# Project 2: Examinings Methods to Improve Linear Regression

*Shannon Chang and Nura Kawa*

*November 4, 2016*

## Abstract

The goal of modeling data is to be able to predict future observations with high accuracy and minimal error. In previous papers we have discussed in detail the *Linear Model*, which uses Least-Squares approximation to regress a response variable on a matrix of predictors. In this paper we discuss performance issues of the Multiple Linear Regression model when fit to large or highly-correlated datasets, such as overfitting. We focus on algorithms created that improve this linear model through dimensionality reduction and shrinkage methods, namely: Partial Least Squares Regression, Principal Component Regression, Ridge Regression, and LASSO.

## Introduction

The linear model is popular and often accurate, but not without its downfalls. The recent increase in availability and dimensionality of data has introduced scenarios where the ordinary Least-Squares regression method performs poorly on certain datasets. As a result, statisticians and mathematicians have developed algorithms that are improvements on the linear model. This paper reproduces content from the book *Introduction to Statistical Learning* by Hastie, Tibshirani, James, and Witten, which discusses four penalized regression algorithms, fits these models to data, and evaluates results.

The book uses a dataset called `Credit`, which the authors have made available publically online for purposes of reproduction: http://www-bcf.usc.edu/~gareth/ISL/data.html

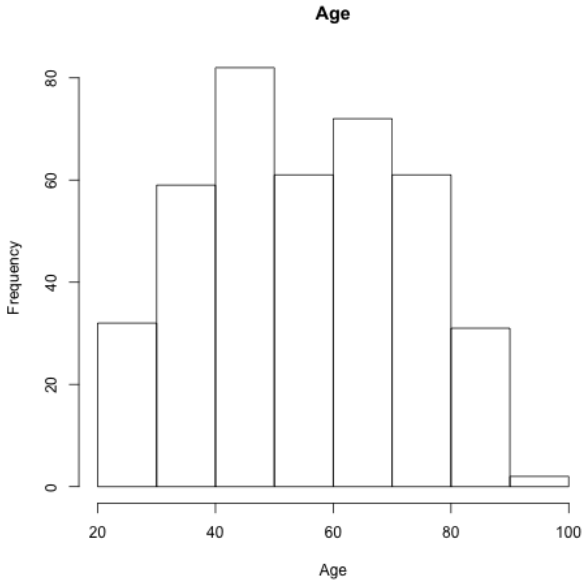
## Data

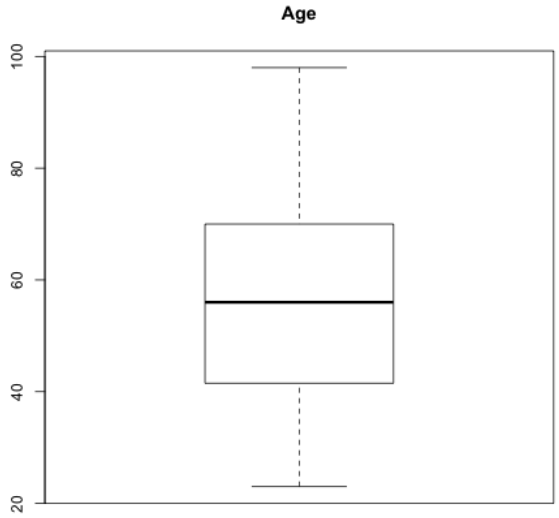
The `Credit.csv` dataset consists of `balance` observations (average credit card debt) for 400 different individuals, as well as observations for a number of quantitative and qualitative variables detailed below:

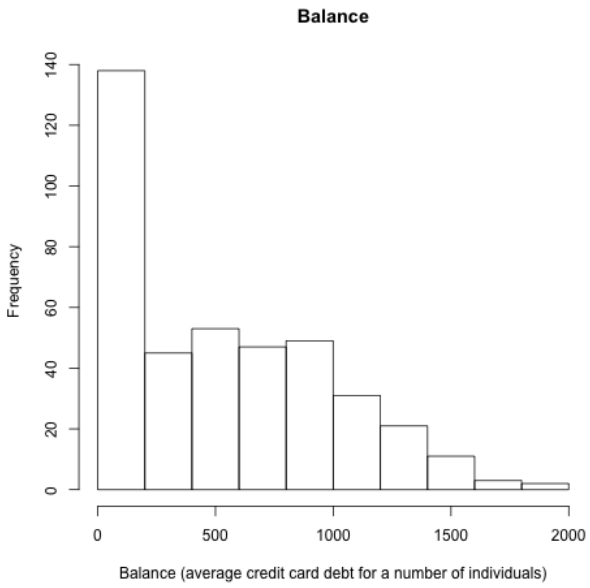
### Quantitative Variables

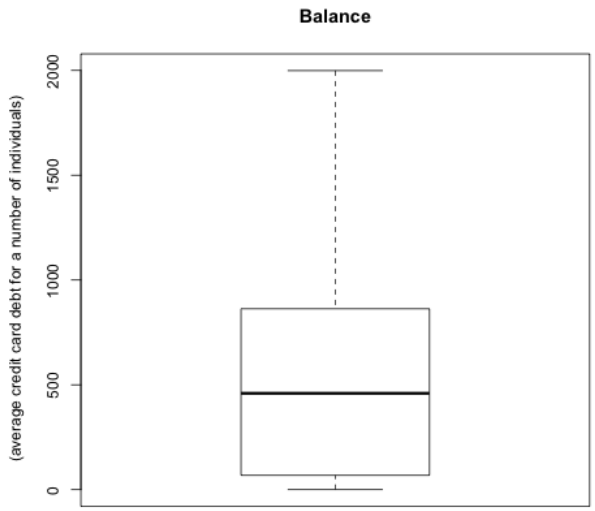
- `age`
- `cards` (number of credit cards)
- `education` (years of education)
- `income`(in thousands of dollars)
- `limit` (credit limit)
- `rating` (credit rating)

