Solutions to Homework 2

Nakul Camasamudram

Problem 1

Members of my project group: - Nakul Camasamudram, MS Computer Science - Summer 2018 - Abhay Kasturia, MS Computer Science - Spring 2018

Problem 2

(a)

```
set.seed(123)
n = 100
x <- runif(n, min = -2, max = 2)
e <- rnorm(n, mean = 0, sd = 4)
y <- 2 +(3*x) + e

mean_x = mean(x)
mean_y = mean(y)

b_1 <- sum((x - mean_x) * (y - mean_y)) / sum((x-mean_x)^2)
b_0 <- mean_y - (b_1 * mean_x)

cat("Estimates:\n")
```

```
> Estimates:
```

```
cat("Coeffecient b1: ", b_1, "\n")
```

```
> Coeffecient b1:  2.910169
```

```
cat("Slope b0: ", b_0, "\n")
```

```
> Slope b0:  1.784498
```

The values of slope(1.78) and coefficient(2.91) are pretty close to the actual values (2 and 3 respectively).

(b)

Stochastic Gradient Descent:

```
set.seed(123)

stochastic_gradient_descent <- function(x, y, learn_rate){
  # set.seed(123) ensures that b_0 and b_1 will never be 0
  b_0 <- runif(1)
  b_1 <- runif(1)

  # learn_rate <- 0.01
  b_0_prev <- 0
  b_1_prev <- 0

  i = 0
```

```r
  while(i < 1000 && (abs(b_1_prev - b_1) > 0.1 || abs(b_0_prev - b_0) > 0.1)){
    i = i + 1
    b_0_prev <- b_0
    b_1_prev <- b_1
    for(k in 1:length(x)){
      b_0 = b_0 + (learn_rate * ((y[k] - (b_0 + b_1*x[k]))*x[k]))
      b_1 = b_1 + (learn_rate * ((y[k] - (b_0 + b_1*x[k]))*x[k]))
    }
    i = i+1
  }
  return(c(i, b_0, b_1))
}

out <- stochastic_gradient_descent(x, y, 0.01)
cat("Estimates:\n")
```

```
> Estimates:
```

```r
cat("Coefficient b1: ", out[2], "\n")
```

```
> Coefficient b1:  2.441518
```

```r
cat("Slope b0: ", out[3], "\n")
```

```
> Slope b0:  2.939866
```

Batch Gradient Descent:

```r
set.seed(123)

batch_gradient_descent <- function(x, y, learn_rate) {
  # set.seed(123) ensures that b_0 and b_1 will never be 0
  b_0 <- runif(1)
  b_1 <- runif(1)

  # learn_rate <- 0.01
  b_0_prev <- 0
  b_1_prev <- 0

  i = 0
  while(i < 1000 && (abs(b_1_prev - b_1) > 0.1 || abs(b_0_prev - b_0) > 0.1)){
    i = i + 1
    b_0_prev <- b_0
    b_1_prev <- b_1
    b_0 = b_0 + learn_rate * sum(1.0 * (y - (b_0 + b_1*x)))
    b_1 = b_1 + learn_rate * sum(x * (y - (b_0 + b_1*x)))
  }

  return(c(i, b_0, b_1))

}

out <- batch_gradient_descent(x, y, 0.01)
cat("Estimates:\n")
```

```
> Estimates:
```

```r
cat("Coefficient b1: ", out[3], "\n")
```

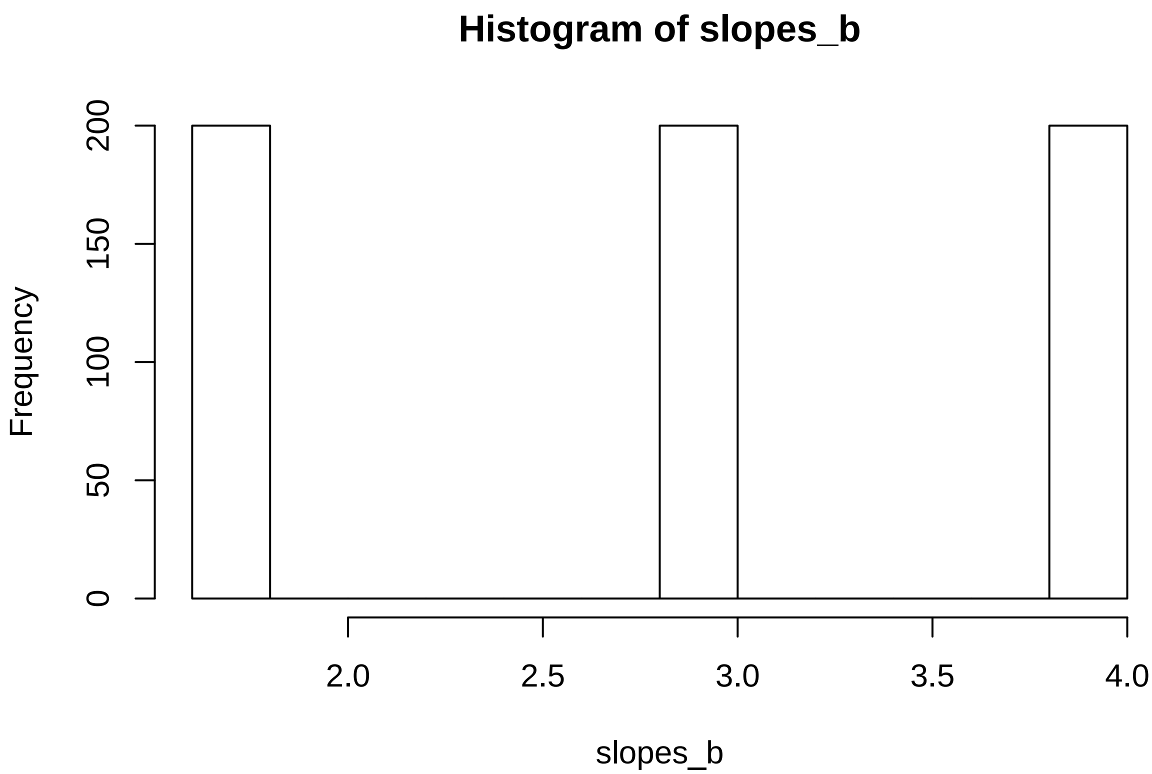
```
> Coefficient b1:  2.89586
```

```r
cat("Slope b0: ", out[2], "\n")
```

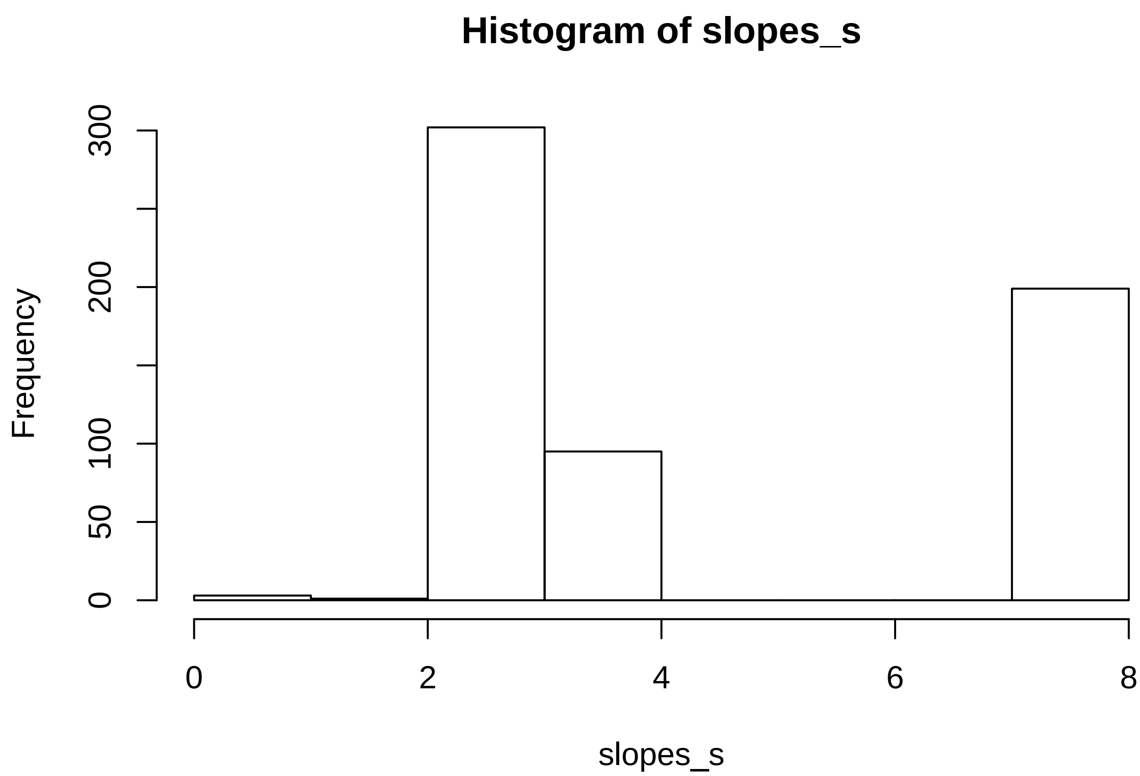
```
> Slope b0:  1.784786
```

(d)

```r
slopes_b <- c()
slopes_s <- c()

for(i in 1:200) {
  slopes_b <- c(slopes_b, batch_gradient_descent(x, y, 0.01))
  slopes_s <- c(slopes_s, stochastic_gradient_descent(x, y, 0.01))
}

hist(slopes_b)
```



