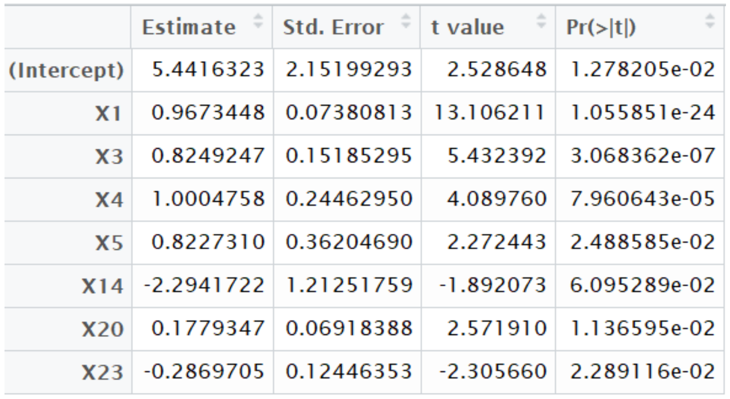
Figure 13: *Adjusted R-Squares vs. Subset Size Plot – Dataset # 2*



	SubsetSize	Regressors	RSquares	AdjustedRSquares	AdjRSquaresPercentChange
1	1	1	0.4893	0.4852	NA
2	2	3, 1	0.5822	0.5754	18.6
3	3	4, 1, 3	0.6385	0.6296	9.4
4	4	1, 4, 3, 5	0.6527	0.6411	1.8
5	5	3, 1, 23, 20, 4	0.6645	0.6504	1.4
6	6	5, 1, 20, 4, 23, 3	0.6776	0.6612	1.7
7	7	1, 4, 3, 5, 23, 14, 20	0.6872	0.6684	1.1
8	8	14, 1, 23, 20, 5, 3, 13, 4	0.6933	0.6721	0.6
9	9	5, 1, 6, 23, 13, 3, 4, 14, 20	0.6984	0.6748	0.4
10	10	23, 1, 3, 4, 6, 14, 22, 20, 13, 5	0.7030	0.6769	0.3
11	11	14, 1, 23, 13, 20, 4, 3, 6, 5, 22, 21	0.7085	0.6801	0.5
12	12	1, 4, 11, 23, 13, 14, 3, 20, 6, 21, 5, 8	0.7160	0.6856	0.8
13	13	1, 14, 23, 3, 4, 20, 11, 13, 5, 22, 21, 6, 8	0.7205	0.6878	0.3
14	14	1, 13, 6, 4, 23, 19, 20, 21, 3, 11, 5, 8, 14, 22	0.7237	0.6885	0.1
15	15	11, 1, 6, 3, 4, 5, 21, 23, 13, 20, 12, 19, 22, 8, 14	0.7242	0.6862	-0.3
16	16	1, 6, 4, 23, 20, 13, 11, 3, 22, 5, 14, 12, 8, 17, 21, 10	0.7265	0.6860	0.0
17	17	1, 4, 23, 11, 3, 20, 13, 14, 12, 16, 6, 5, 21, 22, 19, ...	0.7275	0.6842	-0.3
18	18	1, 11, 3, 22, 4, 13, 12, 20, 23, 21, 6, 5, 14, 8, 10, 1...	0.7298	0.6839	0.0
19	19	1, 11, 3, 22, 4, 13, 12, 20, 23, 21, 6, 5, 14, 8, 10, 1...	0.7312	0.6826	-0.2
20	20	1, 14, 23, 20, 4, 11, 5, 22, 13, 12, 21, 3, 6, 10, 16, ...	0.7323	0.6808	-0.3
21	21	1, 23, 11, 14, 4, 3, 20, 13, 6, 19, 15, 22, 12, 16, 21,...	0.7336	0.6793	-0.2
22	22	1, 11, 3, 22, 4, 13, 12, 20, 23, 21, 6, 5, 14, 8, 10, 1...	0.7350	0.6779	-0.2
23	23	1, 11, 3, 22, 4, 13, 12, 20, 23, 21, 6, 5, 14, 8, 10, 1...	0.7352	0.6749	-0.4
24	24	23, 14, 1, 4, 21, 13, 20, 3, 11, 16, 12, 22, 6, 10, 15,...	0.7359	0.6725	-0.4
25	25	1, 5, 12, 23, 3, 13, 4, 6, 16, 11, 20, 21, 14, 17, 22, ...	0.7362	0.6696	-0.4

Table 8: *Best Probable Subset of Size l – Dataset # 2*

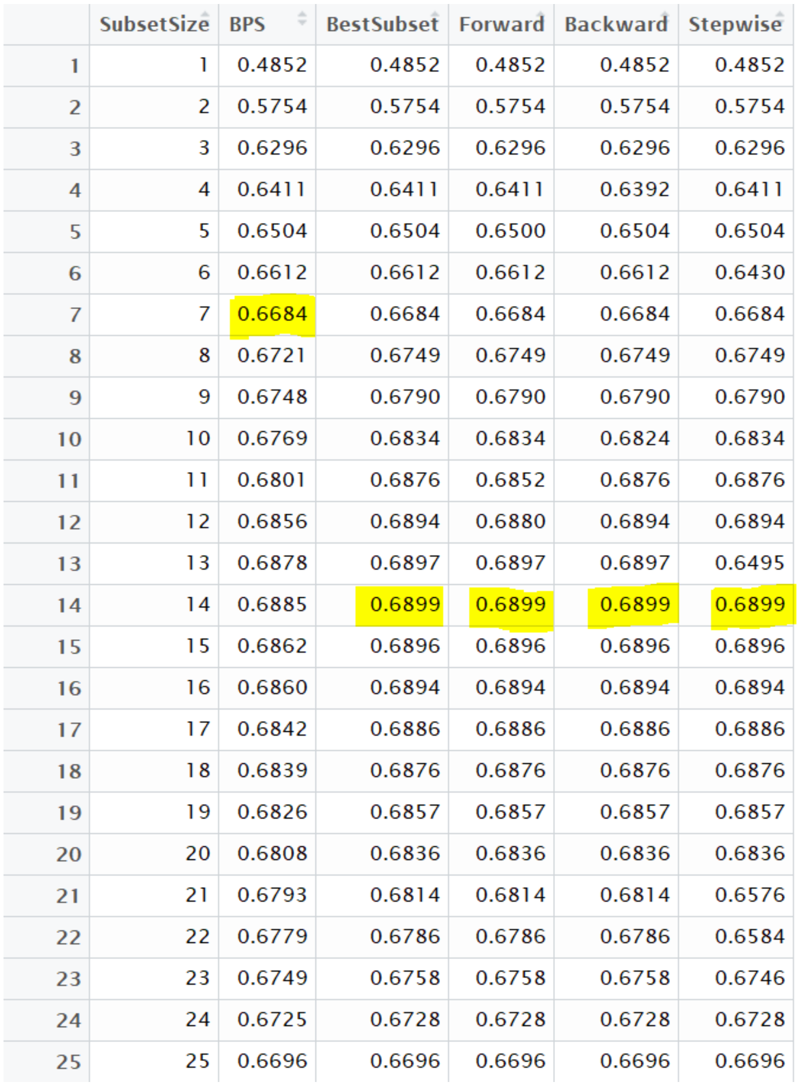
According to our stopping rule, we should select the best probable subset of seven regressors, $X_1, X_3, X_4, X_5, X_{14}, X_{20}, and\ X_{23}$, which yield the following LS estimates:



	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	5.4416323	2.15199293	2.528648	1.278205e-02
X1	0.9673448	0.07380813	13.106211	1.055851e-24
X3	0.8249247	0.15185295	5.432392	3.068362e-07
X4	1.0004758	0.24462950	4.089760	7.960643e-05
X5	0.8227310	0.36204690	2.272443	2.488585e-02
X14	-2.2941722	1.21251759	-1.892073	6.095289e-02
X20	0.1779347	0.06918388	2.571910	1.136595e-02
X23	-0.2869705	0.12446353	-2.305660	2.289116e-02

Table 9: *Best Probable Subset LS Estimates – Dataset # 2*

When we compare the fitted model to the population linear regression model, we can see that we obtained 4 out of the 5 regressor we used to generate the observations of Y, along with 3 nuisance variables as regressors. It is important to note here that not selecting every regressor having a non-zero regression coefficient is not necessarily a bad thing. It just means that the regressor did not contribute to the fitted model. To make proper conclusions, we compare the BPS method with other existing subset selection methods.



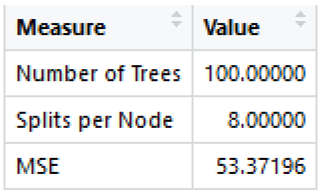
	SubsetSize	BPS	BestSubset	Forward	Backward	Stepwise
1	1	0.4852	0.4852	0.4852	0.4852	0.4852
2	2	0.5754	0.5754	0.5754	0.5754	0.5754
3	3	0.6296	0.6296	0.6296	0.6296	0.6296
4	4	0.6411	0.6411	0.6411	0.6392	0.6411
5	5	0.6504	0.6504	0.6500	0.6504	0.6504
6	6	0.6612	0.6612	0.6612	0.6612	0.6430
7	7	0.6684	0.6684	0.6684	0.6684	0.6684
8	8	0.6721	0.6749	0.6749	0.6749	0.6749
9	9	0.6748	0.6790	0.6790	0.6790	0.6790
10	10	0.6769	0.6834	0.6834	0.6824	0.6834
11	11	0.6801	0.6876	0.6852	0.6876	0.6876
12	12	0.6856	0.6894	0.6880	0.6894	0.6894
13	13	0.6878	0.6897	0.6897	0.6897	0.6495
14	14	0.6885	0.6899	0.6899	0.6899	0.6899
15	15	0.6862	0.6896	0.6896	0.6896	0.6896
16	16	0.6860	0.6894	0.6894	0.6894	0.6894
17	17	0.6842	0.6886	0.6886	0.6886	0.6886
18	18	0.6839	0.6876	0.6876	0.6876	0.6876
19	19	0.6826	0.6857	0.6857	0.6857	0.6857
20	20	0.6808	0.6836	0.6836	0.6836	0.6836
21	21	0.6793	0.6814	0.6814	0.6814	0.6576
22	22	0.6779	0.6786	0.6786	0.6786	0.6584
23	23	0.6749	0.6758	0.6758	0.6758	0.6746
24	24	0.6725	0.6728	0.6728	0.6728	0.6728
25	25	0.6696	0.6696	0.6696	0.6696	0.6696

Table 10: *Subset Selection Methods R^2_{adj} Comparisons – Dataset # 2*

Table 10 in the previous page contains information for comparing the different subset selection methods. The table provides the R^2_{adj} for each subset size using each subset selection method. The cells highlighted in yellow indicates the R^2_{adj} value for the best model selected by each method.

We can observe that the BPS method has the same R^2_{adj} values as most subset selection methods for all subset sizes. Most importantly, the R^2_{adj} values when the subset size equals 7 is the same for all subset selection methods and they all have the same regressors and LS estimates when $l = 7$, so the BPS is performing just as well and reducing the dimensions even more than other methods. According to the subset selection criteria explained in chapter 1, all pre-existing subset selection methods should choose $l = 14$ as the best subset size, which is when R^2_{adj} reaches a maximum. The BPS method does not take any more regressors once the adjusted R-Squared value increases by less than 1%, so we take $l = 7$ regressors as the best probable subset size for the best probable model M^{BPS}.

We now compare the MSE_{BPS} and MSE_{RF}. The table below provides a summary of the number of trees, number of regressors considered at each node, and MSE, for the Random Forest created for dataset # 2:



Measure	Value
Number of Trees	100.00000
Splits per Node	8.00000
MSE	53.37196

Table 11: *Random Forest Summary – Dataset # 2*

The MSE_{RF} is much higher than the MSE_{BPS}, which equals 26.1753, so we conclude that the BPS method yields lower variation than Random Forests for dataset # 2.