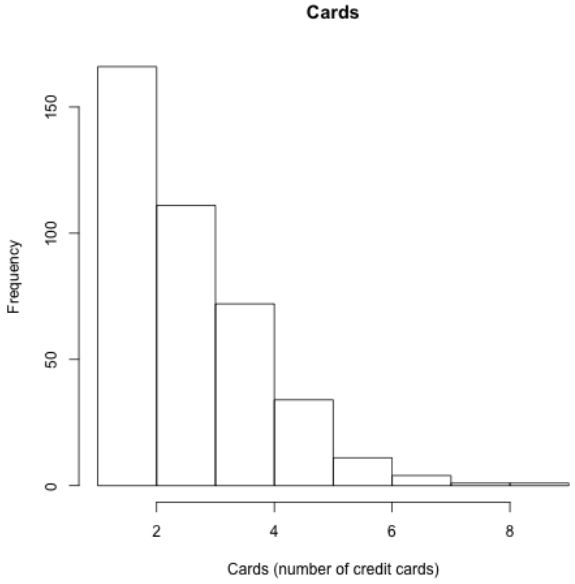
A very general overview of the distribution for each variable is provided in the histograms and boxplots below:

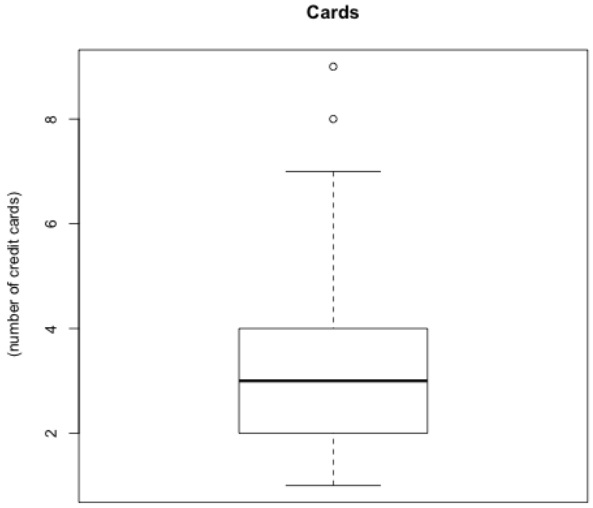


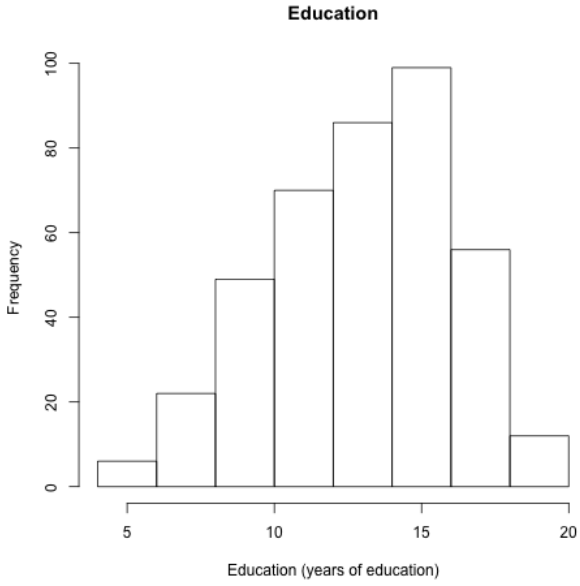



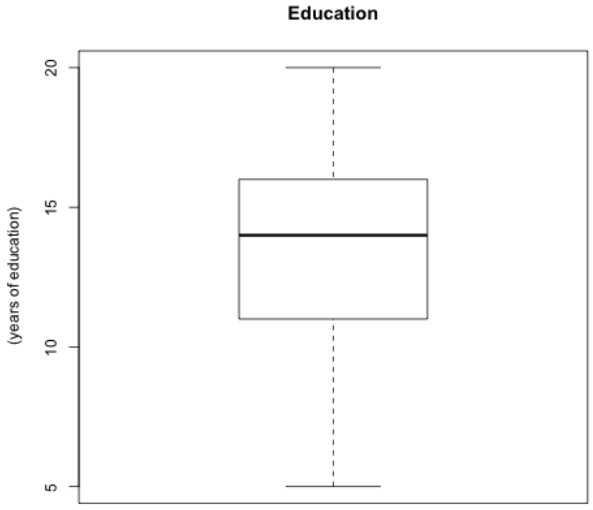



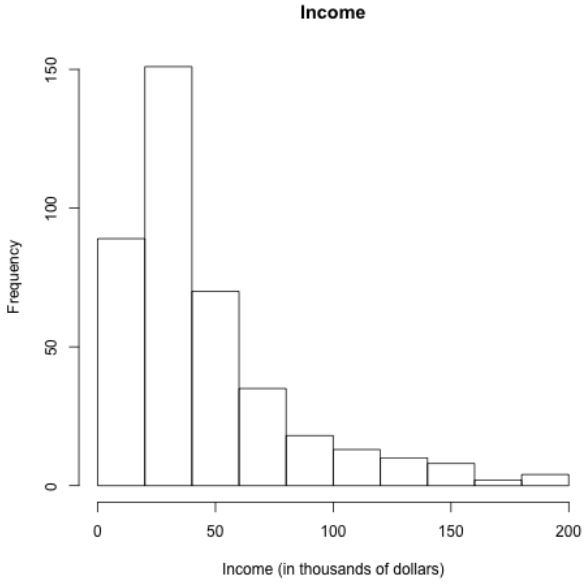



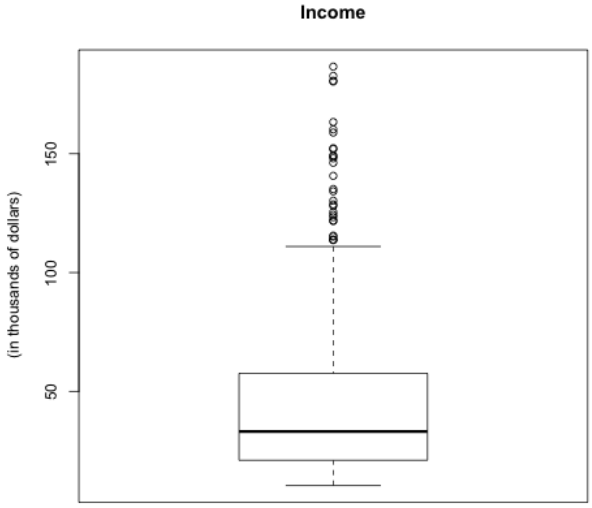

Cards
Frequency
0
50
100
150
2
4
6
8
Cards (number of credit cards)