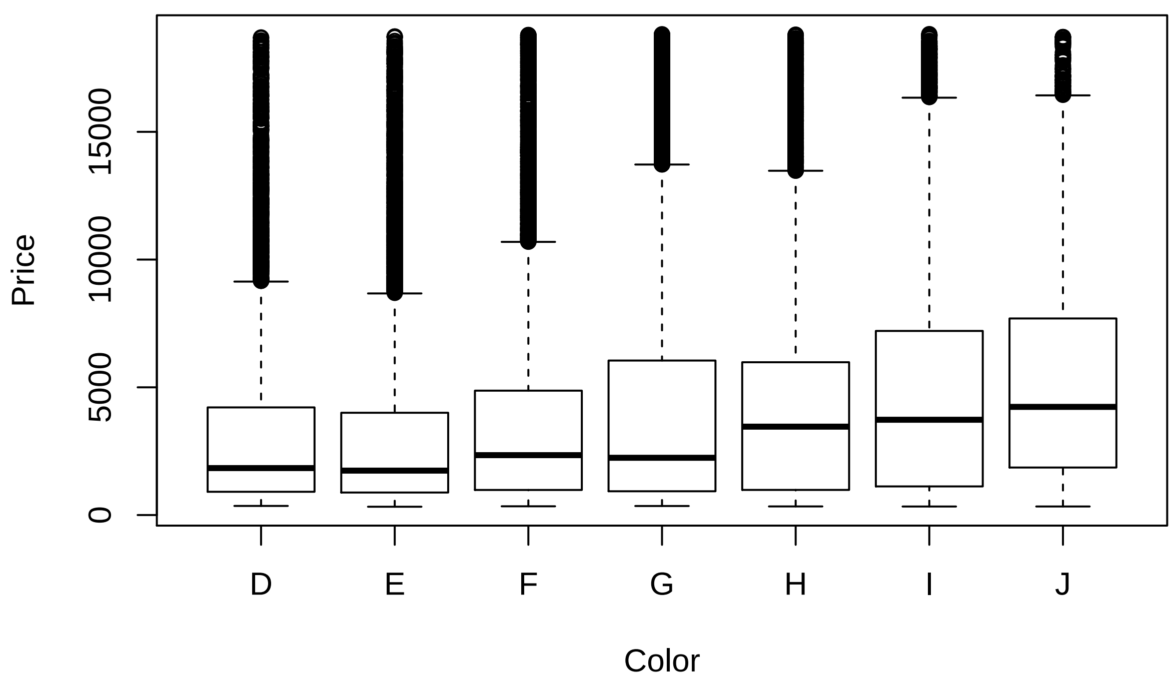

```r
hist(slopes_s)
```



Problem 3

(a)

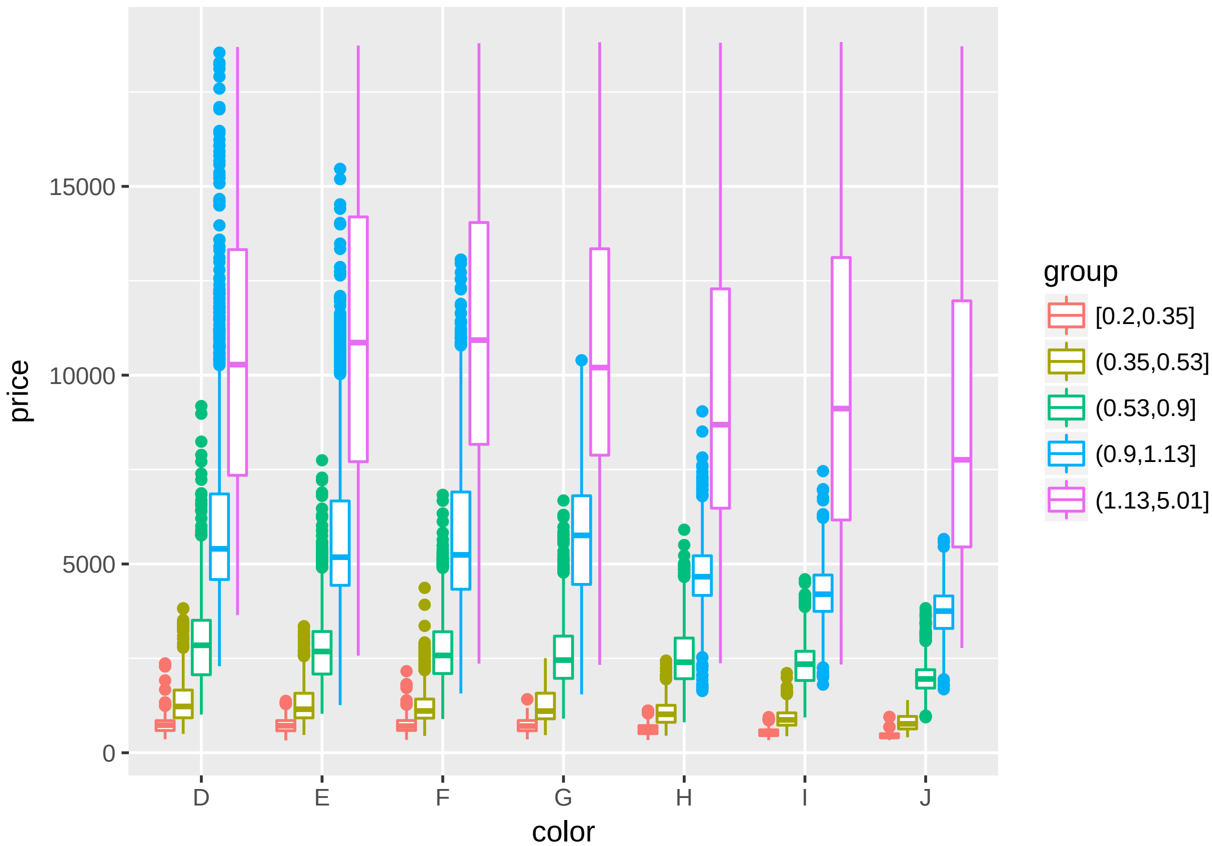
```
data(diamonds)
boxplot(diamonds$price ~ diamonds$color, xlab = "Color", ylab = "Price")
```



By observing the median prices of each color, there seems to be no relation between the price of a diamond and its color. For example, color J which is supposed to be the worst color has higher quartile prices as compared to the best color D. Also, there are no distinct outliers.

(b)

```
other_diamonds <- diamonds
other_diamonds$group <- with(other_diamonds,
                       cut(other_diamonds$carat,
                           breaks = quantile(other_diamonds$carat, prob = seq(0, 1, by = 0.2)),
                           include.lowest = TRUE))
p <- ggplot(other_diamonds, aes(x = color, y = price, color = group))
p + geom_boxplot()
```



Again, there seems to be no relation between the diamond "colors" and "prices". However, there is a directly proportional relationship between "carat" and "price". Also, there is a directly proportional relationship between "carat" and the interquartile range of the prices.

(c)

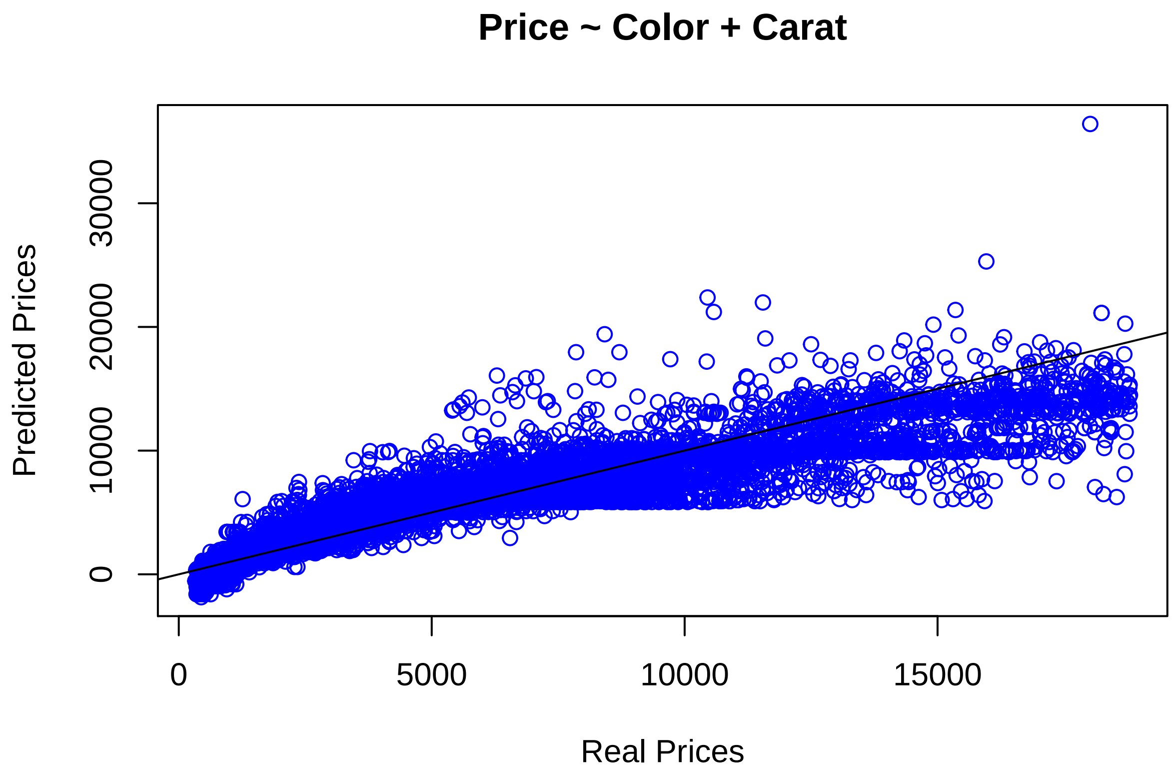
Linear model with predictors "color" and "carat", and response "price"

```
ratio = sample(1:nrow(diamonds), size = 0.7*nrow(diamonds))
d_training <- diamonds[ratio, ]
d_validation <- diamonds[-ratio, ]
d_train <- lm(price ~ color + carat, data = d_training)
summary(d_train)
```

```
>
> Call:
> lm(formula = price ~ color + carat, data = d_training)
```

```
>
> Residuals:
>      Min       1Q   Median       3Q      Max
> -14908.2   -766.1    -72.2    561.6  11625.3
>
> Coefficients:
>             Estimate Std. Error  t value Pr(>|t|)
> (Intercept) -2701.45      16.42 -164.561  < 2e-16 ***
> color.L     -1542.81      26.60  -58.005  < 2e-16 ***
> color.Q      -727.87      24.31  -29.942  < 2e-16 ***
> color.C      -119.38      22.83   -5.229 1.71e-07 ***
> color^4        68.28      20.98    3.255  0.00114 **
> color^5      -158.88      19.81   -8.019 1.10e-15 ***
> color^6      -197.39      17.95  -10.998  < 2e-16 ***
> carat        8071.45      16.76  481.575  < 2e-16 ***
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 1468 on 37750 degrees of freedom
> Multiple R-squared:  0.8643,  Adjusted R-squared:  0.8643
> F-statistic: 3.436e+04 on 7 and 37750 DF,  p-value: < 2.2e-16
```

```r
price_real <- d_validation$price
price_prediction <- predict(d_train, newdata = d_validation)
plot(x = price_real, y = price_prediction, xlab = "Real Prices",
ylab = "Predicted Prices", main = "Price ~ Color + Carat", col = "blue")
abline(a = 0, b = 1)
```



