Practical Work 1 - Regularised Regression Methods

Adib Habbou - Alae Khidour

03-10-2022

IV. Application: GAFAM or BATX data set

Data Set Exploration

The data set provided as a csv file was extracted from data about Facebook monthly active users found here: https://www.kaggle.com/datasets/william153/users-of-different-social-medias

```
facebook_data <- read.csv(file = "facebook_dataset.csv")
dim(facebook_data)
```

```
[1] 25  2
```

Our data set is composed of 2 columns respectively for the quarter and the number of users and contains 56 rows which represents 56 different quarters from the third quarter of 2008 to the second quarter of 2022.

```
head(facebook_data)
```

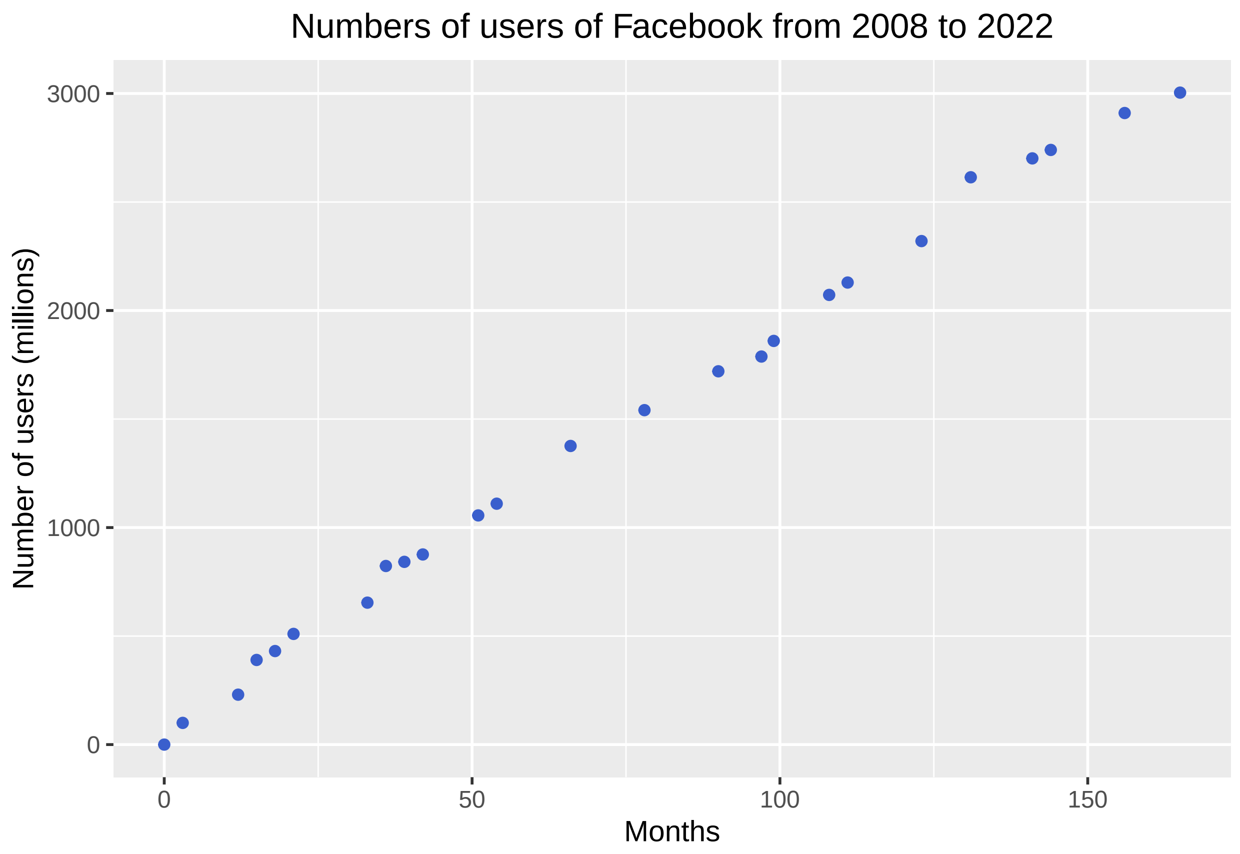
```
  months users
1      0     0
2      3   100
3     12   230
4     15   390
5     18   431
6     21   510
```

```
tail(facebook_data)
```

```
   months users
20    123  2320
21    131  2614
22    141  2701
23    144  2740
24    156  2910
25    165  3004
```