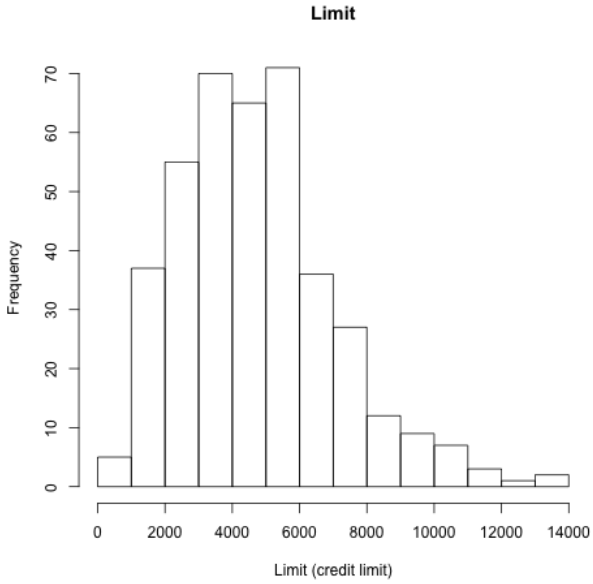

Cards
(number of credit cards)
2
4
6
8

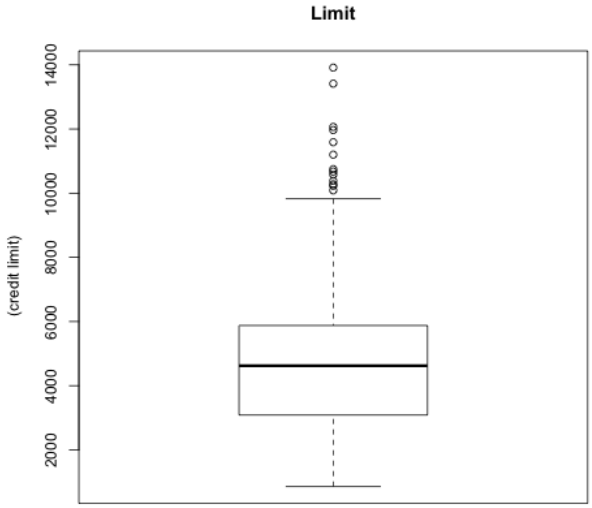

Education
Frequency
0
20
40
60
80
100
5
10
15
20
Education (years of education)

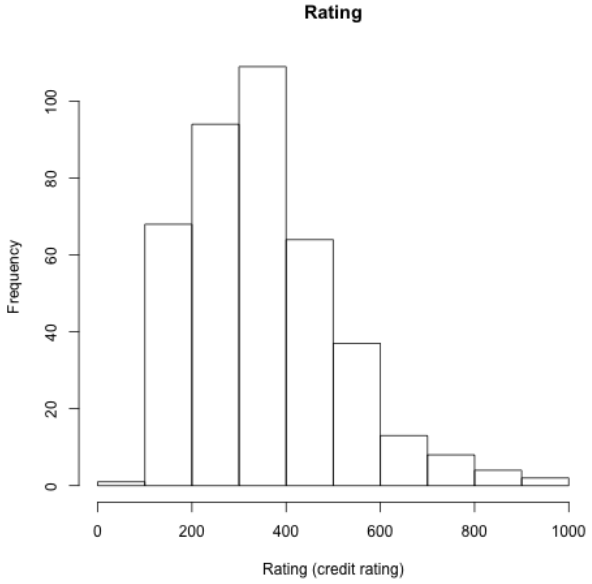

Education
(years of education)
5
10
15
20

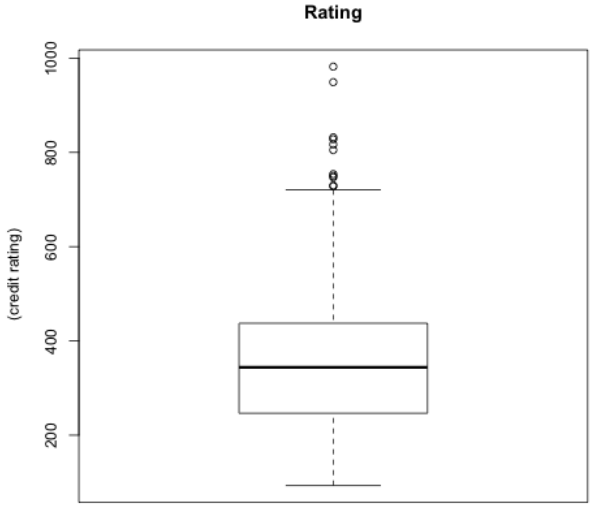

Income
Frequency
0
50
100
150
0
50
100
150
200
Income (in thousands of dollars)


Income
(in thousands of dollars)
50
100
150


Limit
Frequency
0
10
20
30
40
50
60
70
0
2000
4000
6000
8000
10000
12000
14000
Limit (credit limit)


Limit
(credit limit)
2000
4000
6000
8000
10000
12000
14000




A scatterplot matrix for all quantitative variables and a matrix of correlations are displayed below:

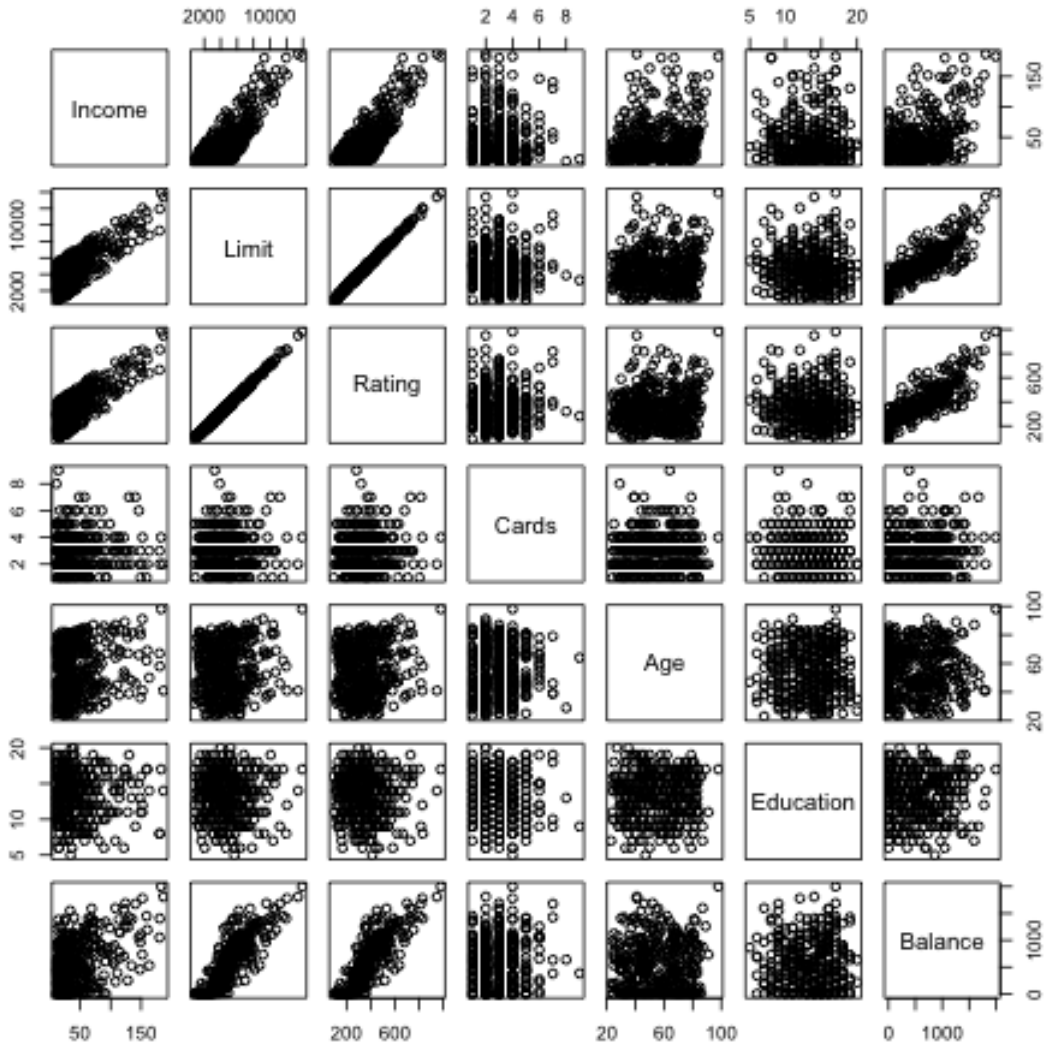



Table 1: Matrix of Correlations for all Quantitative Variables

| | Income | Limit | Rating | Cards | Age | Education | Balance |
|---|---|---|---|---|---|---|---|
| Income | 1.0000 | 0.7921 | 0.7914 | -0.0183 | 0.1753 | -0.0277 | 0.4637 |
| Limit | 0.7921 | 1.0000 | 0.9969 | 0.0102 | 0.1009 | -0.0235 | 0.8617 |
| Rating | 0.7914 | 0.9969 | 1.0000 | 0.0532 | 0.1032 | -0.0301 | 0.8636 |
| Cards | -0.0183 | 0.0102 | 0.0532 | 1.0000 | 0.0429 | -0.0511 | 0.0865 |
| Age | 0.1753 | 0.1009 | 0.1032 | 0.0429 | 1.0000 | 0.0036 | 0.0018 |
| Education | -0.0277 | -0.0235 | -0.0301 | -0.0511 | 0.0036 | 1.0000 | -0.0081 |
| Balance | 0.4637 | 0.8617 | 0.8636 | 0.0865 | 0.0018 | -0.0081 | 1.0000 |

## Qualitative Variables

- `gender` (m/f)
- `student` (yes/no)

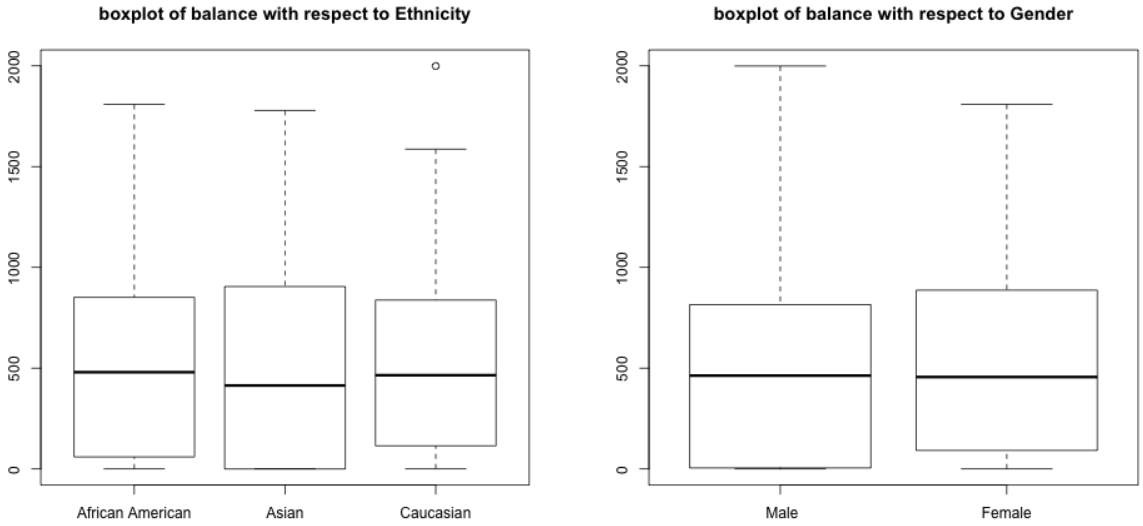
- `married` (yes/no)
- `ethnicity`(caucasian/asian/african american)

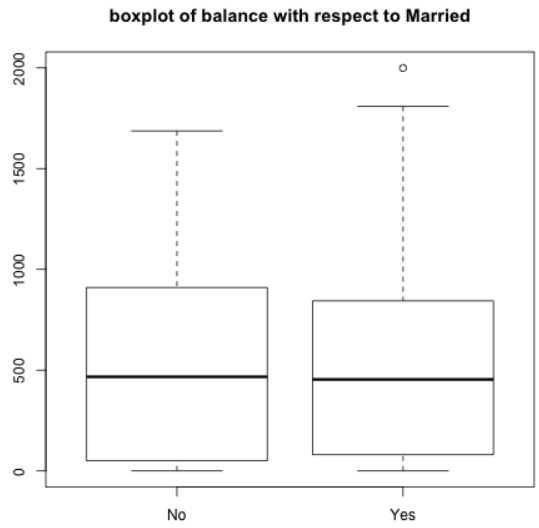
All of these variables are factors with two or three levels, so the best way to explore their distribution is through frequency tables and plots that display `Balance` with respect to a qualitative variable.