Problem 4

Preliminary steps

```
set.seed(123)
credit <- read_csv("Credit.csv") %>%
  select(-X1)
```

```
> Warning: Missing column names filled in: 'X1' [1]
```

```
> Parsed with column specification:
> cols(
>   X1 = col_integer(),
>   Income = col_double(),
>   Limit = col_integer(),
>   Rating = col_integer(),
>   Cards = col_integer(),
>   Age = col_integer(),
>   Education = col_integer(),
>   Gender = col_character(),
>   Student = col_character(),
>   Married = col_character(),
>   Ethnicity = col_character(),
>   Balance = col_integer()
> )
```

```
# Convert column names to lowercase just for convenience
names(credit) <- stringr::str_to_lower(names(credit))
```

a. Select Training set

```
ratio <- sample(1:nrow(credit), 200)
credit_training <- credit[ratio, ]
credit_validation <- credit[-ratio, ]
```

b. Data Exploration

One variable summary statistics:

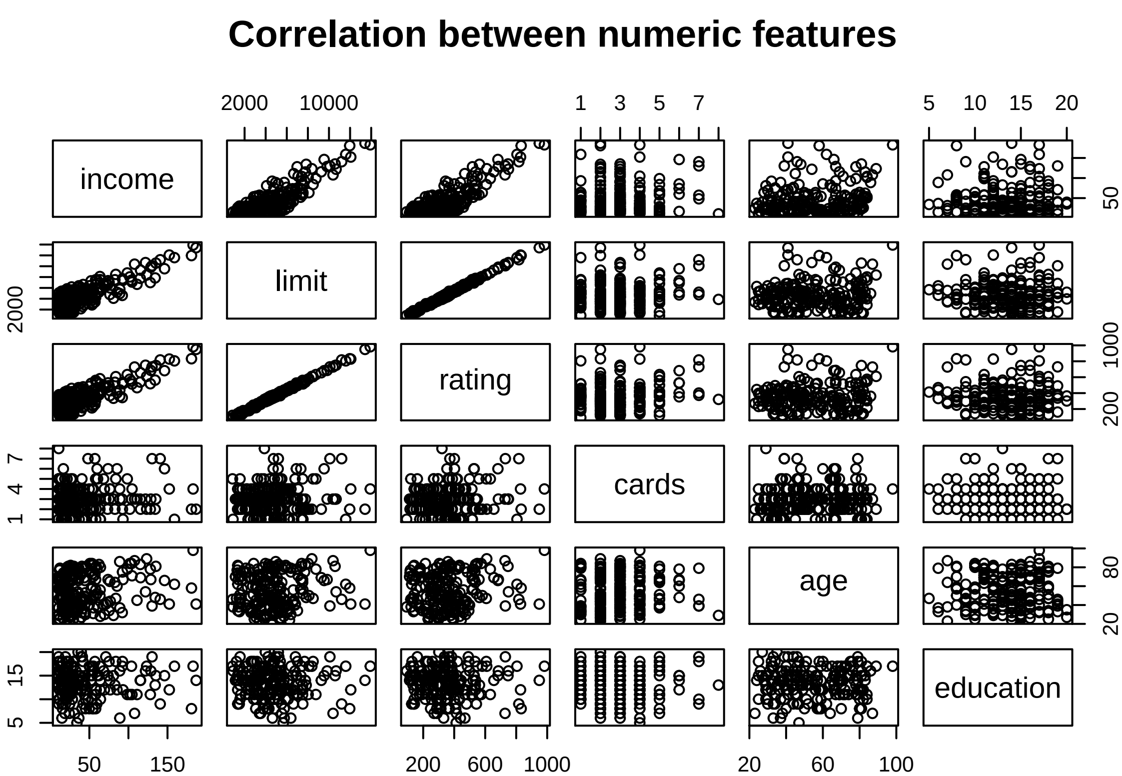
```
summary(credit_training)
```

```
>      income           limit            rating           cards
>  Min.   : 10.35   Min.   :  855   Min.   : 93.0   Min.   :1.00
>  1st Qu.: 20.33   1st Qu.: 3187   1st Qu.:253.8   1st Qu.:2.00
>  Median : 35.23   Median : 4556   Median :344.0   Median :3.00
>  Mean   : 46.47   Mean   : 4813   Mean   :360.2   Mean   :2.98
>  3rd Qu.: 58.04   3rd Qu.: 5912   3rd Qu.:435.5   3rd Qu.:4.00
>  Max.   :186.63   Max.   :13913   Max.   :982.0   Max.   :8.00
>       age          education        gender             student
>  Min.   :23.00   Min.   : 5.00   Length:200         Length:200
>  1st Qu.:40.00   1st Qu.:11.00   Class :character   Class :character
>  Median :54.00   Median :14.00   Mode  :character   Mode  :character
>  Mean   :55.41   Mean   :13.53
>  3rd Qu.:70.00   3rd Qu.:16.00
>  Max.   :98.00   Max.   :20.00
>    married           ethnicity            balance
```

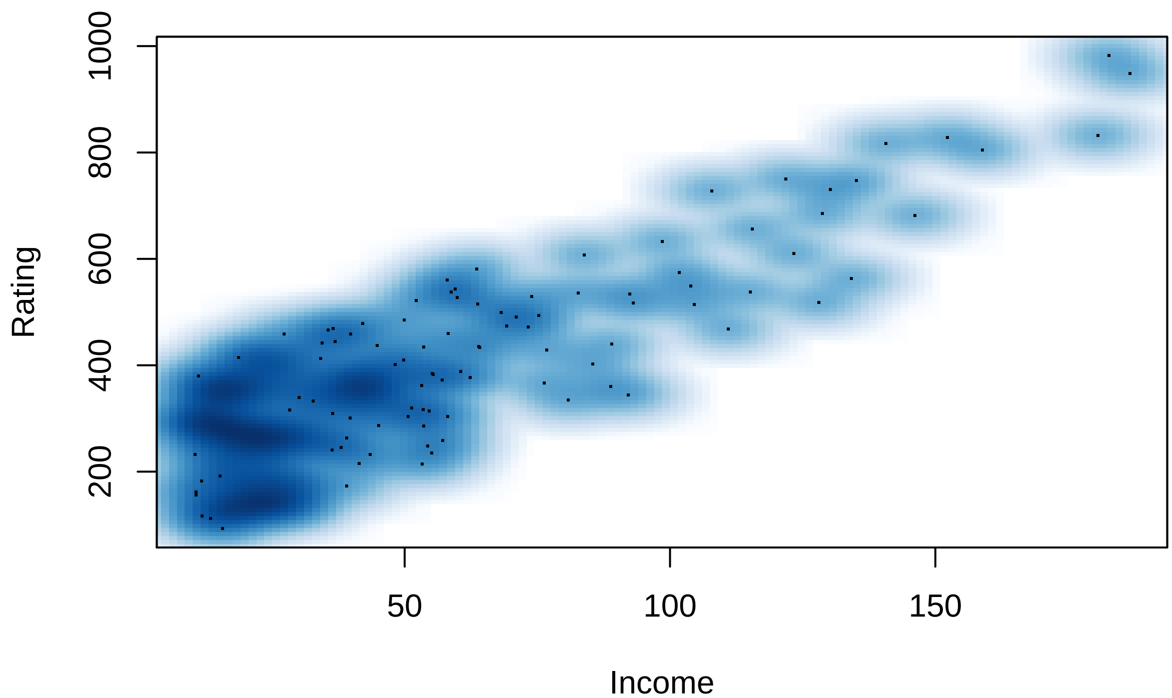
```
>  Length:200          Length:200          Min.   :   0.0
>  Class :character   Class :character   1st Qu.:  79.5
>  Mode  :character   Mode  :character   Median : 459.0
>                                          Mean   : 533.6
>                                          3rd Qu.: 868.5
>                                          Max.   :1999.0
```

Two variable summary statistics:

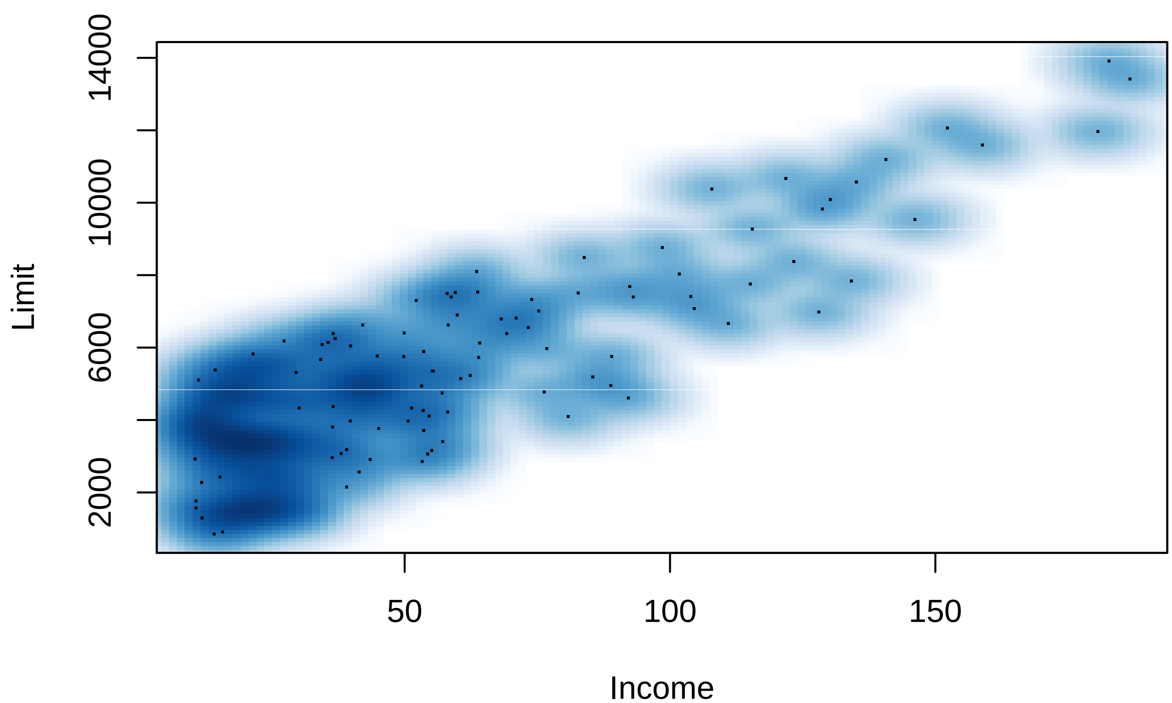
```r
# Observed correlations
# Income: Limit, Rating
# Limit: Income, Rating, Balance
# Rating: Income, Limit, Balance

continuous_training_data <- select(credit_training, income, limit, rating, cards, age, education)
pairs(continuous_training_data, main="Correlation between numeric features")
```




```r
smoothScatter(credit_training$income, credit_training$rating, xlab = "Income", ylab = "Rating")
```




```r
smoothScatter(credit_training$income, credit_training$limit, xlab = "Income", ylab = "Limit")
```




```r
round(cor(continuous_training_data), digits = 2)
```

```
>           income limit rating cards  age education
> income      1.00  0.83   0.83  0.09 0.20      0.01
> limit       0.83  1.00   1.00  0.10 0.11     -0.03
> rating      0.83  1.00   1.00  0.14 0.11     -0.03
> cards       0.09  0.10   0.14  1.00 0.05     -0.05
> age         0.20  0.11   0.11  0.05 1.00      0.00
```

```
> education   0.01 -0.03  -0.03 -0.05 0.00      1.00
```