Dataset # 3:

Number of Predictors: 50

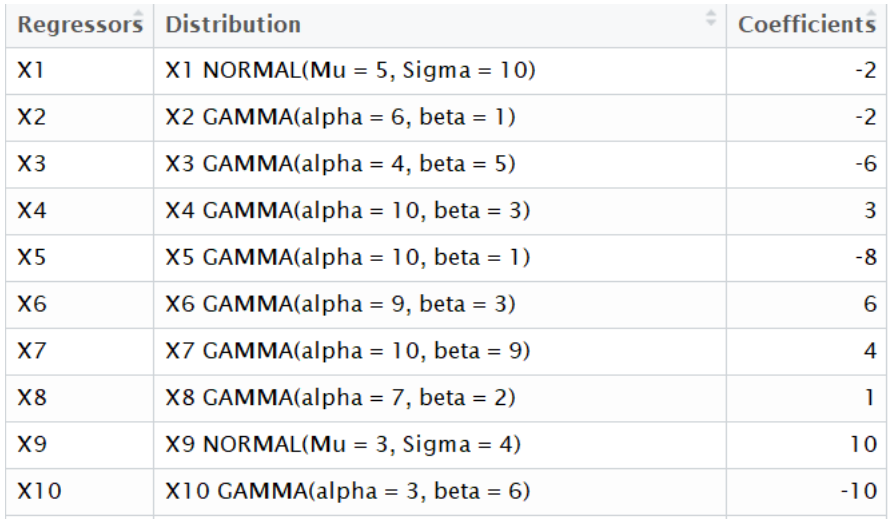
Contributing Predictors: $X_1, X_2, \dots, X_{10}$

Sample Size: 250

Error Term: $\varepsilon \sim N(0, \sigma^2 = 10)$

Maximum Pair-Wise Correlation Value: 0.205

Non-Zero Regression Coefficients:



Regressors	Distribution	Coefficients
X1	X1 NORMAL(Mu = 5, Sigma = 10)	-2
X2	X2 GAMMA(alpha = 6, beta = 1)	-2
X3	X3 GAMMA(alpha = 4, beta = 5)	-6
X4	X4 GAMMA(alpha = 10, beta = 3)	3
X5	X5 GAMMA(alpha = 10, beta = 1)	-8
X6	X6 GAMMA(alpha = 9, beta = 3)	6
X7	X7 GAMMA(alpha = 10, beta = 9)	4
X8	X8 GAMMA(alpha = 7, beta = 2)	1
X9	X9 NORMAL(Mu = 3, Sigma = 4)	10
X10	X10 GAMMA(alpha = 3, beta = 6)	-10

Table 12: *Regression Coefficients – Dataset # 3*

Just like in the two previous data set, any predictor variables that do not appear in the table above have a regression coefficient equal to zero. We generated 1000 models for each subset size i, where $1 \leq i \leq 50$, using R. From each group of 1000 models, the best model of size i, denoted as M_i^{BPS}, is selected. The following plot and table show the adjusted R-squares for each M_i^{BPS} model.

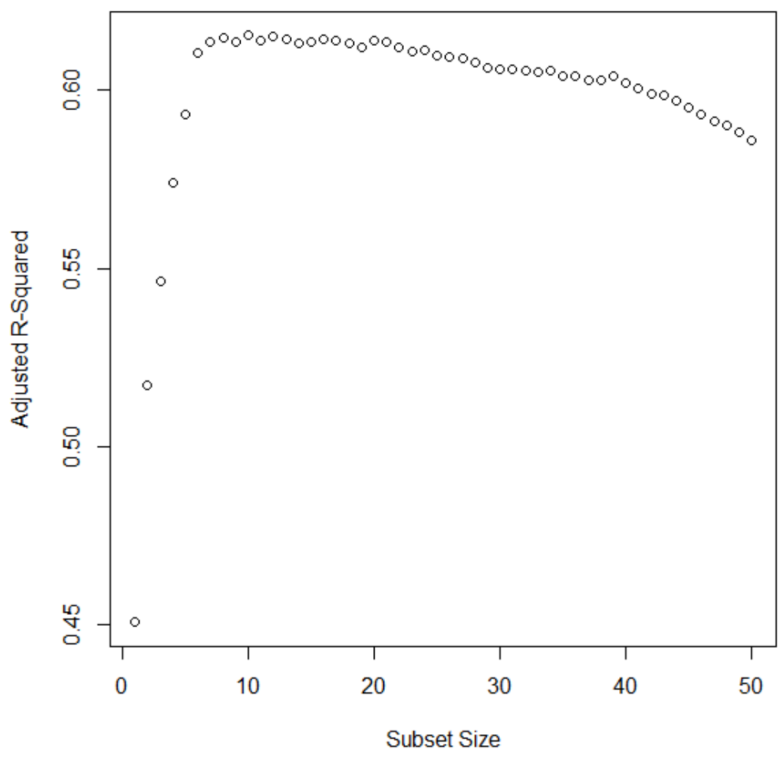
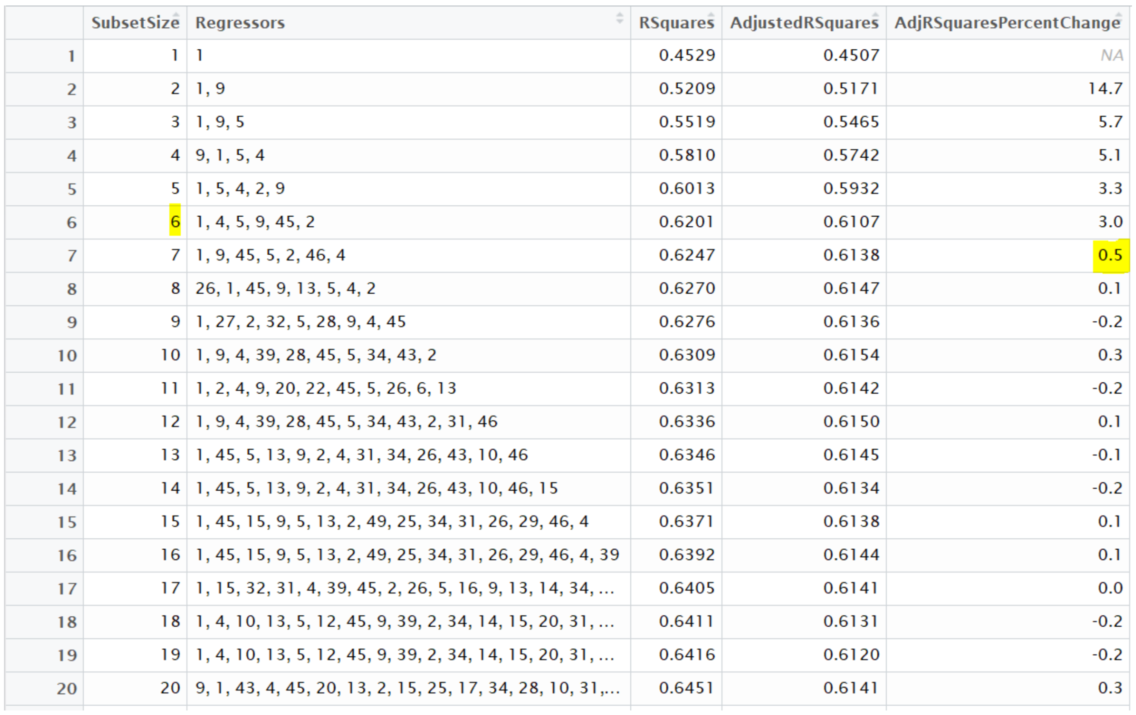


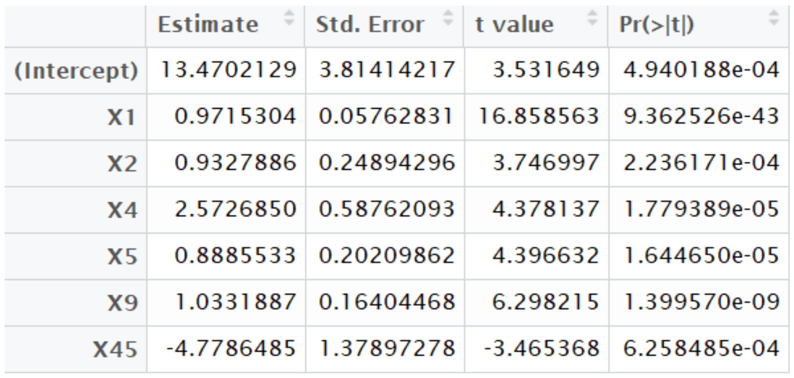
Figure 14: *Adjusted R-Squares vs. Subset Size Plot – Dataset # 3*



	SubsetSize	Regressors	RSquares	AdjustedRSquares	AdjRSquaresPercentChange
1	1	1	0.4529	0.4507	NA
2	2	1, 9	0.5209	0.5171	14.7
3	3	1, 9, 5	0.5519	0.5465	5.7
4	4	9, 1, 5, 4	0.5810	0.5742	5.1
5	5	1, 5, 4, 2, 9	0.6013	0.5932	3.3
6	6	1, 4, 5, 9, 45, 2	0.6201	0.6107	3.0
7	7	1, 9, 45, 5, 2, 46, 4	0.6247	0.6138	0.5
8	8	26, 1, 45, 9, 13, 5, 4, 2	0.6270	0.6147	0.1
9	9	1, 27, 2, 32, 5, 28, 9, 4, 45	0.6276	0.6136	-0.2
10	10	1, 9, 4, 39, 28, 45, 5, 34, 43, 2	0.6309	0.6154	0.3
11	11	1, 2, 4, 9, 20, 22, 45, 5, 26, 6, 13	0.6313	0.6142	-0.2
12	12	1, 9, 4, 39, 28, 45, 5, 34, 43, 2, 31, 46	0.6336	0.6150	0.1
13	13	1, 45, 5, 13, 9, 2, 4, 31, 34, 26, 43, 10, 46	0.6346	0.6145	-0.1
14	14	1, 45, 5, 13, 9, 2, 4, 31, 34, 26, 43, 10, 46, 15	0.6351	0.6134	-0.2
15	15	1, 45, 15, 9, 5, 13, 2, 49, 25, 34, 31, 26, 29, 46, 4	0.6371	0.6138	0.1
16	16	1, 45, 15, 9, 5, 13, 2, 49, 25, 34, 31, 26, 29, 46, 4, 39	0.6392	0.6144	0.1
17	17	1, 15, 32, 31, 4, 39, 45, 2, 26, 5, 16, 9, 13, 14, 34, ...	0.6405	0.6141	0.0
18	18	1, 4, 10, 13, 5, 12, 45, 9, 39, 2, 34, 14, 15, 20, 31, ...	0.6411	0.6131	-0.2
19	19	1, 4, 10, 13, 5, 12, 45, 9, 39, 2, 34, 14, 15, 20, 31, ...	0.6416	0.6120	-0.2
20	20	9, 1, 43, 4, 45, 20, 13, 2, 15, 25, 17, 34, 28, 10, 31,...	0.6451	0.6141	0.3