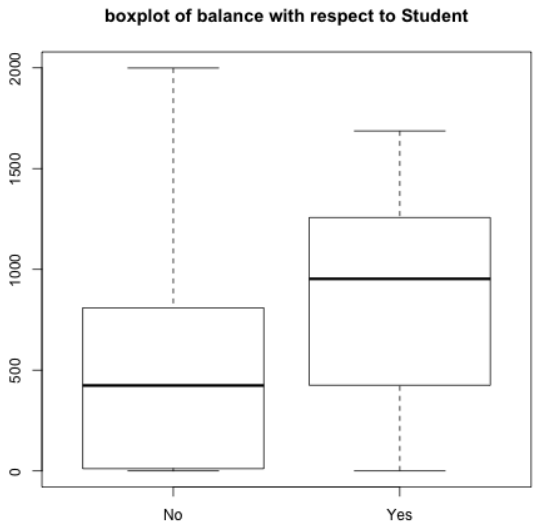
Below is a proportional frequency table for all qualitative variables:

```
##                         Ethnicity African American  Asian Caucasian
## Gender Student Married
##  Male  No      No                          0.0450 0.0400    0.0825
##                Yes                         0.0650 0.0675    0.1425
##        Yes     No                          0.0050 0.0075    0.0100
##                Yes                         0.0075 0.0025    0.0075
## Female No      No                          0.0625 0.0250    0.0825
##                Yes                         0.0500 0.0900    0.1475
##        Yes     No                          0.0050 0.0075    0.0150
##                Yes                         0.0075 0.0150    0.0100
```

Below are conditional boxplots of `Balance` with respect to each qualitative variable:






## Data Pre-Processing

### Scaling

Using data with multiple predictors presents the problem of different scalings. For example, a person's age is typically between 0 and 100, while a person's income is at a much larger scale. In order to use both as predictors for a response, it is essential to standardize their ranges. To do this, we use the R function "scale", which subtracts from each vector its mean and divides each vector by its standard deviation.

### Training and Testing Sets

Oftentimes fitting a model to data is for the purpose of predicting future observations. Given a full dataset we can simulate "past" and "future" observations by dividing the data into *training* and *testing* sets. A model is fit, or "trained", to the training set. The testing set becomes "new" data that the model tries to predict. Since we have the response values of our testing set, we can calculate the difference between our predictions and our true values to assess model. Here is what we did:

```
# take a random sample whose size is 75% of the number of observations of our data
# (number_rows):

# compute total number of rows of our data; should be 400
number_rows <- nrow(scaled_credit)

# set seed
set.seed(10)

# take 75% of the rows as training by randomly sampling from number_rows
training_rows <- sample(1:number_rows, 0.75*number_rows,
                        replace = F)

# assign training and testing sets based on these rows

# y is our response, "Balance".
```

```r
y <- as.matrix(scaled_credit$Balance)

# x is our predictors
x <- as.matrix(scaled_credit[,-ncol(scaled_credit)]) #removing "balance"

# split into training and testing for x and y
y_train <- y[training_rows,]
x_train <- x[training_rows,]
y_test <- y[-training_rows,]
x_test <- x[-training_rows,]
```

# Methods

In this section we discuss conceptually regression algorithms developed to improve the accuracy, bias, and variance of the Least Squares linear model: Ridge Regression, LASSO Regression, Principal Componenent Regression, and Partial Least Squares Regression. We also introduce the Mean Squared Error as a parameter to evaluate quality of fit.

## Improving the Linear Model

The goal of modeling data is to be able to predict future observations with high accuracy and minimal error. A model's prediction error comes from three sources: random error, bias, and variance. Bias measures how far off a model's prediction is from the correct value. Variance is a measure of how much the predictions for a given point vary between different realizations of the model. The following image visualizes the types of error:

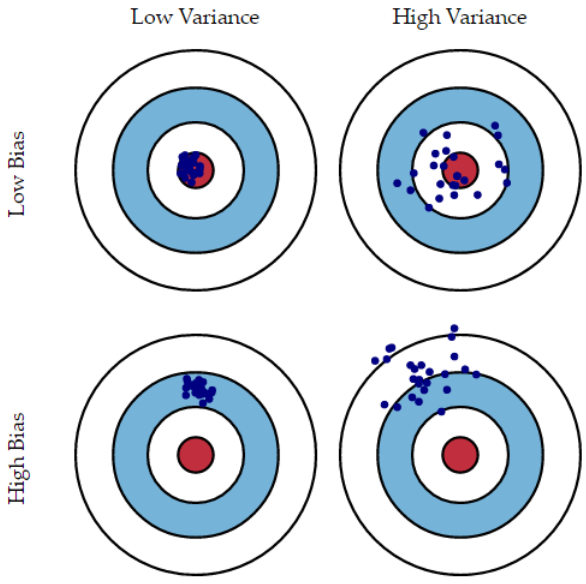



Figure 1: Prediction Error due to Bias and Variance

The linear model with several explanatory variables is given by the equation:

$$Y = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p + \epsilon,$$

and we define our parameter space to be $n$ observations of $p$ predictors. The Least Squares model has low bias and low variance when $n >> p$ (we have many more observations than predictors). However, when $n > p$ (we have more observations than parameters, but not a lot more) the Least Squares model overfits the data and has high variance. If $p > n$, a unique Least Squares solution cannot be found. To correct these problems statisticians have developed algorithms for *Shrinkage* (for cases where $n > p$) and *Dimensionality Reduction* (for cases where $p$ is relatively large compared to $n$).