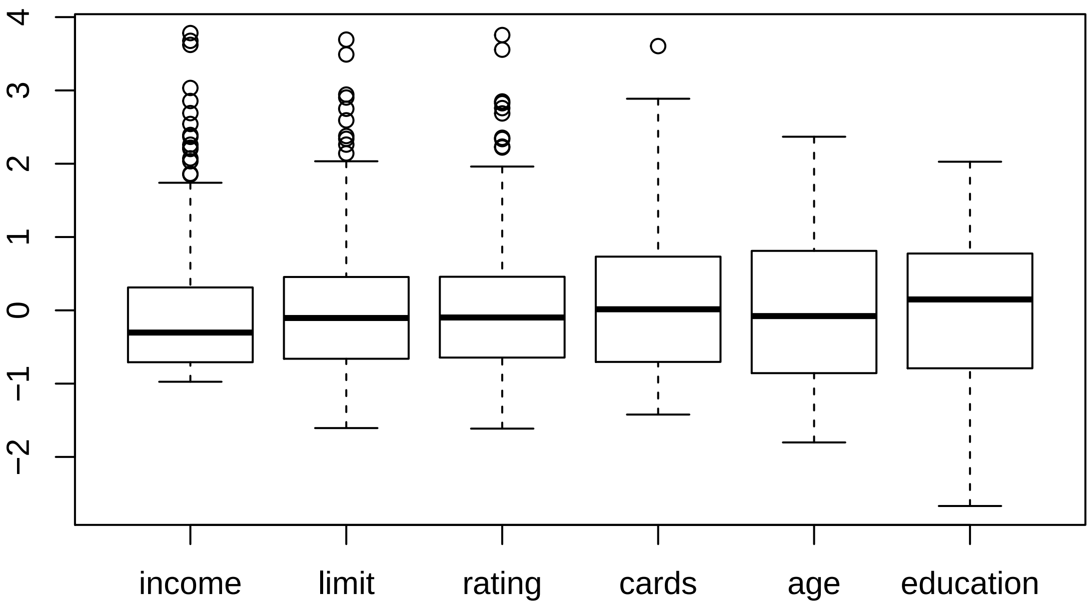
As seen in the above graphs and the correlation table, there is a high correlation between Income and Rating, and, Income and Limit. Correlations can be explored between these features for the model. Since the correlation betweem Rating and Limit is 1.00, from the perspective of the model, they can be used interchangeably, or one of them can be dropped.

```
any(is.na(credit_training))
```

```
> [1] FALSE
```

There are no NA values in the dataset.

```
boxplot(scale(continuous_training_data))
```



From the boxplot above, there are no particular outliers that can be singled out.

c. Assumption of Normality:

```
lm_train <- lm(balance~., data=credit_training)
summary(lm_train)
```

```
>
> Call:
> lm(formula = balance ~ ., data = credit_training)
>
> Residuals:
>     Min      1Q  Median      3Q     Max
> -197.78  -75.64  -16.46   49.95  291.30
>
> Coefficients:
>                      Estimate Std. Error t value Pr(>|t|)
> (Intercept)        -497.05863   51.99710  -9.559  < 2e-16 ***
> income               -7.70263    0.36142 -21.312  < 2e-16 ***
> limit                 0.28497    0.04979   5.724 4.06e-08 ***
> rating               -0.26029    0.74258  -0.351    0.726
> cards                26.59310    6.47478   4.107 5.97e-05 ***
> age                  -0.36364    0.41463  -0.877    0.382
```

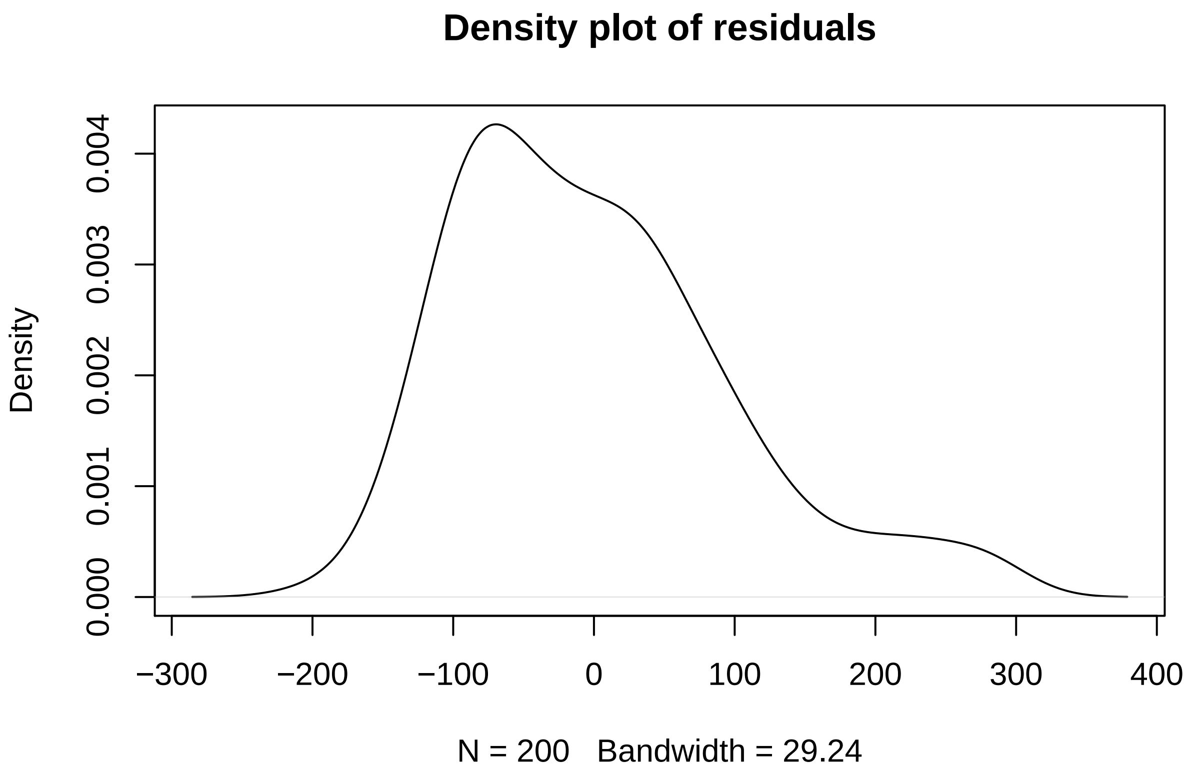
```
> education             -0.79924    2.29377  -0.348    0.728
> genderMale             7.20311   14.64431   0.492    0.623
> studentYes           437.82015   23.99291  18.248  < 2e-16 ***
> marriedYes            10.43592   14.98999   0.696    0.487
> ethnicityAsian        22.10831   20.84572   1.061    0.290
> ethnicityCaucasian     3.85596   17.82009   0.216    0.829
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 101.7 on 188 degrees of freedom
> Multiple R-squared:  0.9574,  Adjusted R-squared:  0.9549
> F-statistic: 383.7 on 11 and 188 DF,  p-value: < 2.2e-16
```

```r
# Evaluate summary(lm_train):
# https://feliperego.github.io/blog/2015/10/23/Interpreting-Model-Output-In-R
```

From the above information, we see that "income", "limit", "cards" and being a student are important features in a linear model that predicts "balance".