Table 13: *Best Probable Subset of Size l – Dataset # 3*

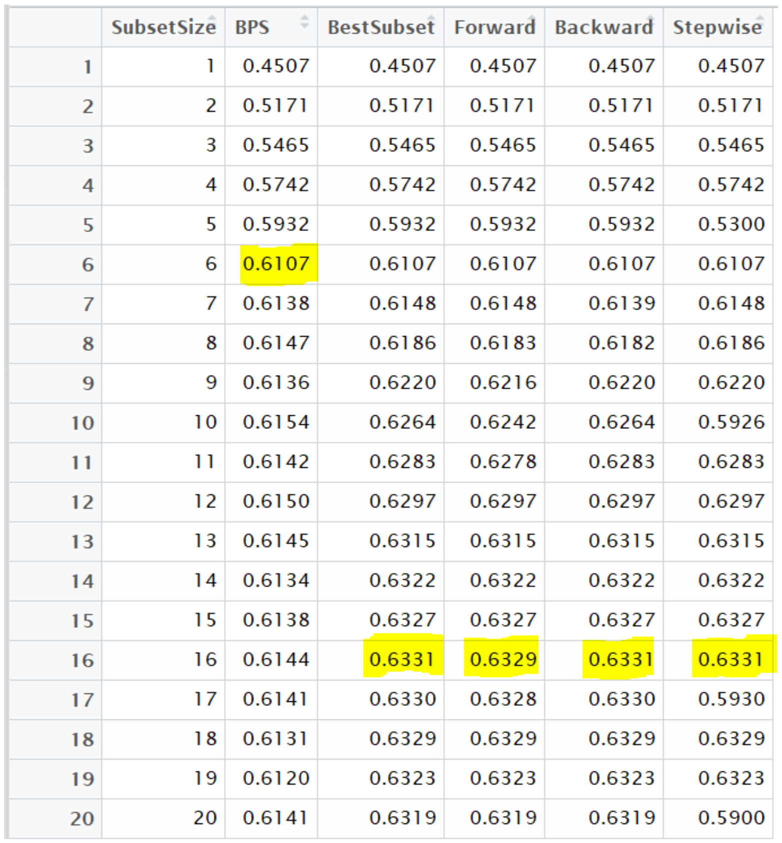
Note that the table above does not contain the entire list of best probable models for all sizes. Rather, the list contains the best probable models for subset sizes $1\ to\ 20$. The rest are omitted as the entire table takes a lot of space and we are only concerned with the first seven rows. Regardless, according to our stopping rule, we should select the best probable subset of six regressors, $X_1, X_2, X_4, X_5, X_9,\ and\ X_{45}$, which yield the following LS estimates:



	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	13.4702129	3.81414217	3.531649	4.940188e-04
X1	0.9715304	0.05762831	16.858563	9.362526e-43
X2	0.9327886	0.24894296	3.746997	2.236171e-04
X4	2.5726850	0.58762093	4.378137	1.779389e-05
X5	0.8885533	0.20209862	4.396632	1.644650e-05
X9	1.0331887	0.16404468	6.298215	1.399570e-09
X45	-4.7786485	1.37897278	-3.465368	6.258485e-04

Table 14: *Best Probable Subset LS Estimates – Dataset # 3*

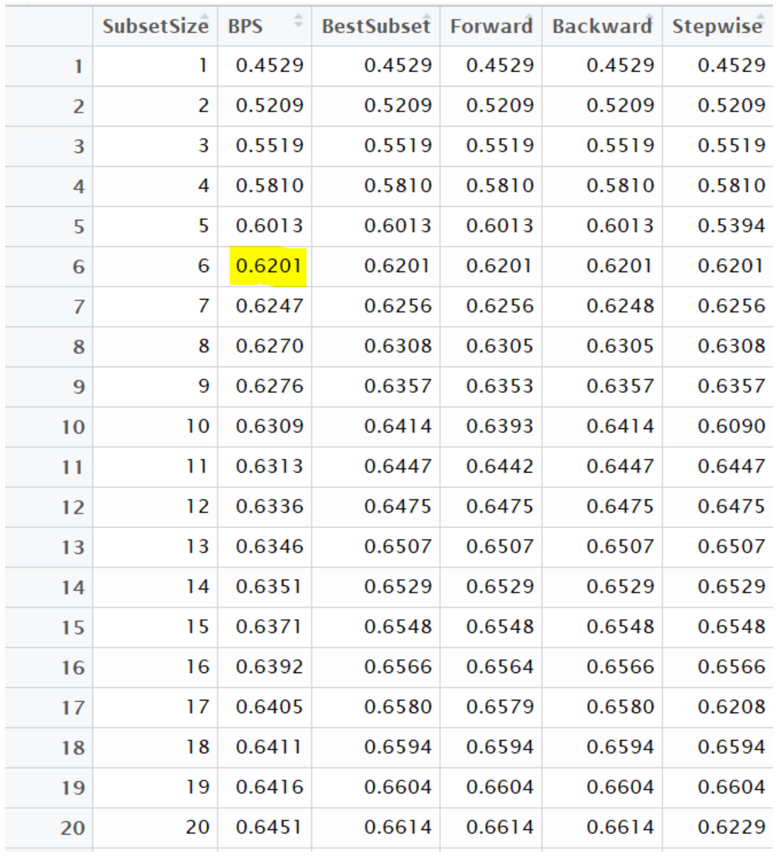
When we compare the fitted model to the population linear regression model, we can see that we obtained 5 out of the 10 regressor we used to generate the observations of Y, along with only one predictor variable (X_{45}) not in the original population model. Let us now see how the BPS method performed compared to other subset selection methods. The table in the next page contains the R^2_{adj} for each subset size selected by each subset selection method. Note from the table that only the first 20 best probable subset sizes are included. We can see that the R^2_{adj} value for a subset selection of size $l = 6$ is equal for all subset selection methods, including BPS.



	SubsetSize	BPS	BestSubset	Forward	Backward	Stepwise
1	1	0.4507	0.4507	0.4507	0.4507	0.4507
2	2	0.5171	0.5171	0.5171	0.5171	0.5171
3	3	0.5465	0.5465	0.5465	0.5465	0.5465
4	4	0.5742	0.5742	0.5742	0.5742	0.5742
5	5	0.5932	0.5932	0.5932	0.5932	0.5300
6	6	0.6107	0.6107	0.6107	0.6107	0.6107
7	7	0.6138	0.6148	0.6148	0.6139	0.6148
8	8	0.6147	0.6186	0.6183	0.6182	0.6186
9	9	0.6136	0.6220	0.6216	0.6220	0.6220
10	10	0.6154	0.6264	0.6242	0.6264	0.5926
11	11	0.6142	0.6283	0.6278	0.6283	0.6283
12	12	0.6150	0.6297	0.6297	0.6297	0.6297
13	13	0.6145	0.6315	0.6315	0.6315	0.6315
14	14	0.6134	0.6322	0.6322	0.6322	0.6322
15	15	0.6138	0.6327	0.6327	0.6327	0.6327
16	16	0.6144	0.6331	0.6329	0.6331	0.6331
17	17	0.6141	0.6330	0.6328	0.6330	0.5930
18	18	0.6131	0.6329	0.6329	0.6329	0.6329
19	19	0.6120	0.6323	0.6323	0.6323	0.6323
20	20	0.6141	0.6319	0.6319	0.6319	0.5900

Table 15: *Subset Selection Methods R^2_{adj} Comparisons – Dataset # 3*

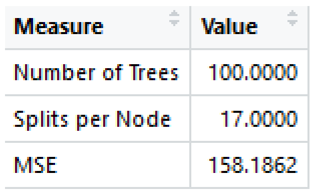
We have equal adjusted R^2s because all methods selected the same regressors for a subset of size $l = 6$. We can see from the table that the other subset selection methods also selected some predictors that had a regression parameter equal to zero in the true population model. Note also that the BPS method selected a smaller subset of regressors which yields a very similar R^2_{adj} value to the subset selected by the other subset selection methods. When we compare the BPS method to the best Best Subset Selection method, we can see that there is a relatively large difference in the number of regressors selected for the fitted model while the difference in their R^2_{adj} is very small. The fitted model using Best Subset Selection has 16 regressors and an R^2_{adj} of 0.6331 while the BPS gives us a model with only 6 regressors and an R^2_{adj} of 0.617 – a small difference in R^2_{adj} for a large difference in number of regressors, which is very useful in high dimensional data. Lastly, the reduced model we get using BPS yields an R-squared value of 0.6201, which is very close to the full regression R-squared, which is 0.6692. The table below provides all R-Squared values for each subset obtained using each subset selection method.



	SubsetSize	BPS	BestSubset	Forward	Backward	Stepwise
1	1	0.4529	0.4529	0.4529	0.4529	0.4529
2	2	0.5209	0.5209	0.5209	0.5209	0.5209
3	3	0.5519	0.5519	0.5519	0.5519	0.5519
4	4	0.5810	0.5810	0.5810	0.5810	0.5810
5	5	0.6013	0.6013	0.6013	0.6013	0.5394
6	6	0.6201	0.6201	0.6201	0.6201	0.6201
7	7	0.6247	0.6256	0.6256	0.6248	0.6256
8	8	0.6270	0.6308	0.6305	0.6305	0.6308
9	9	0.6276	0.6357	0.6353	0.6357	0.6357
10	10	0.6309	0.6414	0.6393	0.6414	0.6090
11	11	0.6313	0.6447	0.6442	0.6447	0.6447
12	12	0.6336	0.6475	0.6475	0.6475	0.6475
13	13	0.6346	0.6507	0.6507	0.6507	0.6507
14	14	0.6351	0.6529	0.6529	0.6529	0.6529
15	15	0.6371	0.6548	0.6548	0.6548	0.6548
16	16	0.6392	0.6566	0.6564	0.6566	0.6566
17	17	0.6405	0.6580	0.6579	0.6580	0.6208
18	18	0.6411	0.6594	0.6594	0.6594	0.6594
19	19	0.6416	0.6604	0.6604	0.6604	0.6604
20	20	0.6451	0.6614	0.6614	0.6614	0.6229

Table 16: *Subset Selection Methods R^2 Comparisons – Dataset # 3*

Finally, the MSE_{BPS} we obtained for dataset # 3 equals 99.6679, a much lower value than the MSE_{RF}, which is provided in the table below:



Measure	Value
Number of Trees	100.0000
Splits per Node	17.0000
MSE	158.1862

Table 17: *Random Forest Summary – Dataset # 3*

Dataset # 4:

Number of Predictors: 100

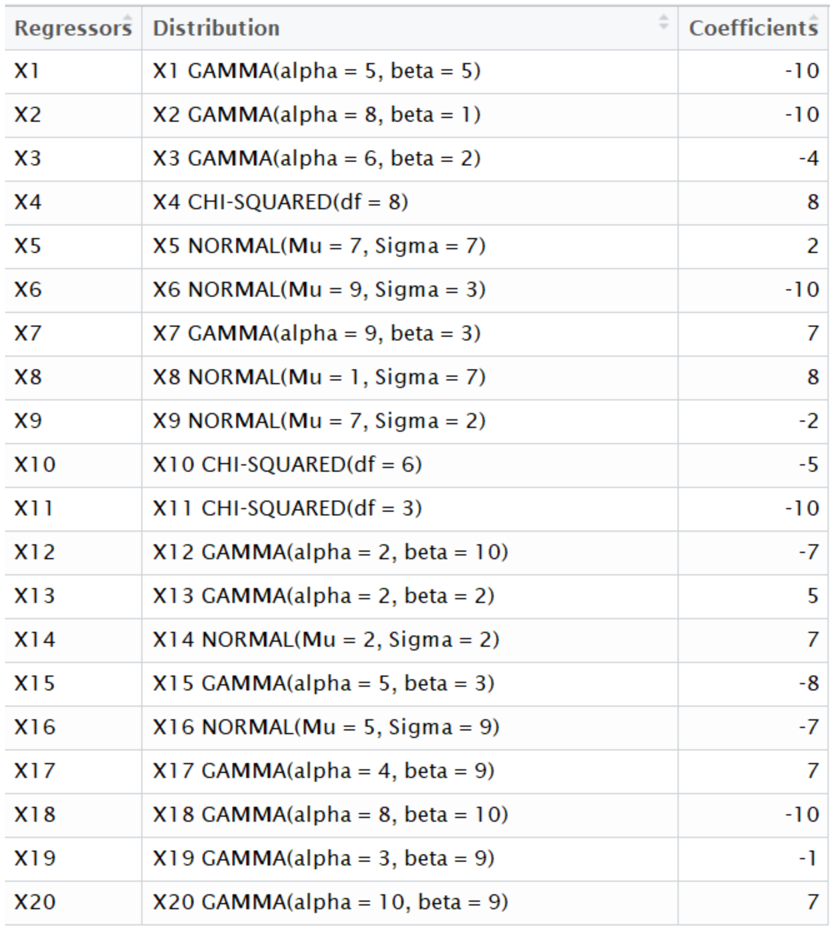
Contributing Predictors: $X_1, X_2, \dots, X_{20}$