## Shrinkage

To reduce variance we fit a model containing all $p$ predictors using a technique that shrinks coefficient estimates towards zero by adding an additional optimization contraint called a *shrinkage penalty.*

### Ridge Regression

Recall that the Least Squares model aims to minimize Residuals Sum of Squares, defined as:

$$RSS = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2$$

Ridge Regression is a penalized linear least-squares model whose tuning parameter $\lambda$ controls the impact of its shrinkage penalty.

The coeffiecient estimates of Ridge Regression, $\hat{\beta_{RIDGE}}$ minimize the RSS + a shrinkage penalty:

$$RSS + \lambda \sum_{j=1}^{p} \beta_j^2$$

.

As $\lambda \to \infty$ the severity of the penalty increases, allowing us to select a lambda to create the most accurate model. We use Cross Validation to tune our parameter (this method is discussed later).

### LASSO

Least Absolute Shrinkage and Selection Operator, commonly called LASSO, has the same algorithm as Ridge Regression, but while Ridge Regression adds an $L_2$ norm as an optimization contraint, LASSO uses an $L_1$ norm. This has the effect of requiring certain coefficients to be shrunk to *exactly* 0 for significantly large values of $\lambda$, performing shrinkage and variable selection simulataneously.

Specifically, LASSO minimizes the following quantity:

$$RSS + \lambda \sum_{j=1}^{p} |\beta_j|$$

.

Like Ridge Regression, the tuning parameter can be optimally selected using Cross Validation.

### Cross Validation

To select the best value of $\lambda$ we select a grid of values, compute coefficients for each value, and compute a measure of error. We select the *lambda* that minimizes the cross-validation error (there is some flexibility of this measure; often Mean Squared Error is used) and use this value of our tuning parameter to fit our final model. There are actually different forms of cross-validation used by statisticians; in this paper we use the method discussed above, called *n-fold* cross-validation, where $n$ is traditionally at least 10.

### Comparing Ridge Regression with LASSO

While LASSO is by far more popular than Ridge Regression in industry due to sparsity of the LASSO model and its ease of interpretability, it does not always outperform Ridge Regression. Both methods **reduce variance**, so as $\lambda$ increases the model variance decreases, but due to the *Bias-Variance Tradeoff*, the bias increases. LASSO often has slightly higher bias than does Ridge Regression. It also is inaccurate in scenarios where no coefficient is truly zero - as it, every predictor is somewhat significant. Thus, while it is often tempting to simply run LASSO on data, it is important to consider fitting Ridge Regression or to use Elastic-Net Regression, a combination of Ridge and LASSO.

## Dimensionality Reduction

In situations where p is relatively large compared to n (there are many predictors per response), Least Squares fails to fit a unique linear model to the data and often overfits predictions. It thus becomes imperative to somehow reduce the dimensionality of our data while losing as little information as possible. We will discuss two regression algorithms that derive low-dimensional feature spaces from data before fitting a linear model.

### Principal Component Regression

The first step of Principal Component Regression is the step of dimensionality reduction, Principal Componenet Analysis (PCA). A popular visualization tool, PCA allows us to view our data as principal components, or vectors that capture the majority of the variance between observations. The process of PCA is mathematically Singular Value Decomposition. Principal components are generated by recursively projecting data in the direction that shows the maximum variance between observations; the first PC has the most variance, followed by the second, and so on.

Principal Component Regression constructs $M$ principal components as predictors and then fits a linear model using the Least Squares method. The idea is that only a few principal components are needed to capture the majority of the variance explained by the data, and that fitting M principal components to a linear model will not overfit the data as would Least Squares regression will all $p$ predictors.

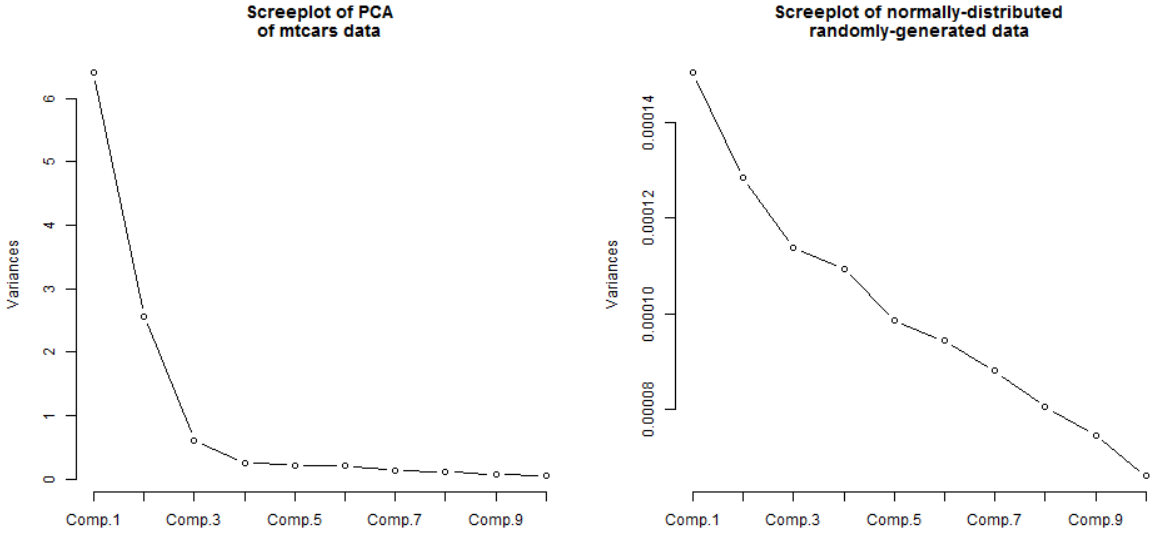
### Partial Least Squares Regression

Partial Least Squares Regression (PLSR) is a supervised version of PCR, where the same M-dimensional space of linear combinations of data is used for predictors, but the response, Y, is used to identify directions that explain both the response and predictors.