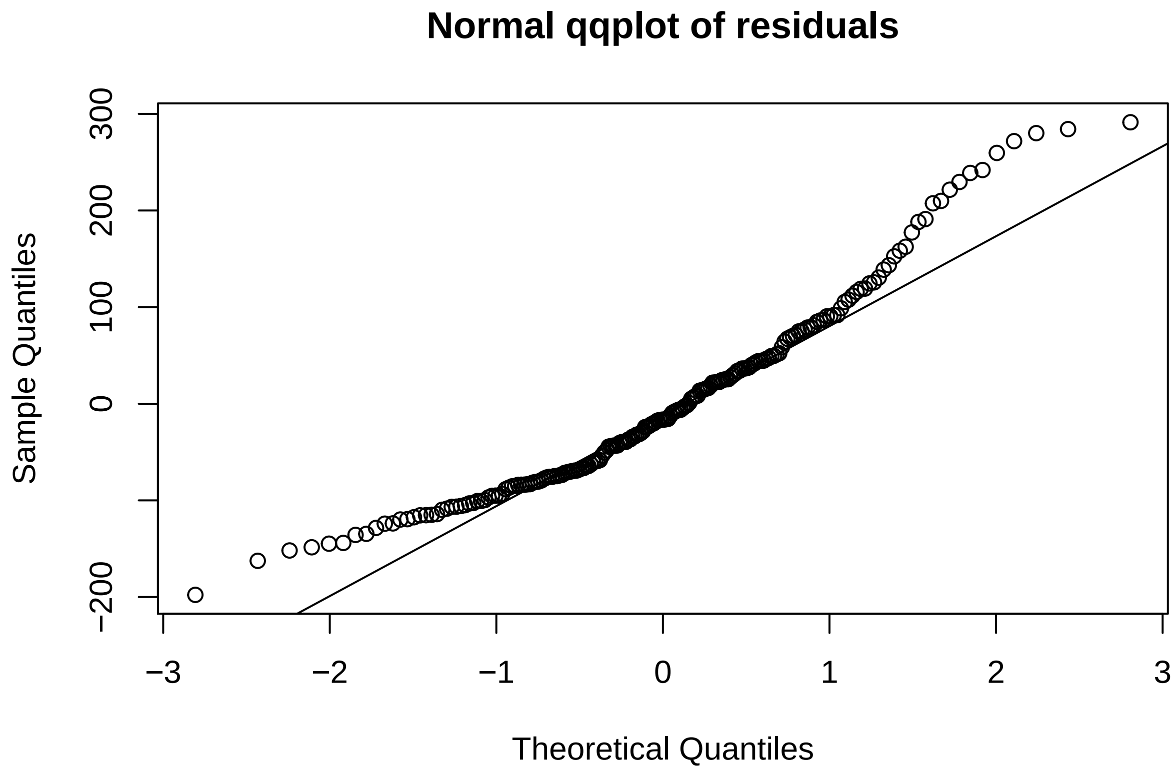
Distribution of residuals:

```r
# Density plot
plot(density(lm_train$residuals), main = "Density plot of residuals")
```



From the above density plot, we see that the first half of the residuals approximate the Normal Distribution. However there are outliers to the right. Let's have a look at the qq plots for a clearer picture.

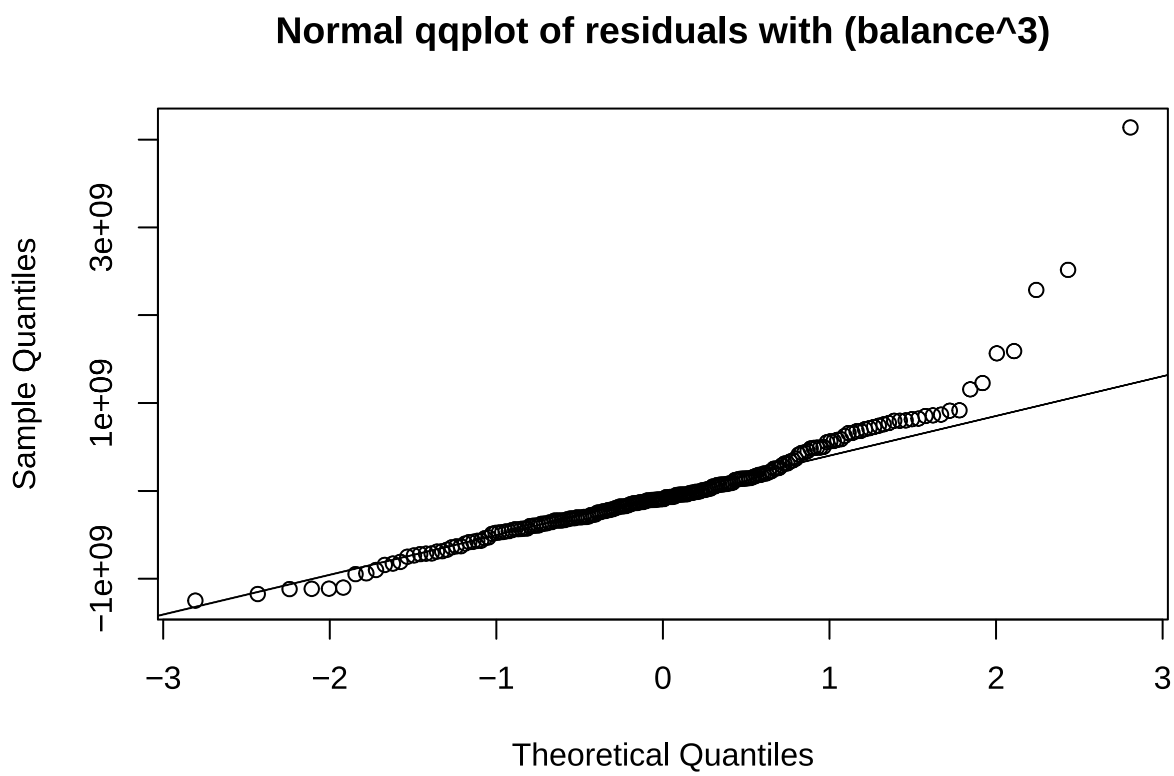
```r
# QQ Plot of residuals
qqnorm(lm_train$residuals, main = "Normal qqplot of residuals")
qqline(lm_train$residuals)
```



The points fall along a line in the middle of the graph, but curve off in the extremities. This means that the training data has more extreme values than would be expected if it truly came from a Normal Distribution.

If *balance* is transformed to $balance^3$, the points to the left are very close to the line, but, the points on the right are further away.

```
lm_train_transformed <- lm((balance^3)~., data=credit_training)
qqnorm(lm_train_transformed$residuals, main = "Normal qqplot of residuals with (balance^3)")
qqline(lm_train_transformed$residuals)
```



Let's check the outlier and see if it makes sense to remove it.

```
i=which(lm_train$residuals==max(lm_train$residuals));
credit_training[i,]
```

```
> # A tibble: 1 x 11
>   income limit rating cards   age education gender student married
>    <dbl> <int>  <int> <int> <int>     <int>  <chr>   <chr>   <chr>
> 1 27.241  1402    128     2    67        15 Female      No     Yes
> # ... with 2 more variables: ethnicity <chr>, balance <int>
```

The above doesn't really stand out so it will remain as a part of the dataset.

d. Variable Selection

Subset Selection

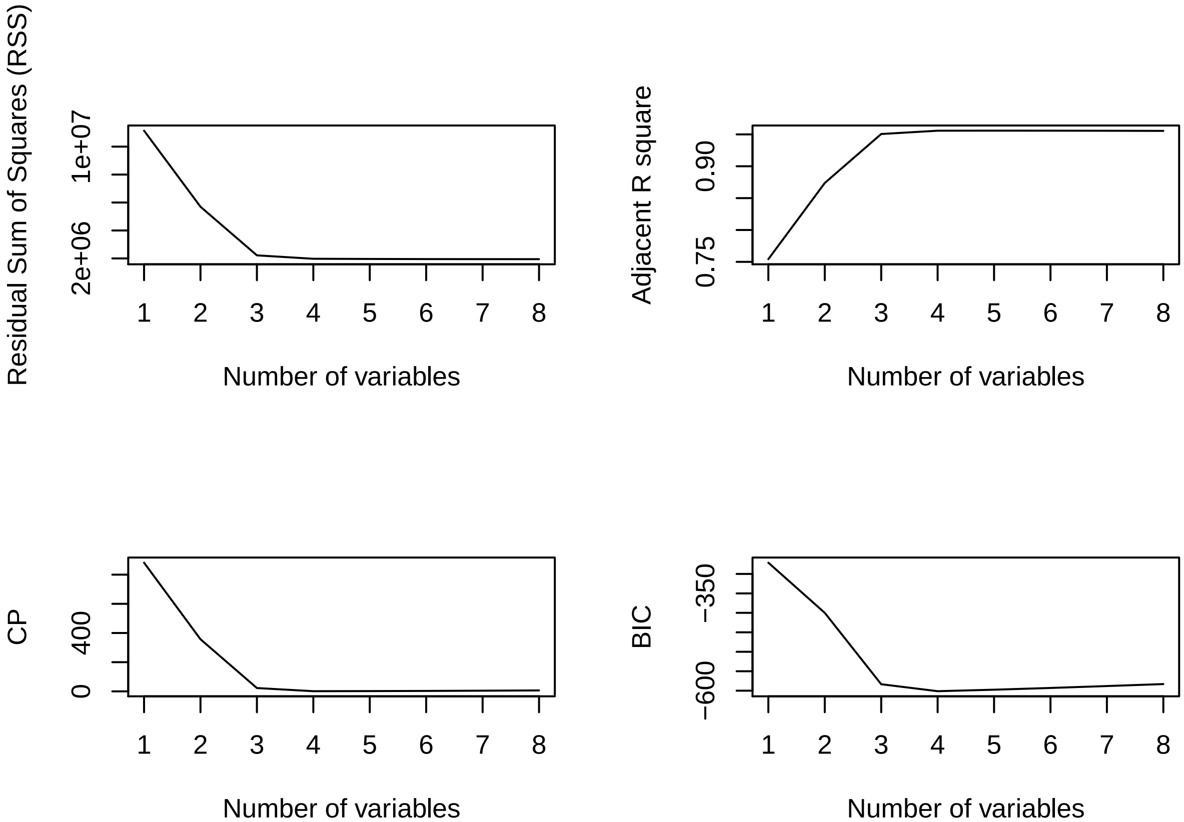
```
regfit.full <- regsubsets(balance~., data = credit_training, really.big = TRUE)
reg.summary <- summary(regfit.full)
reg.summary
```

```
> Subset selection object
> Call: regsubsets.formula(balance ~ ., data = credit_training, really.big = TRUE)
> 11 Variables  (and intercept)
>                    Forced in Forced out
> income                 FALSE      FALSE
> limit                  FALSE      FALSE
> rating                 FALSE      FALSE
> cards                  FALSE      FALSE
> age                    FALSE      FALSE
> education              FALSE      FALSE
```

```
> genderMale              FALSE      FALSE
> studentYes              FALSE      FALSE
> marriedYes              FALSE      FALSE
> ethnicityAsian          FALSE      FALSE
> ethnicityCaucasian      FALSE      FALSE
> 1 subsets of each size up to 8
> Selection Algorithm: exhaustive
>          income limit rating cards age education genderMale studentYes
> 1  ( 1 ) " "    " "   "*"    " "   " " " "       " "        " "
> 2  ( 1 ) "*"    "*"   " "    " "   " " " "       " "        " "
> 3  ( 1 ) "*"    "*"   " "    " "   " " " "       " "        "*"
> 4  ( 1 ) "*"    "*"   " "    "*"   " " " "       " "        "*"
> 5  ( 1 ) "*"    "*"   " "    "*"   " " " "       " "        "*"
> 6  ( 1 ) "*"    "*"   " "    "*"   "*" " "       " "        "*"
> 7  ( 1 ) "*"    "*"   " "    "*"   "*" " "       " "        "*"
> 8  ( 1 ) "*"    "*"   " "    "*"   "*" " "       "*"        "*"
>          marriedYes ethnicityAsian ethnicityCaucasian
> 1  ( 1 ) " "        " "            " "
> 2  ( 1 ) " "        " "            " "
> 3  ( 1 ) " "        " "            " "
> 4  ( 1 ) " "        " "            " "
> 5  ( 1 ) " "        "*"            " "
> 6  ( 1 ) " "        "*"            " "
> 7  ( 1 ) "*"        "*"            " "
> 8  ( 1 ) "*"        "*"            " "
```