Sample Size: 80

Error Term: $\varepsilon \sim N(0, \sigma^2 = 10)$

Maximum Pair-Wise Correlation Value: 0.440

Non-Zero Regression Coefficients:



Regressors	Distribution	Coefficients
X1	X1 GAMMA(alpha = 5, beta = 5)	-10
X2	X2 GAMMA(alpha = 8, beta = 1)	-10
X3	X3 GAMMA(alpha = 6, beta = 2)	-4
X4	X4 CHI-SQUARED(df = 8)	8
X5	X5 NORMAL(Mu = 7, Sigma = 7)	2
X6	X6 NORMAL(Mu = 9, Sigma = 3)	-10
X7	X7 GAMMA(alpha = 9, beta = 3)	7
X8	X8 NORMAL(Mu = 1, Sigma = 7)	8
X9	X9 NORMAL(Mu = 7, Sigma = 2)	-2
X10	X10 CHI-SQUARED(df = 6)	-5
X11	X11 CHI-SQUARED(df = 3)	-10
X12	X12 GAMMA(alpha = 2, beta = 10)	-7
X13	X13 GAMMA(alpha = 2, beta = 2)	5
X14	X14 NORMAL(Mu = 2, Sigma = 2)	7
X15	X15 GAMMA(alpha = 5, beta = 3)	-8
X16	X16 NORMAL(Mu = 5, Sigma = 9)	-7
X17	X17 GAMMA(alpha = 4, beta = 9)	7
X18	X18 GAMMA(alpha = 8, beta = 10)	-10
X19	X19 GAMMA(alpha = 3, beta = 9)	-1
X20	X20 GAMMA(alpha = 10, beta = 9)	7

Table 18: *Regression Coefficients – Dataset # 4*

Note that, unlike the previous datasets, dataset # 4 has high dimensionality, with a sample size of 80 and 100 predictor variables. The table above shows the predictor variables that contributed to the simulation of the response variable Y, along with their respective distributions and regression coefficients; any predictor variables that do not appear in the table above have a regression coefficient equal to zero. We generated 1000 models for each subset size i, where $1 \leq i \leq 80/5$, using R. Recall that one of our stopping rules was that the sample size n has to be at least five times the subset size, or the number of regressors in the reduced model. Since we have 80 observations, we will only look at the best probable subsets of sizes 1 through $\frac{80}{5} = 16$. The following plot and table show the adjusted R-squares for each M_i^{BPS} model.

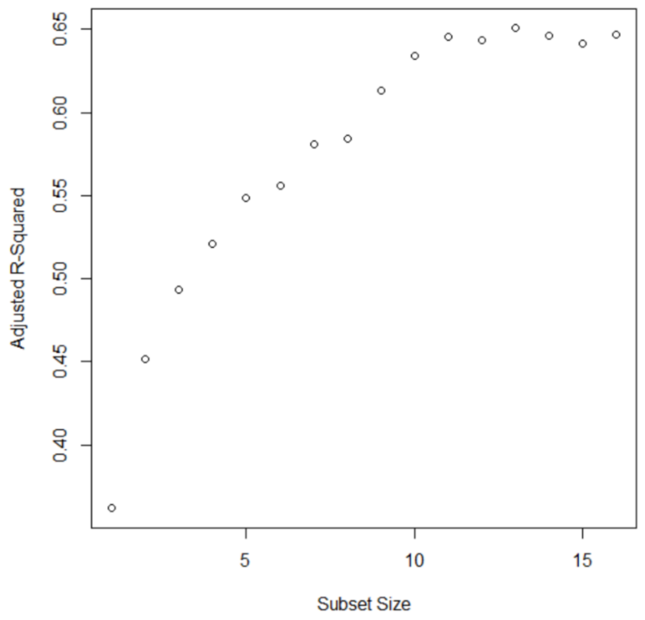
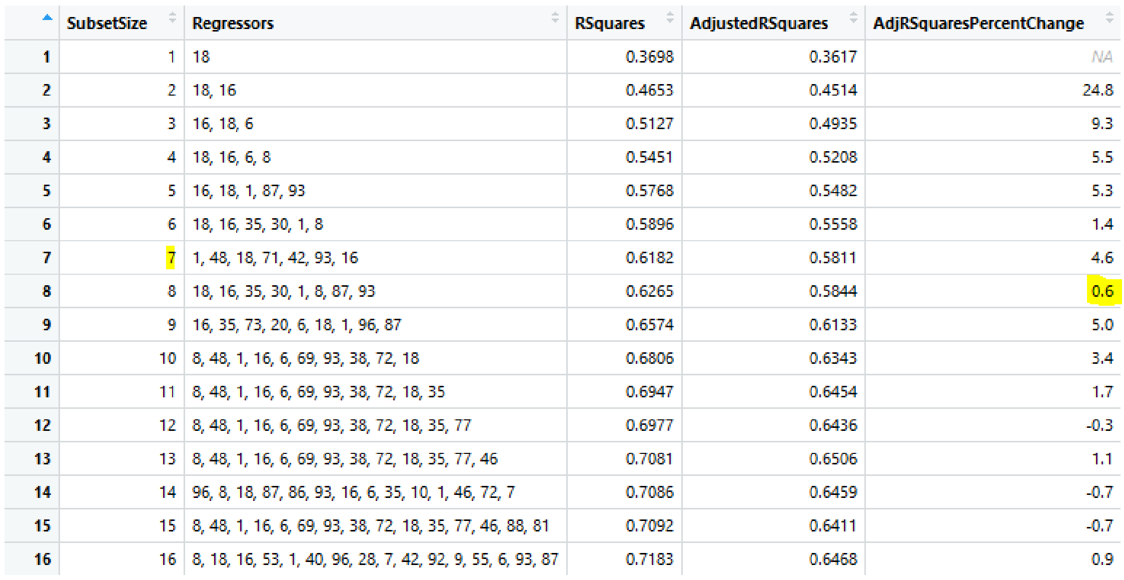


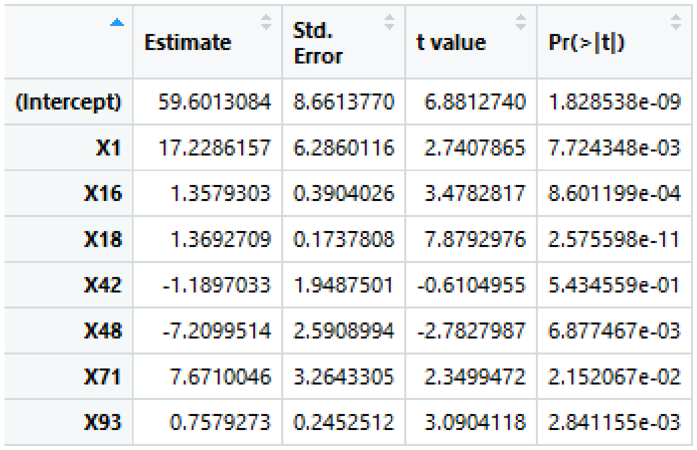
Figure 15: *Adjusted R-Squares vs. Subset Size Plot – Dataset # 4*



	SubsetSize	Regressors	RSquares	AdjustedRSquares	AdjRSquaresPercentChange
1	1	18	0.3698	0.3617	NA
2	2	18, 16	0.4653	0.4514	24.8
3	3	16, 18, 6	0.5127	0.4935	9.3
4	4	18, 16, 6, 8	0.5451	0.5208	5.5
5	5	16, 18, 1, 87, 93	0.5768	0.5482	5.3
6	6	18, 16, 35, 30, 1, 8	0.5896	0.5558	1.4
7	7	1, 48, 18, 71, 42, 93, 16	0.6182	0.5811	4.6
8	8	18, 16, 35, 30, 1, 8, 87, 93	0.6265	0.5844	0.6
9	9	16, 35, 73, 20, 6, 18, 1, 96, 87	0.6574	0.6133	5.0
10	10	8, 48, 1, 16, 6, 69, 93, 38, 72, 18	0.6806	0.6343	3.4
11	11	8, 48, 1, 16, 6, 69, 93, 38, 72, 18, 35	0.6947	0.6454	1.7
12	12	8, 48, 1, 16, 6, 69, 93, 38, 72, 18, 35, 77	0.6977	0.6436	-0.3
13	13	8, 48, 1, 16, 6, 69, 93, 38, 72, 18, 35, 77, 46	0.7081	0.6506	1.1
14	14	96, 8, 18, 87, 86, 93, 16, 6, 35, 10, 1, 46, 72, 7	0.7086	0.6459	-0.7
15	15	8, 48, 1, 16, 6, 69, 93, 38, 72, 18, 35, 77, 46, 88, 81	0.7092	0.6411	-0.7
16	16	8, 18, 16, 53, 1, 40, 96, 28, 7, 42, 92, 9, 55, 6, 93, 87	0.7183	0.6468	0.9

Table 19: *Best Probable Subset of Size l – Dataset # 4*

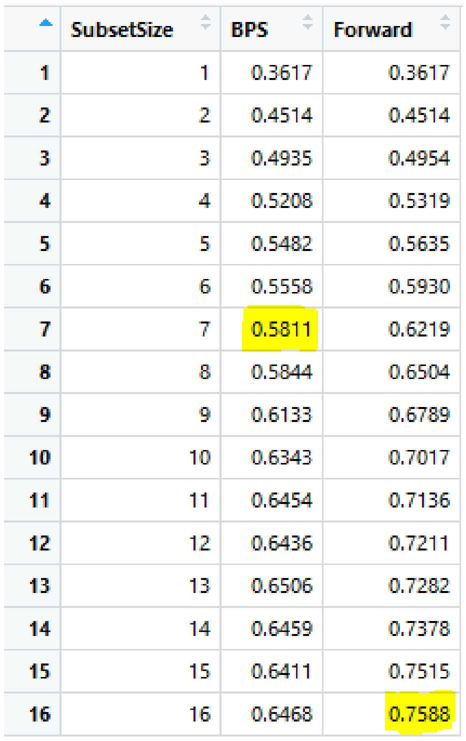
The table above contains the best probable models for subset sizes $1 \; through \; 16$. Regardless, according to our stopping rule, we should select the best probable subset of seven regressors, $X_1, X_{16}, X_{18}, X_{42}, X_{71}, \; and \; X_{93}$, which yield the following LS estimates:



	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	59.6013084	8.6613770	6.8812740	1.828538e-09
X1	17.2286157	6.2860116	2.7407865	7.724348e-03
X16	1.3579303	0.3904026	3.4782817	8.601199e-04
X18	1.3692709	0.1737808	7.8792976	2.575598e-11
X42	-1.1897033	1.9487501	-0.6104955	5.434559e-01
X48	-7.2099514	2.5908994	-2.7827987	6.877467e-03
X71	7.6710046	3.2643305	2.3499472	2.152067e-02
X93	0.7579273	0.2452512	3.0904118	2.841155e-03

Table 20: *Best Probable Subset LS Estimates – Dataset # 4*

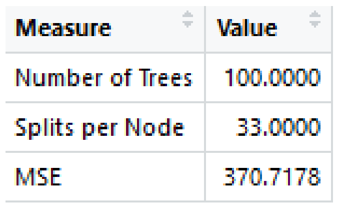
When we compare the fitted model to the population linear regression model, we can see that we obtained 3 out of the 20 regressor we used to generate the observations of Y, along with four other predictor variables. Let us now see how the BPS method performed compared to forward selection. The following table contains the R^2_{adj} for each subset size selected by each subset selection method:



	SubsetSize	BPS	Forward
1	1	0.3617	0.3617
2	2	0.4514	0.4514
3	3	0.4935	0.4954
4	4	0.5208	0.5319
5	5	0.5482	0.5635
6	6	0.5558	0.5930
7	7	0.5811	0.6219
8	8	0.5844	0.6504
9	9	0.6133	0.6789
10	10	0.6343	0.7017
11	11	0.6454	0.7136
12	12	0.6436	0.7211
13	13	0.6506	0.7282
14	14	0.6459	0.7378
15	15	0.6411	0.7515
16	16	0.6468	0.7588

Table 21: *Subset Selection Methods R^2_{adj} Comparisons – Dataset # 4*

Note from the table above that we are only comparing the BPS method with the Forward Selection method. The other methods cannot be used with data that has high dimensionality [3]. We can see that the R^2_{adj} value for a subset selection of size $l = 7$ using the BPS method is slightly lower than the subset of the same size obtained using forward selection. When comparing the BPS to random forests, we see that the MSE_{BPS}, which equals 209.369, is once again much lower than the MSE_{RF}, provided in the table below.