First of all, let's visualize our data with ggplot2 in order to see if we can recognize a specific model.

```r
ggplot(data = facebook_data, mapping = aes(x = months, y = users)) +
  geom_point(size = 1.5, color = "royalblue3") +
  labs(title = "Numbers of users of Facebook from 2008 to 2022",
       x = "Months", y = "Number of users (millions)") +
  theme(plot.title = element_text(hjust = 0.5))
```



From the figure above, we can assume that our data follows a linear model and not an exponential one, so no need to apply the logarithm to our data points.

Regression Model Application

In order to compute the values of our coefficients using the data set we are going to use the OLS estimated values which minimizes the quadratic error on the facebook data set.

$$E(\beta) = \sum_{i=1}^{n} \left(y_i - (\beta_0 + \beta_1 x_{i1})\right)^2$$

$$\hat{\beta} = \arg \min_{\beta \in \mathbb{R}^p} E(\beta)$$

$$\hat{\beta}_{OLS} = \left(X^\top X\right)^{-1} X^\top Y$$

```r
facebook_model <- lm(formula = users ~ months, data = facebook_data)
summary(facebook_model)
```

```

Call:
lm(formula = users ~ months, data = facebook_data)

Residuals:
    Min      1Q  Median      3Q     Max
-105.57  -40.05   11.44   31.66  126.61

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  90.1665    19.9843   4.512 0.000157 ***
months       18.2994     0.2243  81.601  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 56.79 on 23 degrees of freedom
Multiple R-squared:  0.9966,    Adjusted R-squared:  0.9964
F-statistic:  6659 on 1 and 23 DF,  p-value: < 2.2e-16
```

From the summary of our regression we can tell that:

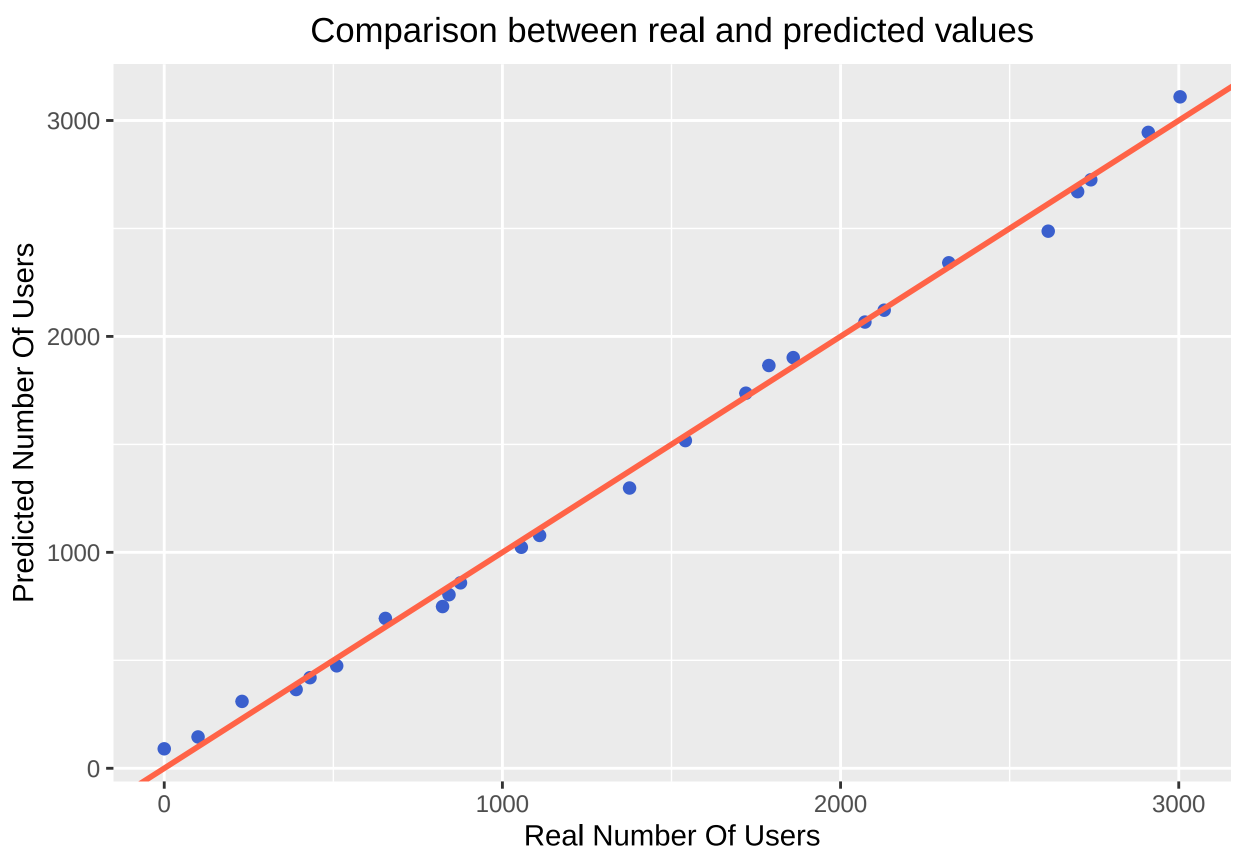
- The residuals represents the distance from the data to the fitted line doesn't exceed 70 and -130;
- The intercept of our line is equal to 99.7726 and the slope of our line is equal to 17.9547;
- The standard error and the t-value are provided to show how the p-values were calculated;
- The p-values tells us if our model is statistically significant in our case there are very low ($< 2.2.10^{-16}$);
- The R^2 tells us that the number of months can explain 99.73% of the variation in the number of users;
- The first degree of freedom corresponds to $p-1$ with $p=2$ the number of variables of the model;
- The second degree of freedom corresponds to $n-p$ with $n=25$ the number of data points.

Study of the Adjustment

Now let's study the bivariate distribution composed of our data and the predicted values.

```r
predict_users <- function (model, param)
{
  return (coef(model)[1] + coef(model)[2] * param)
}
```

```r
bivariate_data <- as.data.frame(cbind(facebook_data$users,
                                      predict_users(facebook_model, facebook_data$months)))
ggplot(data = bivariate_data, mapping = aes(x = V1, y = V2)) +
  geom_point(size = 1.7, color = "royalblue3") +
  geom_abline(size = 1, color = "tomato1") +
  labs(title = "Comparison between real and predicted values",
       x = "Real Number Of Users", y = "Predicted Number Of Users") +
  theme(plot.title = element_text(hjust = 0.5))
```



By looking at the above figure, we can tell that the predicted values are very close to the real ones.

Study of the Normality

```
shapiro.test(facebook_data$users)
```

```
    Shapiro-Wilk normality test

data:  facebook_data$users
W = 0.94363, p-value = 0.1795
```

The Shapiro–Wilk test tests the null hypothesis that a sample $x_1, ..., x_n$ is normally distributed.

The statistic test is:

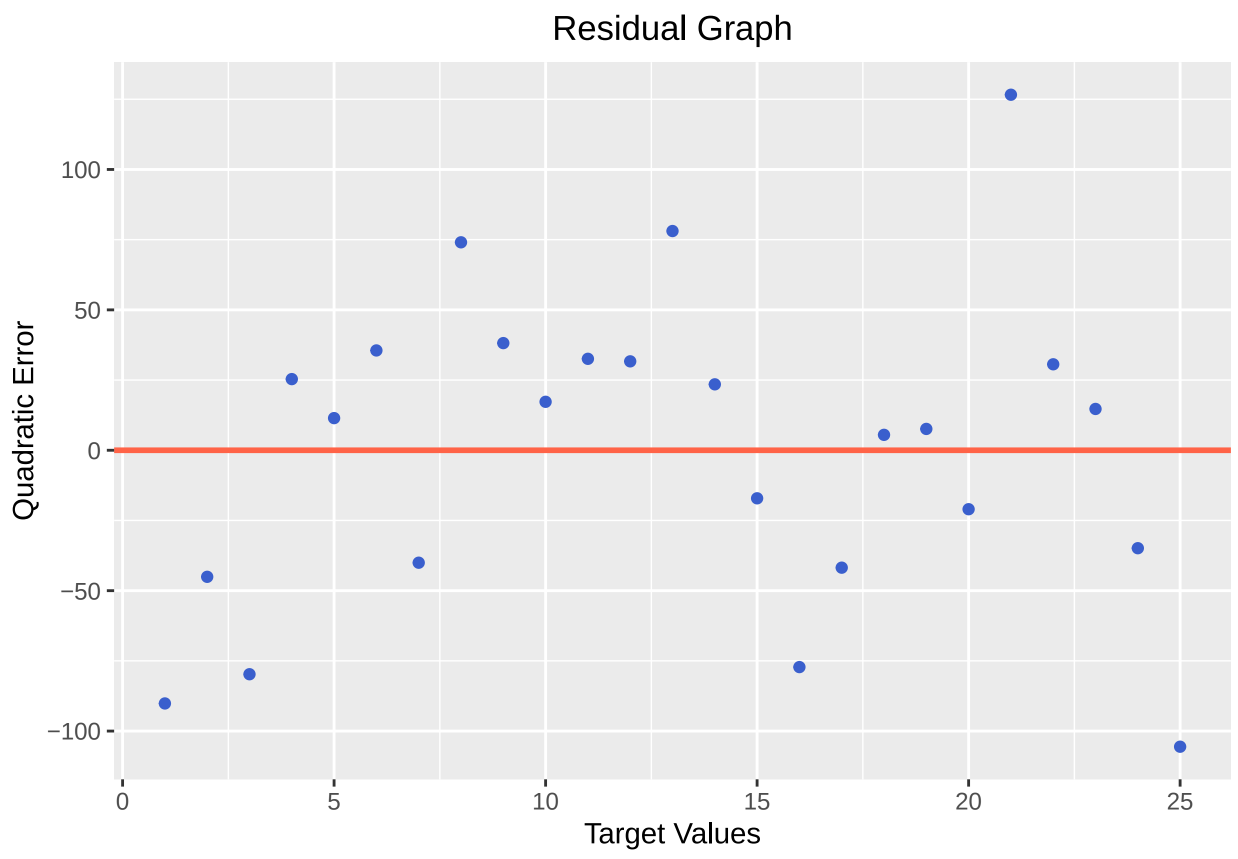
$$W = \frac{(\sum_{i=1}^{n} a_i x_{(i)})^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

Where $x_{(i)}$ is the ith-smallest number in the sample and $\bar{x}$ is the mean of the sample.

Here we can see that the p-value is equal to 0.1795 which is lower that 2% then we can reject the null hypothesis with an alpha level of 2%. Therefore, we can assume that our data is not normally distributed.