Let us estimate the test error by adding a penalty to the training error to account for the bias due to overfitting using four methods: *Adjusted* R^2, *Cp* and *Bayesian information criterion(BIC)*.

```
par(mfrow = c(2,2))
plot(reg.summary$rss, xlab = "Number of variables", ylab = "Residual Sum of Squares (RSS)", type = "l")
plot(reg.summary$adjr2, xlab = "Number of variables", ylab = "Adjacent R square", type = "l")
plot(reg.summary$cp, xlab = "Number of variables", ylab = "CP", type = "l")
plot(reg.summary$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
```



From the above graphs, we see that picking 4 predictors(income, limit, cards, student==Yes) is a good for the model. Let's confirm it

```
which.min(reg.summary$bic)
```

```
> [1] 4
```

Linear model based on 4 predictors

```
subset_select_model <- lm(balance ~ income + limit + cards + student, data = credit_training)
coef(regfit.full, 4)
```

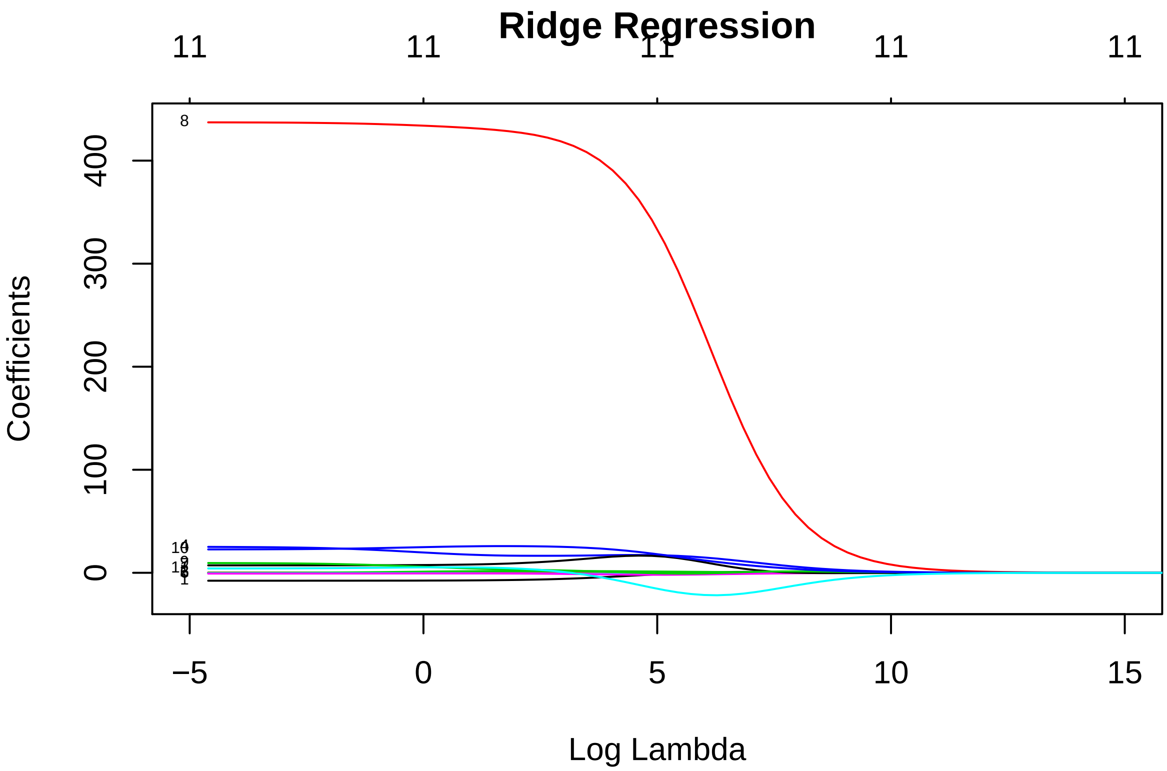
```
>  (Intercept)       income        limit        cards   studentYes
> -517.1692515   -7.7969064    0.2684535   25.2289705  435.7908258
```

The above means that for instance, if the there is a unit increase in "cards", the balance increases by "19.4328604"

e. Variable Selection

Ridge Regression

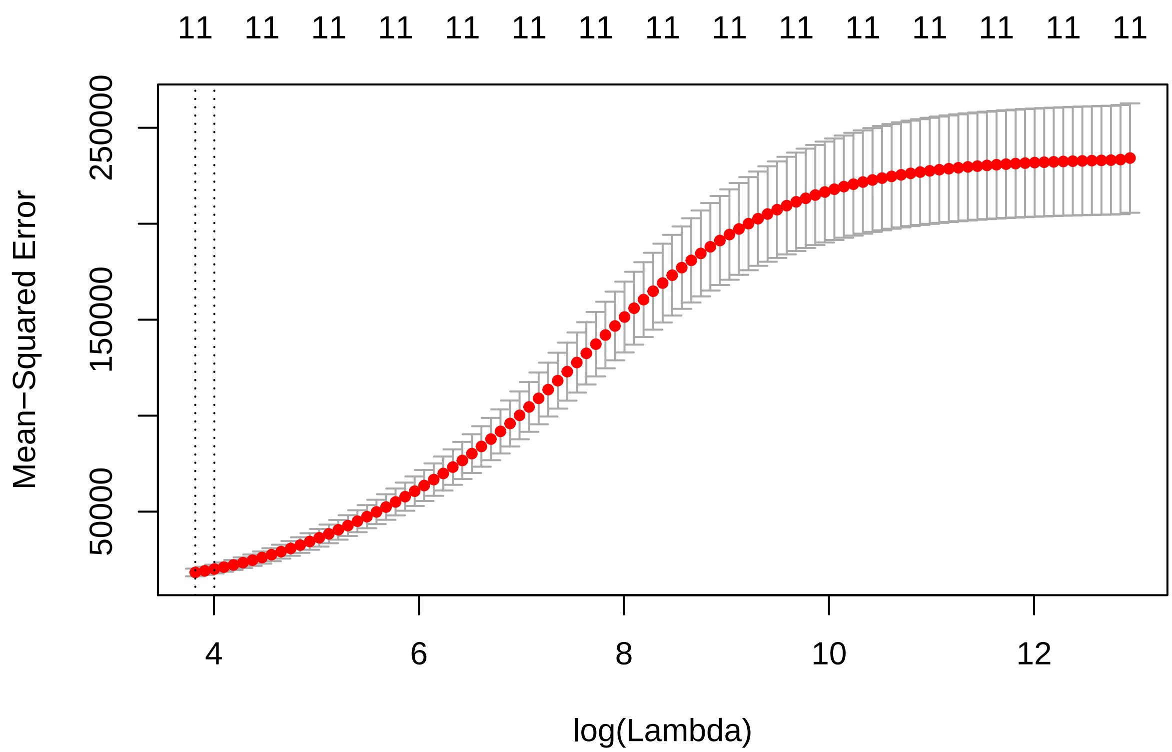
```
x <- model.matrix(balance~., credit_training)[,-1]
y <- credit_training$balance
lambda = 10^seq(10,-2, length = 100)
ridge.mod = glmnet(x, y, alpha=0, lambda = lambda)
plot(ridge.mod, main = "Ridge Regression",label = TRUE, xvar = "lambda", xlim = c(-5,15))
```



Above, I have chosen λ values that range from 10^{10} to 10^{-2}. This covers the $\lambda = 0$ case as well, where the coefficients are the same as the ones in linear regression.

Insead of choosing the λ value arbitrarily, let's perform 10 fold cross validation to pick the best λ that minimizes the Mean Squared Error.

```
cv.out <- cv.glmnet(x,y, alpha = 0)
plot(cv.out)
```



We can see above that regardless of the value of λ, the the number of predictors chosen is always 11. This is because Ridge Regression does not perform variable selection unlike Lasso Regression.

```
bestlam.ridge = cv.out$lambda.min
bestlam.ridge
```

```
> [1] 45.5346
```

```
log(bestlam.ridge)
```

```
> [1] 3.818472
```

According to Ridge Regression, the λ with the least Mean Squared Error is 43.63804. Let's use this value to fit a regression model