After standardizing the data, PLSR generates the first directional vector by computing correlation between the feature space and the calculated regression coefficient of Y regressed onto its predictors, thus placing the highest weight on the most correlated predictors. The second direction is created by regressing each predictor onto the first direction and taking the residuals, which is information not explained by the first component. The second direction is thus computed from these redsiduals. The next step is iterative; compute the third direction from residuals from a regression on the second component, and so on. The final model fits Least Squares onto the computed directions.

**When to reduce dimensions?**

Typically, when there are many predictors with respect to the number of responses, dimensionality reduction is a good idea. However, not all datasets can be reduced to few principal components. As more principal components are used by a model, the bias decreases, but the variance naturally increases. Thus, a good indicator of the performance of PCR or PLSR would be a screeplot of the PCA-created principal components generated in the dimensionality-reduction step. A *screeplot* plots the proportion of variance explained per componenent in a bar- or line-chart. A good PCA has an "elbow" in its screeplot. Below are Principal Component Analyses of two different datasets: one is the popular "mtcars" dataset whose dimension is 32 rows by 11 columns. Another is a matrix of normally-distributed points of dimension 200 rows by 10 columns.




Note that the first dataset has the majority of variance explained by three or four components, making it a good candidate for dimensionality reduction. The right-hand screeplot, however, does not have an "elbow", and requires many components to explain the variance between observation. Thus, PLSR or PCR may not perform well on this data.

## Evaluating Quality of Fit

To measure the quality of a fit model, we need to examine how well the model's predictions actually match the observed data. The most commonly used method to do this is through *mean squared error* (MSE), which is calculated for the training data as:

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

where $\hat{f}$ is a prediction of $f$, a function that describes the numerical relationship between a response variable $Y$ and its predictors $X_1, X_2, ..., X_n$. $\hat{f}(x_i)$ is then the prediction that $\hat{f}$ gives for the _i_th observation.

However, we are not truly examining whether $\hat{f}(x_i) \approx y_i$. Instead, we are examining whether $\hat{f}(x_0) \approx y_0$, where ($x_0$, $y_0$) is a previously unseen test observation that was not a part of the training data. For both formulations of the MSE, the result will be small if the predicted responses are very close to their true values, and will be very large if predicted and true responses differ substantially for some of the observations.

To determine the accuracy of the predictions that we obtain when we fit our model on previously unseen test data, we would choose the method that gives us the *lowest test MSE*. To be specific, for a large number of test observations, we would compute $Ave(y_0 - \hat{f}(x_0))^2$, the average squared prediction error for test observations $(x_0, y_0)$. The model for which this calculated value is the lowest would then be best model overall to fit a dataset of interest.

# Analysis

In this section, we discuss the workflow process for executing Ordinary Least Squares (OLS), Ridge, LASSO, Principal Components, and Partial Least Squares Regressions on the `Credit-scaled.csv` dataset. (`Credit-scaled.csv` contains data from `Credit.csv` that was processed for regression analysis through factor conversion into dummy variables, mean centering, and variable standardization.)

For each regression method, we fit a model on testing data, calculate an MSE value based on the testing data, and fit the model again on the entire dataset for comparison.

All models for methods other than OLS are fit using ten-fold cross-validation. Since cross validation works through resampling data, we set a random seed before running fitting functions for reproducibility purposes. We then select the best model from a list of several produced by each fitting function using a tuning parameter. For shrinkage methods, the tuning parameter is $\lambda$; for dimension reduction methods, the parameter is the number of components.

The `R` code used for all five methods is roughly outlined in the sections below:

## Ordinary Least Squares

```
# Fit ordinary least squares regression
ols_fit <- lm(y_train~x_train)
ols_fit_sum <- summary(ols_fit)

# Compute mean square error for the test set
ols_mse <- mean(summary(lm(y_test~x_test)$residuals^2))

# Refit the OLS regression on the full data set
full_data_ols_fit <- lm(y~x)
```

## Shrinkage Methods

Both methods utilize the `glmnet` package.

### Ridge Regression

```
# Fit the ridge model on training data
ridge_fit <- cv.glmnet(x_train,
                       y_train,
                       alpha = 0, # ridge parameter
                       lambda = grid <- 10^seq(10, -2, length = 100),
                       nfolds = 10, # performs 10-fold cross validation
                       standardize = FALSE, # we already standardized our data
                       intercept = FALSE)
```

```r
# Select the best model
ridge_best_model <- ridge_fit$lambda.min

# Compute mean square error for the test set
ridge_predictions <- predict(ridge_fit, x_test, s = ridge_best_model)

ridge_mse <- mean((y_test - ridge_predictions)^2)

# Refit the ridge model on the full data set
full_data_ridge_fit <- glmnet(x,
                              y,
                              alpha = 0, # ridge parameter
                              lambda = ridge_best_model,
                              standardize = FALSE,
                              intercept = FALSE)
```

**LASSO Regression**

```r
# Fit the LASSO model on the training data
lasso_fit <- cv.glmnet(x_train,
                       y_train,
                       alpha = 1, # LASSO parameter
                       lambda = grid <- 10^seq(10, -2, length = 100),
                       nfolds = 10, # performs 10-fold cross validation
                       standardize = F, # we already standardize our data
                       intercept = F)

# Select the best model
lasso_best_model <- lasso_fit$lambda.min

# Compute mean square error for the test set
lasso_predictions <- predict(lasso_fit, x_test, s = lasso_best_model)

lasso_mse <- mean((y_test - lasso_predictions)^2)

# Refit the LASSO model on the full data set
full_data_lasso_fit <- glmnet(x,
                              y,
                              alpha=1, #LASSO
                              lambda = lasso_best_model,
                              standardize = F,
                              intercept = F)
```

## Dimension Reduction Methods

Both methods utilize the `pls` package.