Measure	Value
Number of Trees	100.0000
Splits per Node	33.0000
MSE	370.7178

Table 22: *Random Forest Summary – Dataset # 4*

Dataset # 5: College Dataset

Number of Predictors: 16

Sample Size: 777

Important Information:

The College dataset selected for this section was taken from the ISLR package in R. The dataset contains a total of 18 variables. From these, we have selected the Graduation Rates, *Grad.Rate*, as the response variable. We also dropped out the *Private* variable because it is non-numeric, and it would interfere with the correlation matrix. The remaining 16 variables were used as predictor variables. We now use various methods to select a subset of the predictor variables to fit a model for the graduation rates, which we have renamed as Y in the dataset.

Model Assumptions:

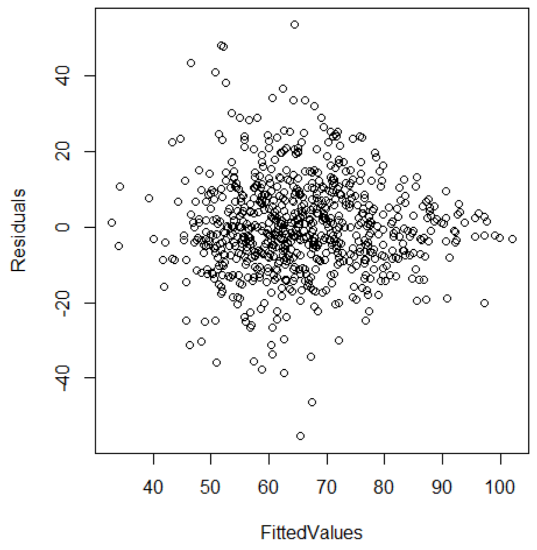


Figure 16: *Residuals vs. Fitted Values – Dataset # 5*

The Fitted Values vs. Residuals plot shown above shows no pattern of the data, so we can assume that the response variable is linear and the error terms have a constant variance.

The correlation matrix, however, shows very high values amongst many pairs of regressors; the correlation values go as high as 0.943 between the variables *Apps* and *Accept*, which are the number of applications received and number of applications accepted, respectively. Regardless of the collinearity, we continue by selecting the best probable model using the BPS method.

Note that the sample size is very large compared to the number of predictor variables. Since we do not know the true model for the College dataset, we use cross validation. 80% of the observations were randomly selected for the train set and the remaining 20% of the observations were used for the test set. The "random" selection of observations was done in R, using $set.seed(1)$ and $sample()$ to generate the random indexes for the observations that belong to the train data set. We proceed with multiple subset selection methods to find out which subset of predictors can contribute to the fitted model. The following plot and table show the adjusted R-squares for each M_i^{BPS} model.

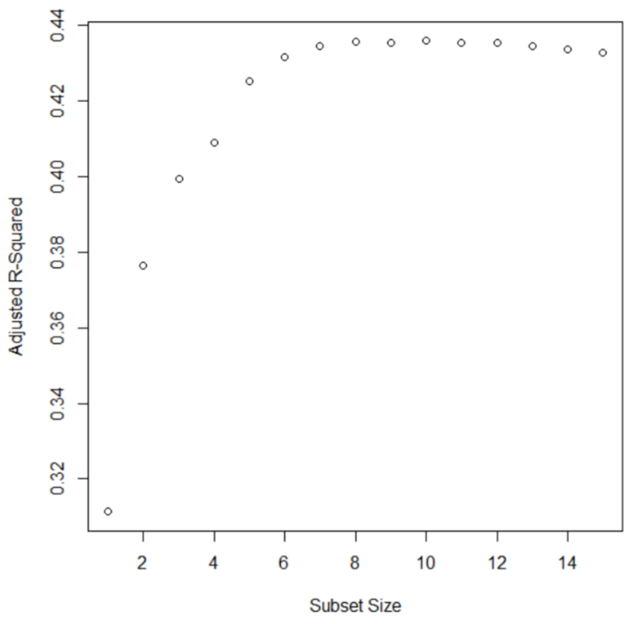
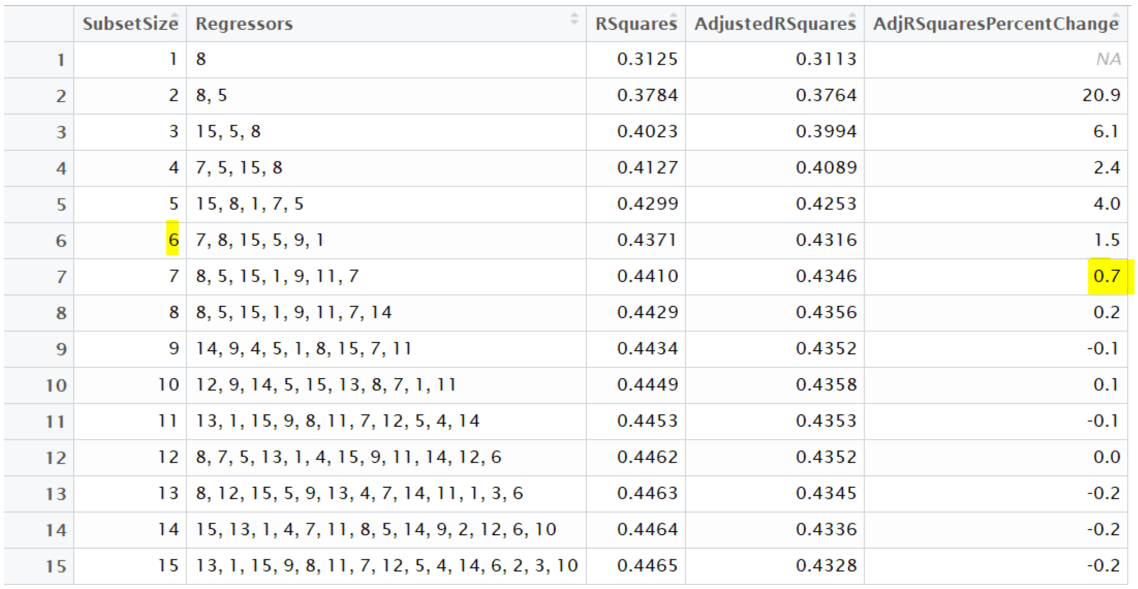


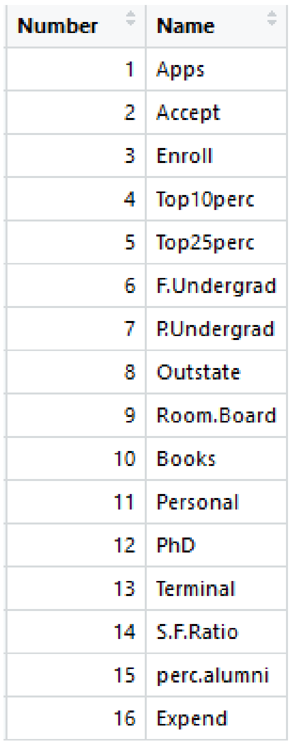
Figure 17: *Adjusted R-Squares vs. Subset Size Plot – Dataset # 5*



	SubsetSize	Regressors	RSquares	AdjustedRSquares	AdjRSquaresPercentChange
1	1	8	0.3125	0.3113	NA
2	2	8, 5	0.3784	0.3764	20.9
3	3	15, 5, 8	0.4023	0.3994	6.1
4	4	7, 5, 15, 8	0.4127	0.4089	2.4
5	5	15, 8, 1, 7, 5	0.4299	0.4253	4.0
6	6	7, 8, 15, 5, 9, 1	0.4371	0.4316	1.5
7	7	8, 5, 15, 1, 9, 11, 7	0.4410	0.4346	0.7
8	8	8, 5, 15, 1, 9, 11, 7, 14	0.4429	0.4356	0.2
9	9	14, 9, 4, 5, 1, 8, 15, 7, 11	0.4434	0.4352	-0.1
10	10	12, 9, 14, 5, 15, 13, 8, 7, 1, 11	0.4449	0.4358	0.1
11	11	13, 1, 15, 9, 8, 11, 7, 12, 5, 4, 14	0.4453	0.4353	-0.1
12	12	8, 7, 5, 13, 1, 4, 15, 9, 11, 14, 12, 6	0.4462	0.4352	0.0
13	13	8, 12, 15, 5, 9, 13, 4, 7, 14, 11, 1, 3, 6	0.4463	0.4345	-0.2
14	14	15, 13, 1, 4, 7, 11, 8, 5, 14, 9, 2, 12, 6, 10	0.4464	0.4336	-0.2
15	15	13, 1, 15, 9, 8, 11, 7, 12, 5, 4, 14, 6, 2, 3, 10	0.4465	0.4328	-0.2

Table 23: *Best Probable Subset of Size l – Dataset # 5*

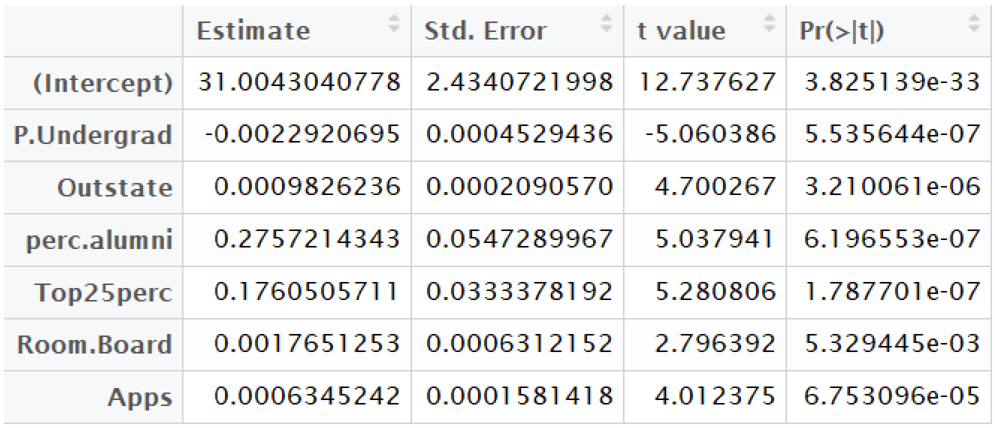
Note that the regressors have a number rather than a name. The table below matches each predictor number with its name:



Number	Name
1	Apps
2	Accept
3	Enroll
4	Top10perc
5	Top25perc
6	F.Undergrad
7	P.Undergrad
8	Outstate
9	Room.Board
10	Books
11	Personal
12	PhD
13	Terminal
14	S.F.Ratio
15	perc.alumni
16	Expend

Table 24: *Regressor Numbers and Names – Dataset # 5*

According to the BPS method, the best probable subset has seven predictors: *Apps*, *Top25perc*, *P.Undergrad*, *Outstate*, *Room.Board*, *perc.alumni*, and *Expend*. These seven predictors yield an R-squared of 0.4482, almost as large as the full model's R-Squared, which is 0.4587. The reduced model has the following LS estimates:



	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	31.0043040778	2.4340721998	12.737627	3.825139e-33
P.Undergrad	-0.0022920695	0.0004529436	-5.060386	5.535644e-07
Outstate	0.0009826236	0.0002090570	4.700267	3.210061e-06
perc.alumni	0.2757214343	0.0547289967	5.037941	6.196553e-07
Top25perc	0.1760505711	0.0333378192	5.280806	1.787701e-07
Room.Board	0.0017651253	0.0006312152	2.796392	5.329445e-03
Apps	0.0006345242	0.0001581418	4.012375	6.753096e-05

Table 25: *Best Probable Subset LS Estimates – Dataset # 5*

Note that from the two regressors that had the highest pair-wise correlation (*Apps* and *Accept*), only *Apps* made it into the reduced model. The table in the next page contains the R^2_{adj} for each subset size selected by each subset selection method.

Note that the BPS selected the smallest subset out of all the subset selection methods, with a slightly smaller R^2_{adj}. We can see that the R^2_{adj} value for a subset selection of size $l = 7$ is the same for all methods. This is because all methods chose the same predictor variables when the subset size equals 7.

SubsetSize	BPS	BestSubset	Forward	Backward	Stepwise
1	0.3113	0.3113	0.3113	0.3113	0.3113
2	0.3764	0.3764	0.3764	0.3764	0.3764
3	0.3994	0.3994	0.3994	0.3994	0.3994
4	0.4089	0.4089	0.4089	0.4089	0.4089
5	0.4253	0.4253	0.4253	0.4253	0.2968
6	0.4316	0.4316	0.4316	0.4316	0.4316
7	0.4346	0.4346	0.4346	0.4346	0.4346
8	0.4356	0.4356	0.4356	0.4356	0.4356
9	0.4352	0.4357	0.4357	0.4352	0.4357
10	0.4358	0.4358	0.4354	0.4358	0.4354
11	0.4353	0.4357	0.4348	0.4357	0.4348
12	0.4352	0.4352	0.4341	0.4352	0.4352
13	0.4345	0.4345	0.4345	0.4345	0.4345
14	0.4336	0.4337	0.4337	0.4337	0.4131
15	0.4328	0.4328	0.4328	0.4328	0.4328

Table 26: *Subset Selection Methods R^2_{adj} Comparisons – Dataset # 5*

When comparing the BPS to random forests, we see that the MSE_{BPS}, which equals 173.3057, is somewhat lower than the MSE_{RF}, provided in the table below.

Measure	Value
Number of Trees	100.0000
Splits per Node	5.0000
MSE	178.4219

Table 27: *Random Forest Summary – Dataset # 5*

Note that both MSE_{BPS} and MSE_{RF} are calculated using the predictions from the test data set using cross validation.

Here we conclude the analysis and comparison of the Best Probable Subset and other subset selection methods with simulated or pre-existing datasets.

CHAPTER V

CONCLUSIONS AND REMARKS

In this paper, we have discussed some concepts in Linear Regression, including the Least Squares Estimators (LSE), model assumptions, measures of fit, and subset selection. We also discussed some of the predominant methods used in linear regression for subset selection and the concept of regression trees and random forests, some of which can be used when dealing with high dimensional data.

We continued our discussion by proposing a new subset selection method, which we called *Best Probable Subset (BPS)*, that seems to have great potential based on our simulations and can be used when we are dealing with high dimensional data. The *Best Probable Subset (BPS)* method is summarized here:

1. Create a probability pool, called *Linear Relationship Pool (LRP)*, for selecting a subset of regressors by using the following formula for each regressor of the k regressors.

   $$p_j = P(X_j) = \frac{r_j^2}{\sum_{i=1}^{k} r_i^2}, 1 \le j \le k$$

   Where p_j is the probability of choosing the jth predictor for the reduced model and r_j^2represents the squared correlation between the jth predictor and the response variable.

2. Begin by finding the null model M_0^{BPS}. For $1 \le i \le \min(k, \frac{n}{5})$:

   a. Select a subset of size i without replacement from the predictor by using the LRP.

   b. Continue finding the model with the best possible subset size i until M_i^{BPS} yields an ${R^2}_{adj}$ than is less than 1% greater than the ${R^2}_{adj}$ of M_{i-1}^{BPS}.

3. Since the adjusted R-squared continues to increase by more than 1% until M_{i-1}^{BPS}, the best probable model is $M^{BPS} = M_{i-1}^{BPS}$.

We used the BPS method on multiple simulated dataset and obtained promising results when compared to Best Subset Selection and other subset selection methods by keeping up with an equal R_{adj}^2 in the 1st, 2nd, 3rd, and 5th datasets and a slightly lower R_{adj}^2 on the 4th dataset. The method was also compared with random forests and outperformed it with a lower MSE in every single dataset.

For future research projects, we would like to do further research on the theoretical side of the BPS method, which seems to have potential with low and high dimensional data. Without showing the theoretical proof, we can guarantee that the best probable subset will choose the best subset of size l as the number of model selection repetitions (1000 in this paper) increase. A further research on this subset could include this proof. We would also like to investigate subset selection based on some other alternative subset selection probability rules. A way to improve this study would be to test the method with many more datasets of different sizes and properties and see how it performs compared to other subset selection methods. Furthermore, for 1000 models with a subset size l, we could select the model that yields the lowest MSE instead of the highest R^2. More research can be done on the performance of the method overall by finding the average proportion of the R^2 of the full model that is explained by the R^2 of the best probable model throughout many datasets.

REFERENCES

1. Montgomery, Douglas C., Elizabeth A. Peck, and Geoffrey Vining. *Introduction to Linear Regression Analysis*. 5th ed. Hoboken, NJ: Wiley, 2012.

2. Mandenhall, William. *Mathematical Statistics with Applications*. 7th ed. Boston, MA: Thomson Brooks/Cole, 2008.

3. James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*. 1st ed. New York, NY: Springer, 2013.

4. Hocking, Ronald R. *Methods and Applications of Linear Models: Regression and the Analysis of Variance*. 2nd ed. Hoboken, NJ: John Wiley and Sons, 2003.

5. Rohatgi, Vijay K. *An Introduction to Probability and Statistics*. 2nd ed. New York, NY: John Wiley and Sons, 2001.

6. Walpole, Ronald E. *Probability and Statistics for Engineers and Scientists*. 6th ed. New Jersey, NJ: Prentice-Hall, 2002.

7. Lyman, R. Ott. *An Introduction to Statistical Methods and Data Analysis*. 6th ed. Bermont, CA: Cengage, 2008.

8. Rawlings, John O. *Applied Regression Analysis: A Research Tool*. 1st ed. New York, NY: Springer, 1988.

9. Kotu, Vijay. *Predictive Analytics and Data Mining*. Waltham, MA: Elsevier, 2015.

10. Hastie, Trevor. *The Elements of Statistical Learning*. 2nd ed. Stanford, CA: Springer, 2016.

11. James, Gareth, Daniela Witten, Trevor Hastie, and Rob Tibshirani. "Package 'ISLR'." R-Project. October 19, 2017. https://cran.r-project.org/web/packages/ISLR/ISLR.pdf.

12. Strobl, Carolin, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. "Conditional Variable Importance for Random Forests." *BMC Bioinformatics*9, no. 1 (2008). doi:10.1186/1471-2105-9-307.

13. Genuer, Robin, Jean-Michel Poggi, and Christine Tuleau-Malot. "Variable Selection Using Random Forests." *Pattern Recognition Letters*31, no. 14 (2010): 2225-236. doi:10.1016/j.patrec.2010.03.014.