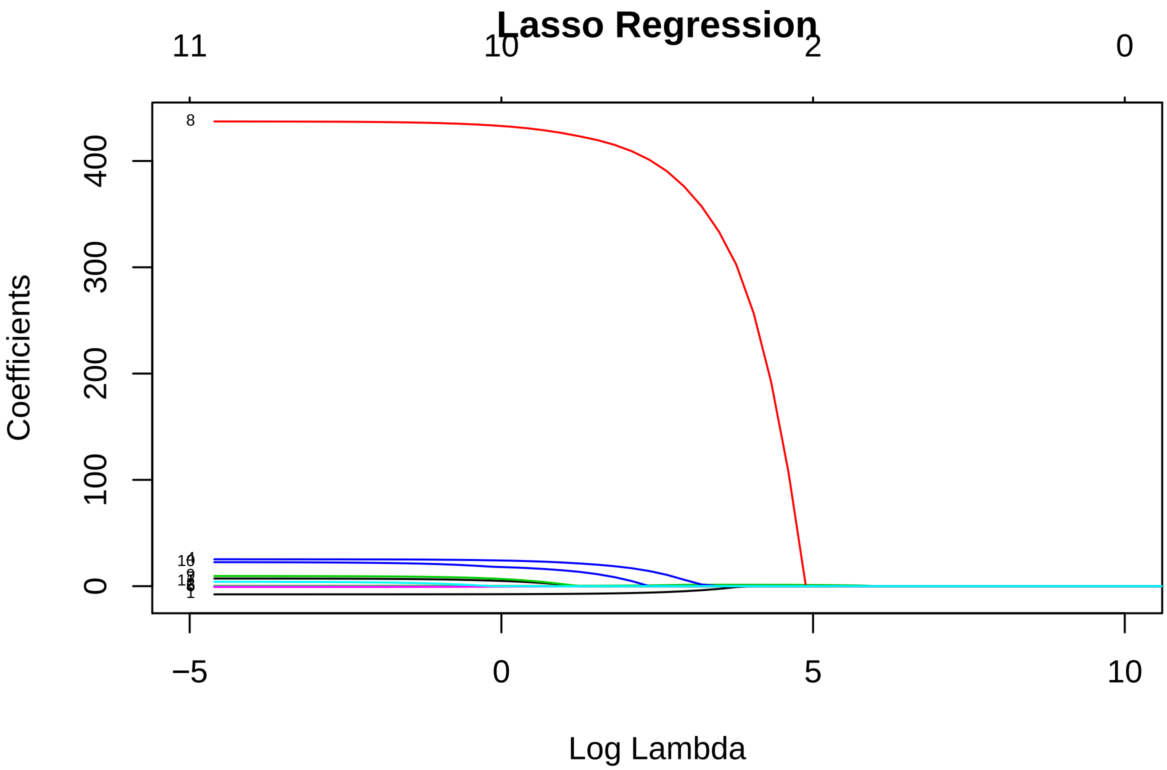
```
ridge.mode <- glmnet(x, y, alpha=0, lambda = bestlam.ridge)
predict(ridge.mode, s = bestlam.ridge, type = "coefficients")
```

```
> 12 x 1 sparse Matrix of class "dgCMatrix"
>                                 1
> (Intercept)        -392.7970344
> income               -4.3497196
> limit                 0.1104793
> rating                1.5692634
> cards                16.9840834
> age                  -0.9350089
> education            -1.4976213
> genderMale           14.9052485
> studentYes          398.8964427
> marriedYes            1.2281071
> ethnicityAsian       23.4091999
> ethnicityCaucasian   -4.5438835
```

Lasso Regression

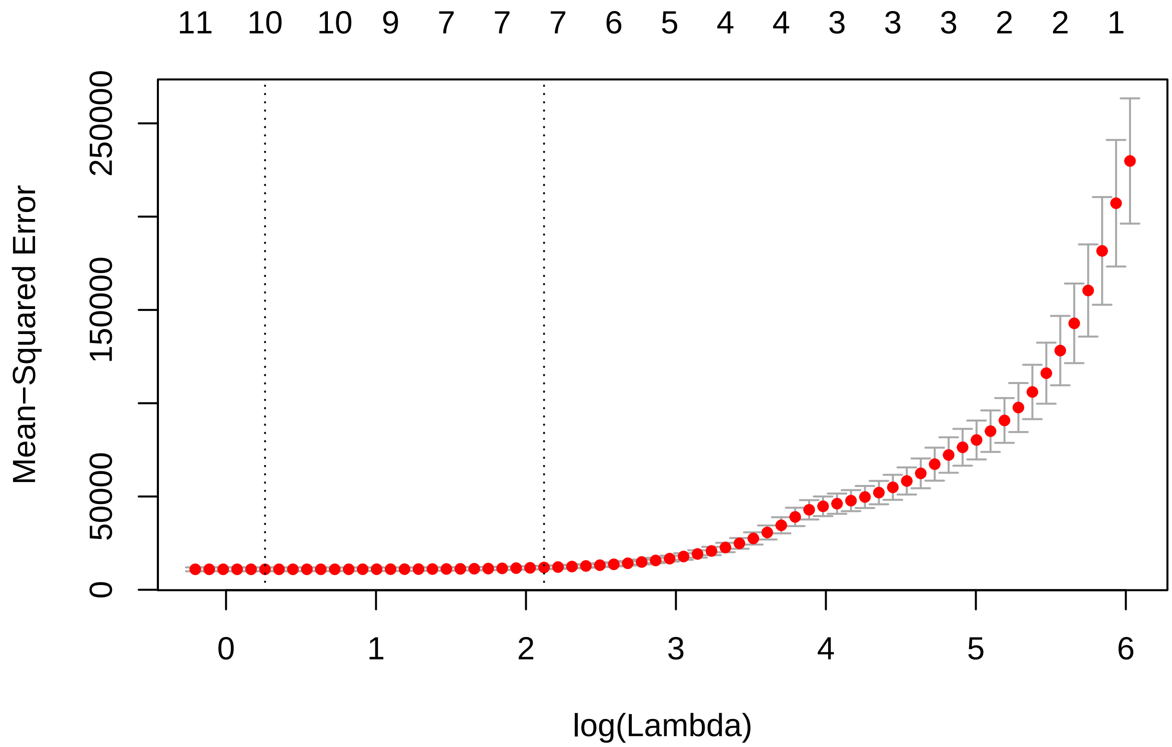
Performing Lasso Regression with the same range of λ values,

```
lasso.mod <- glmnet(x,y, alpha = 1, lambda = lambda)
plot(lasso.mod, main = "Lasso Regression", label = TRUE, xvar = "lambda", xlim = c(-5,10))
```



Using cross validation to get the best λ,

```
cv.out <- cv.glmnet(x,y,alpha = 1)
plot(cv.out)
```



Lasso Regression does variable selection unlike Ridge Regression. The two vertical lines above show the range of the number of predictors that minimize the Mean Square Error. In this case, 9 or 10.

```
bestlam.lasso <- cv.out$lambda.min
bestlam.lasso
```

```
> [1] 1.296842
```

```
log(bestlam.lasso)
```

```
> [1] 0.2599318
```

According to the above, the best λ value possible is 0.7805316. Let's use this to fit a regression model.

```
lasso.mode <- glmnet(x, y, alpha=1, lambda = bestlam.lasso)
predict(lasso.mode, s = bestlam.lasso, type = "coefficients")[1:12,]
```

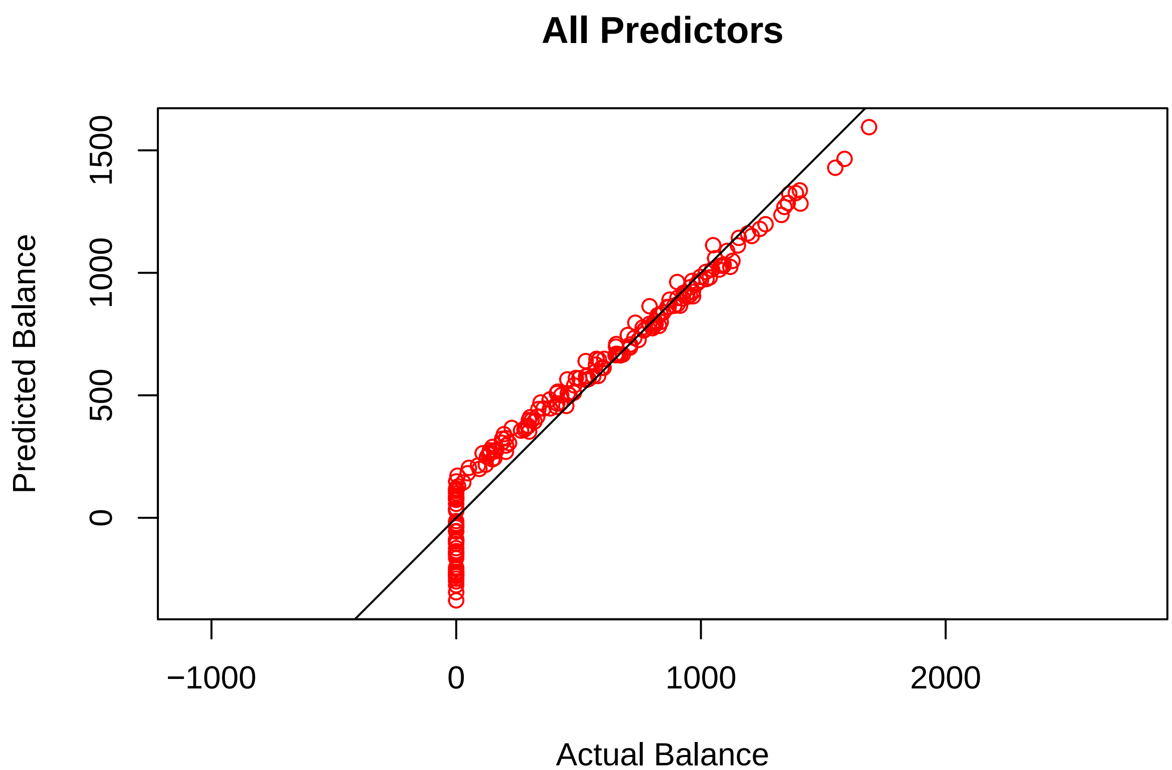
```
>       (Intercept)            income             limit
>      -494.9115716        -7.5053094         0.2645393
>            rating             cards               age
>         0.0000000        24.3446676        -0.3405924
>         education        genderMale        studentYes
>        -0.3471740         4.3090127       431.9691521
>        marriedYes    ethnicityAsian ethnicityCaucasian
>         6.2706000        17.3048143         0.0000000
```

Ignoring the predictors with value 0, we see that Lasso Regression has chosen 10 predictors. They are "income", "limit", "rating", "cards", "age", "education", "gender==Male", "student=yes", "married=yes", "ethnicity=Asian", "ethnicity=Caucasian".