### Principal Components Regression (PCR)

```
# Fit the PCR model on the training data
pcr_fit <- pcr(y_train~x_train,
               scale = FALSE,
               validation = "CV" # performs 10-fold cross validation
               )

# Select the best model
pcr_best_model <- which.min(pcr_fit$validation$PRESS) #best number of components

# Compute mean square error for the test set
pcr_predictions <- predict(pcr_fit, x_test, s = pcr_best_model)

pcr_mse <- mean((y_test - pcr_predictions)^2)

# Refit the PCR model on the full data set
full_data_pcr_fit <- pcr(y~x,
                         validation = "CV")
```

### Partial Least Squares Regression (PLSR)

```
# Fit the PCR model on the training data
pls_fit <- plsr(y_train ~ x_train,
                scale = FALSE,
                validation = "CV")

# Select the best model
pls_best_model <- which.min(pls_fit$validation$PRESS) #best number of components

# Compute mean square error for the test set
pls_predictions <- predict(pls_fit, x_test, s = pls_best_model)

pls_mse <- mean((y_test - pls_predictions)^2)

# Refit the PLSR model on the full data set
full_data_pls_fit <- plsr(y ~ x,
                          scale = FALSE,
                          validation = "CV",
                          ncomp = pls_best_model)
```

## Results

We now look at the results of fitting four regression models to our dataset, evaluating regresson coefficients and MSE values to find a best fit.

### Regression Coefficients

Below is a table of all regression coefficients for each fit to the full dataset, as well as a visualization.

| | ridge | lasso | pcr | pls | ols |
|---|---|---|---|---|---|
| Income | -0.568707 | -0.551661 | -0.598171 | -0.598170 | -0.598171 |
| Limit | 0.718658 | 0.925047 | 0.958439 | 0.958436 | 0.958439 |
| Rating | 0.593059 | 0.367875 | 0.382479 | 0.382480 | 0.382479 |
| Cards | 0.044253 | 0.044998 | 0.052865 | 0.052853 | 0.052865 |
| Age | -0.025385 | -0.016660 | -0.023033 | -0.023027 | -0.023033 |
| Education | -0.005880 | 0.000000 | -0.007469 | -0.007482 | -0.007469 |
| GenderFemale | -0.010678 | 0.000000 | -0.011593 | -0.011628 | -0.011593 |
| StudentYes | 0.273184 | 0.266813 | 0.278155 | 0.278163 | 0.278155 |
| MarriedYes | -0.011028 | 0.000000 | -0.009054 | -0.009085 | -0.009054 |
| EthnicityAsian | 0.016379 | 0.000000 | 0.015951 | 0.015951 | 0.015951 |
| EthnicityCaucasian | 0.011012 | 0.000000 | 0.011005 | 0.010998 | 0.011005 |

All models had coefficients within a very close range, with no surprising or easily-spotted differences. In all fits the largest positive coefficient is `Limit`, and the largest negative coefficient is `Income`, meaning that regardless of model, these are the most significant predictors in our dataset. Our Ridge model set slightly higher coefficients for `Ranking` and `Married(Yes)`. The LASSO shrunk the predictors `Education`, `Gender`, `Married`, and `Ethnicity` to zero, while all other methods had very small coefficient values for these predictors.

**MSE**

We next look at the results of our predictions from models fit to the training data. The Mean Squared Errors for each method are shown below:

| | mse |
|---|---|
| ridge | 0.05049 |
| lasso | 0.05112 |
| pcr | 0.05021 |
| pls | 0.04998 |
| ols | 0.06294 |

Interestingly, all methods have very close MSE values, so it is not easy to select a best method. By lowest MSE our winner is Parial Least Squares Regression; however, re-fitting the model several times could result in slightly different prediction error rates for all method. Most importantly, all penalized regression methods out-performed Ordinary Least Squares, showing that for this dataset these methods did indeed improve the performance of a basic linear model.

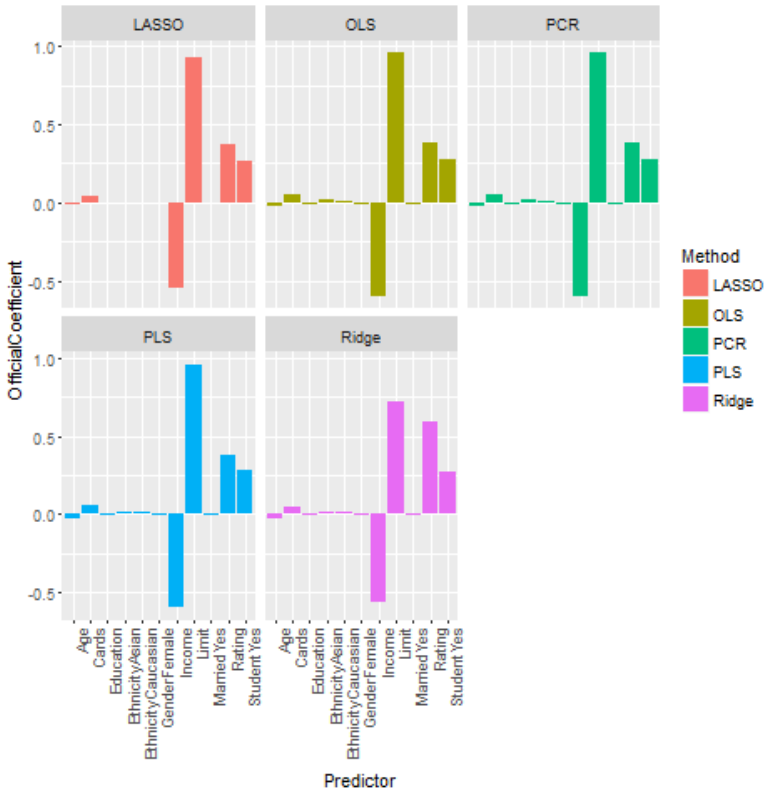



Figure 2: Coefficient Plot

# Conclusions

Our results show highly similar test MSE values and coefficients. The question now becomes the selection of a "winning method". Ultimately, we find that our best method is Partial Least Squares Regression due to its having the smallest Mean Squared Error.

However, as mentioned before, reproducing this paper several times could have slightly different results in terms of both coefficients and MSE values. It is also important to note that the `Credit` dataset is relatively "small", with only 400 observations and 11 predictors. Our penalized regression methods only showed a small improvement over Least Squares. Our results would have likely been more varied had we been given more observations and/or more predictors. There likely would have been a larger difference in Mean Squared Error as well, making it easier to select a "winning method".

Thus, we conclude that the best model for this dataset is PLS Regression, possibly due to the combination of supervised prediction with dimensionality reduction. However, the difference between performance of all methods is quite small, showing that PLS won only narrowly.