APPENDIX

APPENDIX A

Program R-Code

The following code can be copied and pasted into R-Studio. This code will simulate data and do all the analysis explained throughout the entirety of this paper. To avoid running errors caused by different versions, it is advised to use Microsoft R Open 3.5.1 and R-Studio 1.0.143, which are the versions I used when writing the program. Without further ado, here is the R-Code:

```
####START OF SCRIPT####--------------------------------------------------------------------

######MASTER'S THESIS - ELIESER NODARSE######
####THE PURPOSE OF THIS PROGRAM IS TO ANALYZE THE IMPACT OF
####DATA REDUCTION IN LINEAR REGRESSION

####INSTALL NECESSARY PACKAGES####

   install.packages(c("dplyr",
   "stringr",
   "DT",
   "MASS",
   "gtools",
   "ISLR",
   "leaps",
   "glmnet",
   "quantmod"
  ))

library(dplyr)
library(stringr)
library(DT)
library(MASS)
library(gtools)
library(ISLR)
library(leaps)
library(glmnet)
library(quantmod)
```

```
####CREATING THE DATA SET####
 ###CREATING A POOL OF DISTRIBUTIONS###
   #Here we create a pool of distributions which will be chosen randomly (with
   #replacement) for the GLM analysis#
        Var_Size <- 100
        Sample_Size <- 80
        Dist_Pool <- c("CHI_SQUARED", "GAMMA", "NORMAL")

 ###PREDICTOR VARIABLES###
   #Creates independent variables from a pool of possible distributions
   #Refer to Dist_Pool for all possible distributions

        Y <- rep(0, Sample_Size)
        FULL_REGRESSION_RSQUARES <- c()
        DATA_COMPLETE <- data.frame(Y, matrix(rep(rep(0,Sample_Size),Var_Size),
        ncol = Var_Size))
        Regressor_Names <- c()

    ##CREATING THE SIGNIFICANT PREDICTOR VARIABLES##

        i <- 2 #Index of complete data frame
        Significant <- round(Var_Size*0.20)

        while(i <= Significant + 1) { #We only want the first 20% of the variables#

           temp_Dist <- sample(Dist_Pool, 1, T) #Chooses the distribution of predictor
           variable Xi at random from the distribution pool

           if(temp_Dist == "CHI_SQUARED") {

                 df <- sample(1:10,1,T) #Generates data for Chi-Squared distribution
                 DATA_COMPLETE[,i] <- rchisq(Sample_Size, df)
                 #Names respective column in full data frame
                 Regressor_Names <- c(Regressor_Names, paste("X", i - 1, " CHI-
                 SQUARED(df = ", df, ")", sep = ""))

           } else if(temp_Dist == "GAMMA") {

                 alpha <- sample(1:10,1,T)
                 beta <- sample(1:10,1,T)
                 #Generates data for Gamma distribution
                 DATA_COMPLETE[,i] <- rgamma(Sample_Size, alpha, beta)
                 #Names respective column in full data frame
```

```
                Regressor_Names <- c(Regressor_Names, paste("X", i - 1, "
                GAMMA(alpha = ", alpha, ", beta = ", beta, ")", sep = ""))


        } else if(temp_Dist == "NORMAL") {

                MU <- sample(1:10,1,T)
                sigma <- sample(1:10,1,T)
                #Generates data for Normal distribution
                DATA_COMPLETE[,i] <- rnorm(Sample_Size, MU, sigma)
                #Names respective column in full data frame
                Regressor_Names <- c(Regressor_Names, paste("X", i - 1, "
                NORMAL(Mu = ", MU, ", Sigma = ", sigma, ")", sep = ""))
        }
        #Goes to the next column in data frame to generate another independent variable
        i <- i + 1
}

####RESPONSE VARIABLE####
  #Creates a dependent variable Y with n observations
  #Y is a function linear function of the significant regressors plus some random error#
  #The number of observations is pre-determined#

  i <- 2
  Y <- rep(0, Sample_Size)
  Coefficients <- c()

  while(i <= Significant + 1){

      CurrentCoeff <- sample(c(-10:-1, 1:10), 1)
      #Saves the REAL model coefifcients
      Coefficients <- c(Coefficients, CurrentCoeff)

      Y <- Y + DATA_COMPLETE[,i]
      i <- i + 1
  }

  MU <- 0
  sigma <- round(Var_Size*0.20)
  Error <- rnorm(Sample_Size, MU, sigma)

  Y <- Y + Error

  DATA_COMPLETE[,1] <- Y
```

```
##CREATING THE INSIGNIFICANT PREDICTOR VARIABLES##

   Dist_Pool <- c("POISSON", "BINOMIAL", "CHI_SQUARED",
   "EXPONENTIAL", "GAMMA", "NORMAL")
   i <- round(Var_Size*0.20) + 2 #Index of complete data frame
   Significant <- round(Var_Size*0.20)

   while(i <= ncol(DATA_COMPLETE)) {
   #We only want the first 20% of the variables#

   temp_Dist <- sample(Dist_Pool, 1, T)
   #Chooses the distribution of predictor variable Xi at random from the
   #distribution pool

   if (temp_Dist == "UNIFORM") {

           #Generates data for Uniform distribution
           DATA_COMPLETE[,i] <- runif(Sample_Size)
           #Names respective column in full data frame
           Regressor_Names <- c(Regressor_Names, paste("X", i - 1, " UNIFORM",
           sep = ""))

   } else if(temp_Dist == "POISSON") {

           lambda <- sample(1:10,1,T)
           #Generates data for Poisson distribution
           DATA_COMPLETE[,i] <- rpois(Sample_Size, lambda = lambda)
           #Names respective column in full data frame
           Regressor_Names <- c(Regressor_Names, paste("X", i - 1, "
           POISSON(lambda = ", lambda, ")", sep = ""))


   } else if(temp_Dist == "BINOMIAL") {

           n <- sample(1:10,1,T)
           p <- runif(1)
           #Generates data for Binomial distribution
           DATA_COMPLETE[,i] <- rbinom(Sample_Size, n, p)
           #Names respective column in full data frame
           Regressor_Names <- c(Regressor_Names, paste("X", i - 1, "
           BINOMIAL(n = ", n, ", p = ", p, ")", sep = ""))


   } else if(temp_Dist == "BETA") {
```

```
            alpha <- sample(1:10,1,T)
            beta <- sample(1:10,1,T)
            #Generates data for Beta distribution
            DATA_COMPLETE[,i] <- rbeta(Sample_Size, alpha, beta)
            #Names respective column in full data frame
            Regressor_Names <- c(Regressor_Names, paste("X", i - 1, " BETA(alpha
            = ", alpha, ", beta = ", beta, ")", sep = ""))


      } else if(temp_Dist == "CHI_SQUARED") {

            df <- sample(1:10,1,T)
            #Generates data for Chi-Squared distribution
            DATA_COMPLETE[,i] <- rchisq(Sample_Size, df)
            #Names respective column in full data frame
            Regressor_Names <- c(Regressor_Names, paste("X", i - 1, " CHI-
            SQUARED(df = ", df, ")", sep = ""))


      } else if(temp_Dist == "EXPONENTIAL") {

            beta <- sample(1:10,1,T)
            #Generates data for Exponential dsitribution
            DATA_COMPLETE[,i] <- rexp(Sample_Size, beta)
            #Names respective column in full data frame
            Regressor_Names <- c(Regressor_Names, paste("X", i - 1, "
            EXPONENTIAL(beta = ", beta, ")", sep = ""))


      } else if(temp_Dist == "GAMMA") {

            alpha <- sample(1:10,1,T)
            beta <- sample(1:10,1,T)
            #Generates data for Gamma distribution
            DATA_COMPLETE[,i] <- rgamma(Sample_Size, alpha, beta)
            #Names respective column in full data frame
            Regressor_Names <- c(Regressor_Names, paste("X", i - 1, "
            GAMMA(alpha = ", alpha, ", beta = ", beta, ")", sep = ""))


      } else if(temp_Dist == "NORMAL") {

            MU <- sample(1:10,1,T)
            sigma <- sample(1:10,1,T)
            #Generates data for Normal distribution
```

```
                DATA_COMPLETE[,i] <- rnorm(Sample_Size, MU, sigma)

                #Names respective column in full data frame
                Regressor_Names <- c(Regressor_Names, paste("X", i - 1, "
                NORMAL(Mu = ", MU, ", Sigma = ", sigma, ")", sep = ""))


         }

         Coefficients <- c(Coefficients, 0)
         #Goes to the next column in data frame to generate another independent variable
         i <- i + 1
   }

   NAMES <- data.frame()
   NAMES <- data.frame(Regressors = names(DATA_COMPLETE[,-1]), Distribution =
   Regressor_Names, Coefficients = Coefficients)

   Corr_Matrix <- data.frame(round(cor(DATA_COMPLETE[,-1]), 3))

####R-SQUARES OF FULL REGRESSION####
  #The R-Squared for the FULL regression
  #Full regression includes ALL independent variables

   FULL_REG_RSQUARED <- summary(lm(Y ~ ., data =
   DATA_COMPLETE))$r.squared #Regression on ALL predictor variables


    ##INDIVIDUAL REGRESSION WITH EACH PREDICTOR VARIABLE##
    #Obtains the individual R-Squared value for each independent variable when
    #regressed with the dependent variable

    INDIVIDUAL_RSQUARES <- c() #Creates an empty list of values for the individual
    R-Squares

    index <- 1 #Index for the independent variable

    #Obtains each individual R-Squared value and passes it to the list of individual R-
    Squares
    while(index <= Var_Size){

         tempRegression <- lm(Y ~ DATA_COMPLETE[,index + 1], data =
         DATA_COMPLETE) #Creates temporary regression Y ~ Xi
```

```
          INDIVIDUAL_RSQUARES <- c(INDIVIDUAL_RSQUARES,
          summary(tempRegression)$r.squared)

          index <- index + 1
  }

####CREATING A POOL OF PROBABILITIES AND A POOL OF NAMES####
  #Creates a probability pool based on the individual R-Squares of each regressor
  #The regressor with the highest individual R-Squared value has the highest number of
  #appearances in the probability pool
  LRP <- INDIVIDUAL_RSQUARES/sum(INDIVIDUAL_RSQUARES)

  #Rounds the individual R-Sqaures
  INDIVIDUAL_RSQUARES <- round(INDIVIDUAL_RSQUARES, 9)
  #Sum of all individual R-Squares
  INDIVIDUAL_RSQUARES_SUM <- sum(INDIVIDUAL_RSQUARES)
  PROBABILITIES <- INDIVIDUAL_RSQUARES/INDIVIDUAL_RSQUARES_SUM
  PROBABILITY_POOL <- c() #Creates an empty pool of probabilities

  maxIndex <- Var_Size #Index for last regressor in the data
  #Obtains the name for each gregressor
  NAMES <- names(DATA_COMPLETE[,2:(maxIndex + 1)])
  index <- 1 #Index for first regressor in the data

  #Assigns a probability to each regressor
  while(index <= maxIndex){
    #Maximum number of zeros between the decimal point and the first integer after the
    #decimal point in the smallest marginal R-Squared#
    m <- attr(regexpr("(?<=\\.)0+", format(min(INDIVIDUAL_RSQUARES), scientific =
    FALSE), perl = TRUE), "match.length")

    PROBABILITY_POOL <- c(PROBABILITY_POOL, rep(index,
    round(PROBABILITIES[index] * 10^(m+1)))) #Adds probability of current regressor
    to the probability pool

    index <- index + 1 #Goes to the next regressor

  }

  PROBABILITY_POOL


####CREATING A RANDOM POOL OF EQUAL PROBABILITIES####
  RANDOM_PROBABILITY_POOL <- seq(1:Var_Size)
  RANDOM_PROBABILITY_POOL


####LRP SELECTION####
  RUNS <- 1000 #Total number of times we will select regressors based on the created
probability pool
  CURRENT_RUN <- 1 #start on first run
  REGRESSORS_INFO <- data.frame()

  #The following process will be repeated for each run
  while(CURRENT_RUN <= RUNS){

    Prob <- PROBABILITY_POOL #We need the probability pool for each run

    ##Choosing Regressors - Probability Pool##
    #Chooses regressors using the probability pool created earlier#

    Ordered_Indexes_Sel <- c() #Regressor Indexes ordered by selection
    Ordered_Names_Sel <- c() #Regressor Names ordered by selection
    Selected_Index <- c()
    Selected_Indexes <- c()

    index <- 1

    limit <- min(Var_Size, Sample_Size/5)
    while(index <= limit){
      #Using Probability Pool#
      picked <- sample(Prob,1,T) #The index of the regressor that was picked at random

      Prob <- Prob[!Prob %in% picked] #Removes the picked index from the Probability
pool

      Ordered_Indexes_Sel <- c(Ordered_Indexes_Sel, picked) #Gets the index of the
picked regressor

      ##UPDATES INDEXES SELECTED UP TO THAT POINT##
      if(length(Selected_Index) == 0){