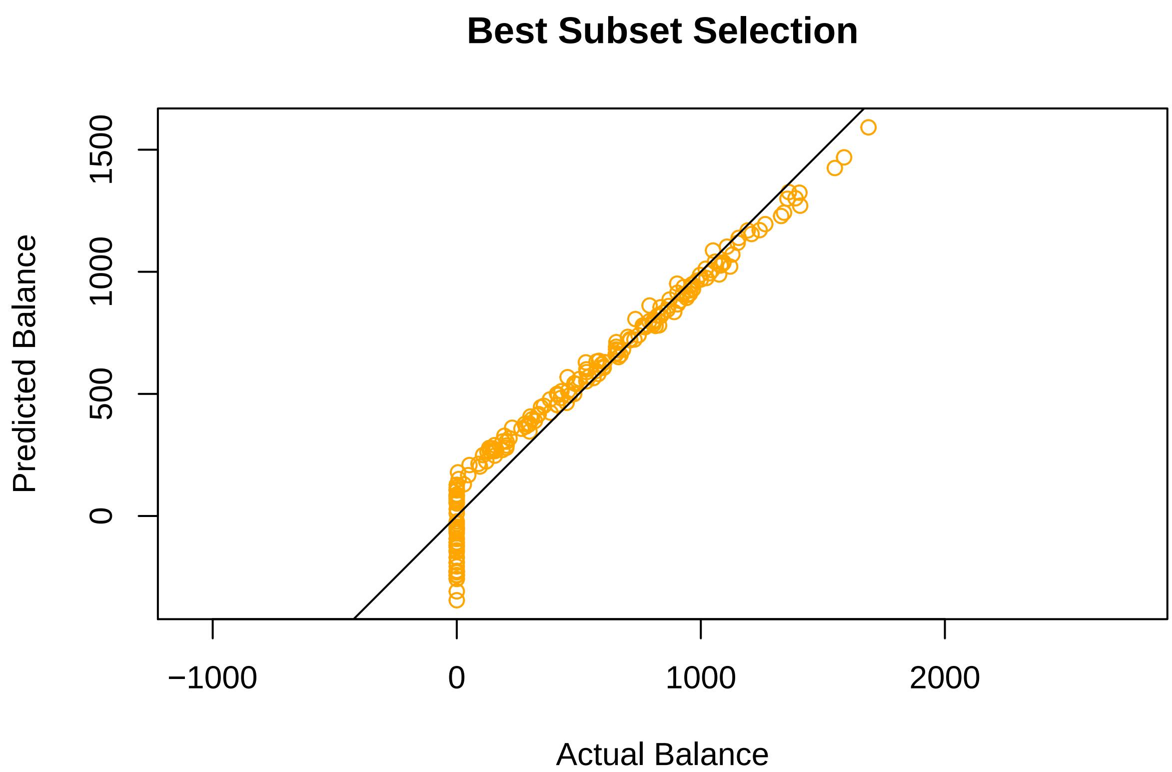
f & e. Performance Evaluation + Interpretation

We have four models: * Regression with all predictors * Regression with predictors from subset selection * Ridge Regression * Lasso Regression

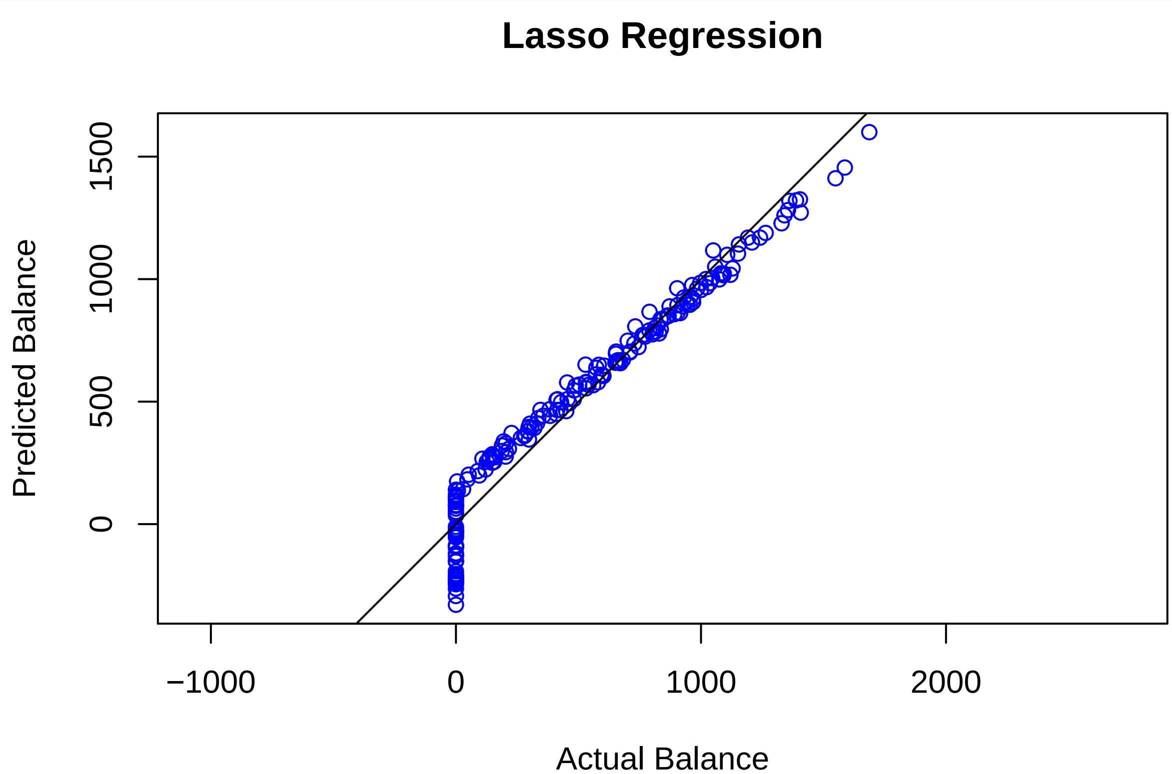
```r
# Variables to plot
real_balance <- credit_validation$balance
reg_all_pred <- predict(lm_train, newdata = credit_validation)
subset_pred <- predict(subset_select_model, newdata = credit_validation)

newx = data.matrix(model.matrix(balance~., credit_validation)[, -1])
lasso_pred <- predict(lasso.mode, newx = newx)
ridge_pred <- predict(ridge.mode, newx = newx)

# Visualizing the four models
par(mfrow = c(1, 1))
plot(x = real_balance, y = reg_all_pred, xlab = "Actual Balance",
ylab = "Predicted Balance", main = "All Predictors", col = "red", asp=1)
abline(a = 0, b = 1)
```



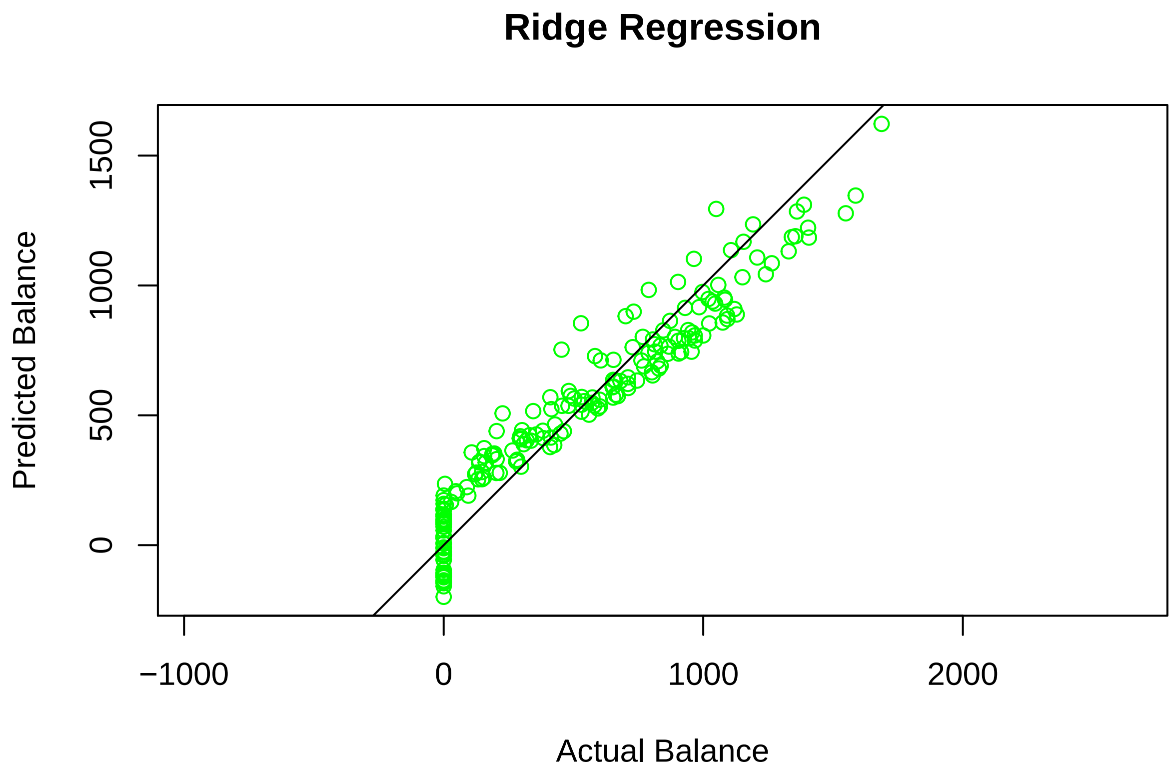

```r
plot(x = real_balance, y = subset_pred, xlab = "Actual Balance",
ylab = "Predicted Balance", main = "Best Subset Selection", col = "orange", asp=1)
abline(a = 0, b = 1)
```




```
plot(x = real_balance, y = lasso_pred, xlab = "Actual Balance",
ylab = "Predicted Balance", main = "Lasso Regression", col = "blue", asp=1)
abline(a = 0, b = 1)
```



```r
plot(x = real_balance, y = ridge_pred, xlab = "Actual Balance",
ylab = "Predicted Balance", main = "Ridge Regression", col = "green", asp=1)
abline(a = 0, b = 1)
```




```r
# Mean Squared Errors of the above four models
error_reg_all_pred <- mean((reg_all_pred - real_balance)^2)
error_subset_pred <- mean((subset_pred - real_balance)^2)
error_lasso_pred <- mean((lasso_pred - real_balance)^2)
error_ridge_pred <- mean((ridge_pred - real_balance)^2)
error_reg_all_pred
```

```
> [1] 9932.756
```

```r
error_subset_pred
```

```
> [1] 9781.906
```

```r
error_lasso_pred
```

```
> [1] 9772.657
```

```r
error_ridge_pred
```

```
> [1] 15932.86
```

```r
order(c(error_reg_all_pred,
          error_subset_pred,
          error_lasso_pred,
          error_ridge_pred),
      decreasing = F)
```

```
> [1] 3 2 1 4
```

Lasso Regression has the lowest MSE and hence is the best choice.