        Selected_Index <- c(as.character(picked))

      }else{

        Selected_Index <- paste0(Selected_Index, ", ", picked)

      }
  Selected_Indexes <- c(Selected_Indexes, Selected_Index)

        #Gets the name of the picked regressor
        Ordered_Names_Sel <- c(Ordered_Names_Sel,
        colnames(DATA_COMPLETE)[picked + 1])

        index = index + 1 #Goes to the next pick

}

##DATA_COMPLETE FRAME WITH NAMES AND R-sQUARES##
        #Creates a data frame that contains the regressors in the order by which they were
        picked along with their simple regression R-Squared and Joint R-Squares#

        index <- 1 #Start with the first regressor picked
        #Joint R-Squares for each group of regressors using probability pool
        Ordered_Joint_RSquares_Sel <- c()
        Ordered_Adjusted_RSquares_Sel <- c()

        limit <- min(Var_Size, Sample_Size/5)
        while(index <= limit){
        #Based on Probability Pool#
                #Gets data for Y (1) and selected regressors
                tempJointData <- DATA_COMPLETE[,c(1,
                (Ordered_Indexes_Sel[1:index]) + 1)]
                #NOTE: Regressor index is not the same as the index in the
                #DATA_COMPLETE data frame. Each regressor index is bumbed by one
                #unit in the data frame#
                TempJointReg <- lm(Y ~ ., data = tempJointData)
                #DATA_COMPLETEndexes_Ordered_Sel only contains regressor
                #indexes so it goes 1 - Var_Size#
                #Gets the Joint R-Squared for the selected regressors#
                Ordered_Joint_RSquares_Sel <- c(Ordered_Joint_RSquares_Sel,
                summary(TempJointReg)$r.squared)
                #Gets the Joint R-Squared for the selected regressors
                Ordered_Adjusted_RSquares_Sel <- c(Ordered_Adjusted_RSquares_Sel,
                summary(TempJointReg)$adj.r.squared)

                index <- index + 1 #Goes to the next regressor to be added

        }

#Combines all necessary information into one place#
```

```
    REGRESSORS_INFO_curr <- data.frame(Regressor = Ordered_Names_Sel,
JointRSquares = Ordered_Joint_RSquares_Sel, AdjustedRSquares =
Ordered_Adjusted_RSquares_Sel,
                                 Index = Ordered_Indexes_Sel, SelectionOrder = seq(1,limit),
Regressors = Selected_Indexes)

    ##ADDING A COLUMN FOR RUN NUMBER TO R-SQUARES DATA FRAME##

            Run <- rep(CURRENT_RUN, limit) #Run number
            REGRESSORS_INFO_curr <- dplyr::mutate(REGRESSORS_INFO_curr, Run)
            #Adds run number to the combined info

    ##UPDATING ALL FULL REGRESSION R-SQUARES##
            #Binds this run's information to the general combined information#
            REGRESSORS_INFO <- rbind(REGRESSORS_INFO,
            REGRESSORS_INFO_curr) #Binds current data frame to big data frame

            CURRENT_RUN <- CURRENT_RUN + 1 #Goes to the next run

  }


####RANDOM SELECTION####
  RUNS <- 1000 #Total number of times we will select regressors based on the created
  probability pool
  CURRENT_RUN <- 1 #start on first run
  RANDOM_REGRESSORS_INFO <- data.frame()

  #The following process will be repeated for each run
  while(CURRENT_RUN <= RUNS){

   #We need the random probability pool for each run
    Prob2 <- RANDOM_PROBABILITY_POOL

    ##Choosing Regressors - Probability Pool##
    #Chooses regressors using the probability pool created earlier#

    Random_Ordered_Indexes_Sel <- c()
    Random_Ordered_Names_Sel <- c()
    Random_Selected_Index <- c()
    Random_Selected_Indexes <- c()

    index <- 1

    limit <- min(Var_Size, Sample_Size/5)
```

```
while(index <= limit){

  #Using Random Probability Pool#
        #The index of the regressor that was picked at random
        random_picked <- sample(Prob2,1,T)
        #Removes the picked index from the Random Probability pool
        Prob2 <- Prob2[!Prob2 %in% random_picked]

        Random_Ordered_Indexes_Sel <- c(Random_Ordered_Indexes_Sel,
        random_picked) #Gets the index of the picked regressor

  ##UPDATES INDEXES SELECTED UP TO THAT POINT##
  if(length(Random_Selected_Indexes) == 0){

        Random_Selected_Index <- c(as.character(random_picked))

  }else{

        Random_Selected_Index <- paste0(Random_Selected_Index, ", ",
        random_picked)

  }

        Random_Selected_Indexes <- c(Random_Selected_Indexes,
        Random_Selected_Index)

        #Gets the name of the picked regressor
        Random_Ordered_Names_Sel <- c(Random_Ordered_Names_Sel,
        colnames(DATA_COMPLETE)[random_picked + 1])

        index = index + 1 #Goes to the next pick

}

##DATA_COMPLETE FRAME WITH NAMES AND R-sQUARES##
        #Creates a data frame that contains the regressors in the order by which they were
        #picked along with their simple regression R-Squared and Joint R-Squares#

        index <- 1 #Start with the first regressor picked
        #Joint R-Squares for each group of regressors using probability pool
        Ordered_Joint_RSquares_Sel <- c()
        #Joint R-Squares for each group of regressors using random probability pool
        Random_Ordered_Joint_RSquares_Sel <- c()
        #Joint R-Squares for each group of regressors using random probability pool
        Random_Ordered_Adjusted_RSquares_Sel <- c()
```

```
      limit <- min(Var_Size, Sample_Size/5)
      while(index <= limit){
  #Based on Ramdom Probability Pool#
      #Gets data for Y (1) and selected regressors
      tempJointData <- DATA_COMPLETE[,c(1,
      (Random_Ordered_Indexes_Sel[1:index]) + 1)]
  #NOTE: Regressor index is not the same as the index in the DATA_COMPLETE
  #data frame. Each regressor index is bumbed by one unit in the data frame#
      TempJointReg <- lm(Y ~ ., data = tempJointData)
  #DATA_COMPLETEndexes_Ordered_Sel only contains regressor indexes so it goes
  #1 - Var_Size#
  #Gets the Joint R-Squared for the selected regressors
  Random_Ordered_Joint_RSquares_Sel <- c(Random_Ordered_Joint_RSquares_Sel,
  summary(TempJointReg)$r.squared)
  Random_Ordered_Adjusted_RSquares_Sel <-
  c(Random_Ordered_Adjusted_RSquares_Sel,
  summary(TempJointReg)$adj.r.squared)

  index <- index + 1 #Goes to the next regressor to be added

}

#Combines all necessary information into one place#
      RANDOM_REGRESSORS_INFO_curr <- data.frame(Regressor =
      Random_Ordered_Names_Sel, JointRSquares =
      Random_Ordered_Joint_RSquares_Sel, AdjustedRSquares =
      Random_Ordered_Adjusted_RSquares_Sel,
                                  Index = Random_Ordered_Indexes_Sel, SelectionOrder =
      seq(1,limit), Regressors = Random_Selected_Indexes)


##ADDING A COLUMN FOR RUN NUMBER TO R-SQUARES DATA FRAME##

      Run <- rep(CURRENT_RUN, limit) #Run number
      RANDOM_REGRESSORS_INFO_curr <-
      dplyr::mutate(RANDOM_REGRESSORS_INFO_curr, Run) #Adds run number
      to the random combined info


##UPDATING ALL FULL REGRESSION R-SQUARES##
      #Binds this run's information to the general combined information#
      RANDOM_REGRESSORS_INFO <- rbind(RANDOM_REGRESSORS_INFO,
      RANDOM_REGRESSORS_INFO_curr)
```

```
        CURRENT_RUN <- CURRENT_RUN + 1 #Goes to the next run

    }


        rm(RANDOM_REGRESSORS_INFO_curr, REGRESSORS_INFO_curr,
        tempJointData)

    ####SUMMARY TABLE WITH ALL BEST PROBABLE SUBSETS####
        #Adjusted R-Squares using BPS#

        BEST_PROBABLE_SUBSETS <- REGRESSORS_INFO
        BEST_PROBABLE_SUBSETS <-
        BEST_PROBABLE_SUBSETS[order(BEST_PROBABLE_SUBSETS$JointRSq
        uares, decreasing = TRUE),]
        BEST_PROBABLE_SUBSETS <- group_by(BEST_PROBABLE_SUBSETS,
        SelectionOrder)
        BEST_PROBABLE_SUBSETS <- top_n(BEST_PROBABLE_SUBSETS, 1,
        JointRSquares)
        # View(BEST_PROBABLE_SUBSETS)
          BEST_PROBABLE_SUBSETS <- top_n(BEST_PROBABLE_SUBSETS, 1,
        Run)
        BEST_PROBABLE_SUBSETS <- ungroup(BEST_PROBABLE_SUBSETS)
          BEST_PROBABLE_SUBSETS <-
        BEST_PROBABLE_SUBSETS[order(BEST_PROBABLE_SUBSETS$Selection
        Order, decreasing = FALSE),]

        BEST_PROBABLE_SUBSETS <- data.frame(SubsetSize =
        BEST_PROBABLE_SUBSETS$SelectionOrder,
                          Regressors = BEST_PROBABLE_SUBSETS$Regressors,
                          RSquares = BEST_PROBABLE_SUBSETS$JointRSquares,
                                AdjustedRSquares =
        BEST_PROBABLE_SUBSETS$AdjustedRSquares,
                          AdjRSquaresPercentChange =
              Delt(BEST_PROBABLE_SUBSETS$AdjustedRSquares)*100)
        BEST_PROBABLE_SUBSETS$RSquares <-
        round(BEST_PROBABLE_SUBSETS$RSquares, 4)
        BEST_PROBABLE_SUBSETS$AdjustedRSquares <-
        round(BEST_PROBABLE_SUBSETS$AdjustedRSquares, 4)
        names(BEST_PROBABLE_SUBSETS)[5] <- c("AdjRSquaresPercentChange")
              BEST_PROBABLE_SUBSETS[,5] <-
        round(BEST_PROBABLE_SUBSETS[,5], 1)

   ####DATA FRAMES WITH BEST SUBS FOR ALL SUBSET SEL. METHODS####
        ##R-Squares Table##
        SQUARES_TABLE <- data.frame(SubsetSize =
        BEST_PROBABLE_SUBSETS$SubsetSize,
                    BPS = BEST_PROBABLE_SUBSETS$RSquares)


  #Best Subset Selection#
        modelSubset <- regsubsets(Y~., data = DATA_COMPLETE, nvmax = Var_Size)
        #nvmax is the number of variables to consider#
        summarySubset <- summary(modelSubset)

  #Forward Selection#
        modelForward <- regsubsets(Y~., data = DATA_COMPLETE, nvmax = limit,
        method = "forward") #nvmax is the number of variables to consider#
        summaryForward <- summary(modelForward)

  #Backward Elimination#
        modelBackward <- regsubsets(Y~., data = DATA_COMPLETE, nvmax =
        Var_Size, method = "backward") #nvmax is the number of variables to consider#
        summaryBackward <- summary(modelBackward)

  #StepWise Regression#
        modelStepwise <- regsubsets(Y~., data = DATA_COMPLETE, nvmax =
        Var_Size, method = "seqrep") #nvmax is the number of variables to consider#
        summaryStepwise <- summary(modelStepwise)

RSQUARES_TABLE <- mutate(RSQUARES_TABLE, BestSubset =
summarySubset$rsq, Forward = summaryForward$rsq,
                  Backward = summaryBackward$rsq, Stepwise = summaryStepwise$rsq)
  RSQUARES_TABLE[,-1] <- round(RSQUARES_TABLE[,-1], 4)

##Adjusted R-Squares Table##
        ADJUSTED_RSQUARES_TABLE <- data.frame(SubsetSize =
        BEST_PROBABLE_SUBSETS$SubsetSize,
                         BPS = BEST_PROBABLE_SUBSETS$AdjustedRSquares)


        ADJUSTED_RSQUARES_TABLE <-
        mutate(ADJUSTED_RSQUARES_TABLE, BestSubset = summarySubset$adjr2,
        Forward = summaryForward$adjr2,
                       Backward = summaryBackward$adjr2, Stepwise =
        summaryStepwise$adjr2)
        ADJUSTED_RSQUARES_TABLE[,-1] <-
        round(ADJUSTED_RSQUARES_TABLE[,-1], 4)

####BPS + BPS MSE####
```

```
        PLOT <- plot(x = BEST_PROBABLE_SUBSETS$SubsetSize, y =
        BEST_PROBABLE_SUBSETS$AdjustedRSquares,
          xlab = "Subset Size", ylab = "Adjusted R-Squared")
        BPS_Model <- lm(Y~Apps + Top25perc + P.Undergrad + Outstate +
        Room.Board + perc.alumni + Expend, data = DATA_COMPLETE)
        BPSModelCoeff <- as.data.frame(summary(BPS_Model)$coeff)
         BPS_MSE <- sum((BPS_Model$fitted.values -
        DATA_COMPLETE$Y)^2)/Sample_Size

        FittedValues <- BPS_Model$fitted.values
        Residuals <- Y - FittedValues

        plot(FittedValues, Residuals)


####RANDOM FORESTS####
        RF <- randomForest(Y~.,data = DATA_COMPLETE, ntree = 100, mtry =
        Var_Size/3)

        RF_DATA <- data.frame(Measure = c("Number of Trees", "Splits per Node",
        "MSE"),
        Value = c(RF$ntree, RF$mtry, sum((RF$predicted-
        DATA_COMPLETE$Y)^2)/Sample_Size))


####END OF SCRIPT####-----------------------------------------------------------------------
```