Study of the Residuals

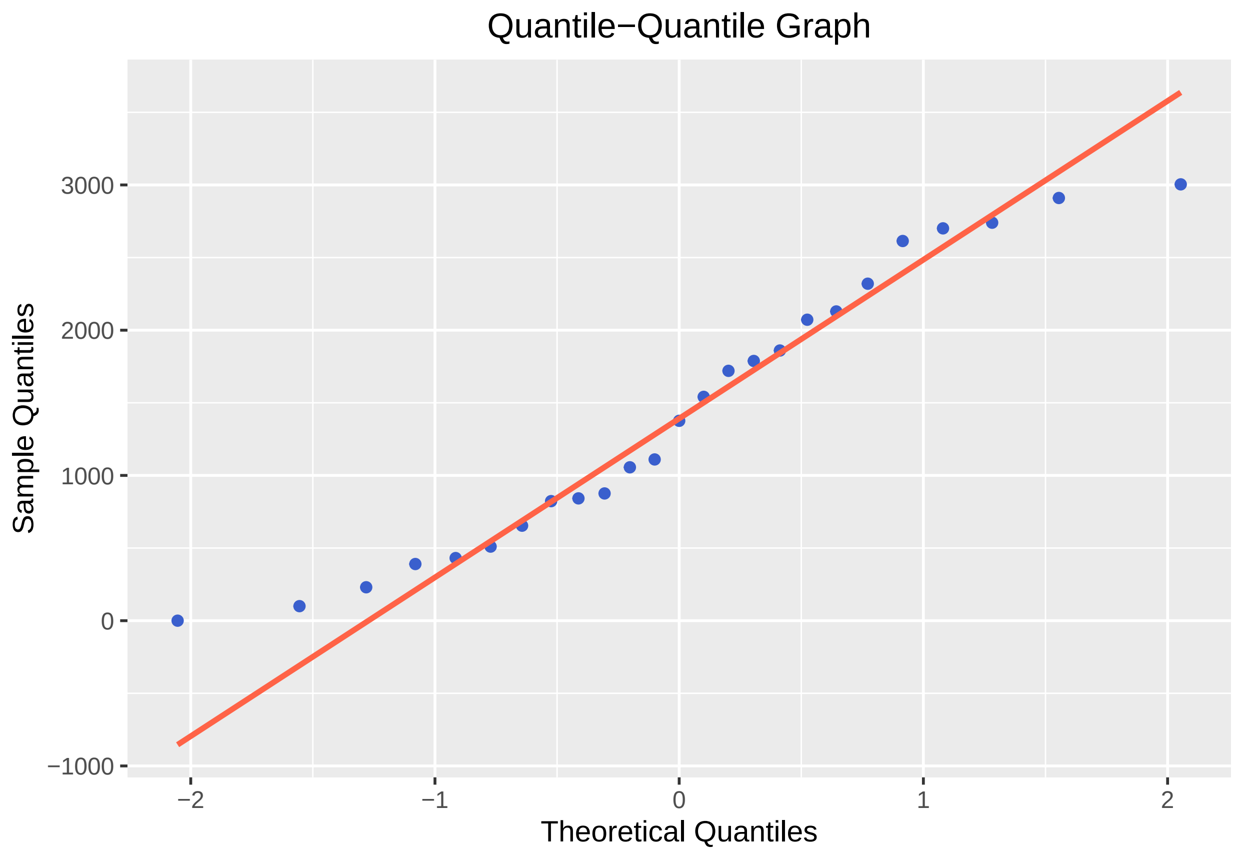
```r
residuals <- as.data.frame(facebook_model$residuals)
ggplot(data = residuals, mapping = aes(x = seq(1, nrow(residuals)), y = facebook_model$residuals)) +
  geom_point(size = 1.5, color = "royalblue3") +
  geom_hline(yintercept = 0, size = 1, color = "tomato1") +
  labs(title = "Residual Graph", x = "Target Values", y = "Quadratic Error") +
  theme(plot.title = element_text(hjust = 0.5))
```



The distribution of points is random, so there is no more information to capture from the residuals.

Study of the Quantiles

```r
ggplot(data = facebook_data, mapping = aes(sample = users)) +
  stat_qq(size = 1.5, color = "royalblue3") +
  stat_qq_line(size = 1, color = "tomato1") +
  labs(title = "Quantile-Quantile Graph", x = "Theoretical Quantiles", y = "Sample Quantiles") +
  theme(plot.title = element_text(hjust = 0.5))
```



Here we are comparing two probability distributions by plotting their quantiles against each other.

We can say that the two distributions are similar if they fit the first bisector (y = x).

Let's compare the theoretical estimated values with the real target values: they fit the y = x line so we can assume that the two distributions are similar.

Then, the linear model is the right model to use in this situation.

Random Partitionning

Now, let's split our data set into training and testing data set in order to study the predictive power of our model which refers to it's ability to compute good predictions for new data that he has never seen before.

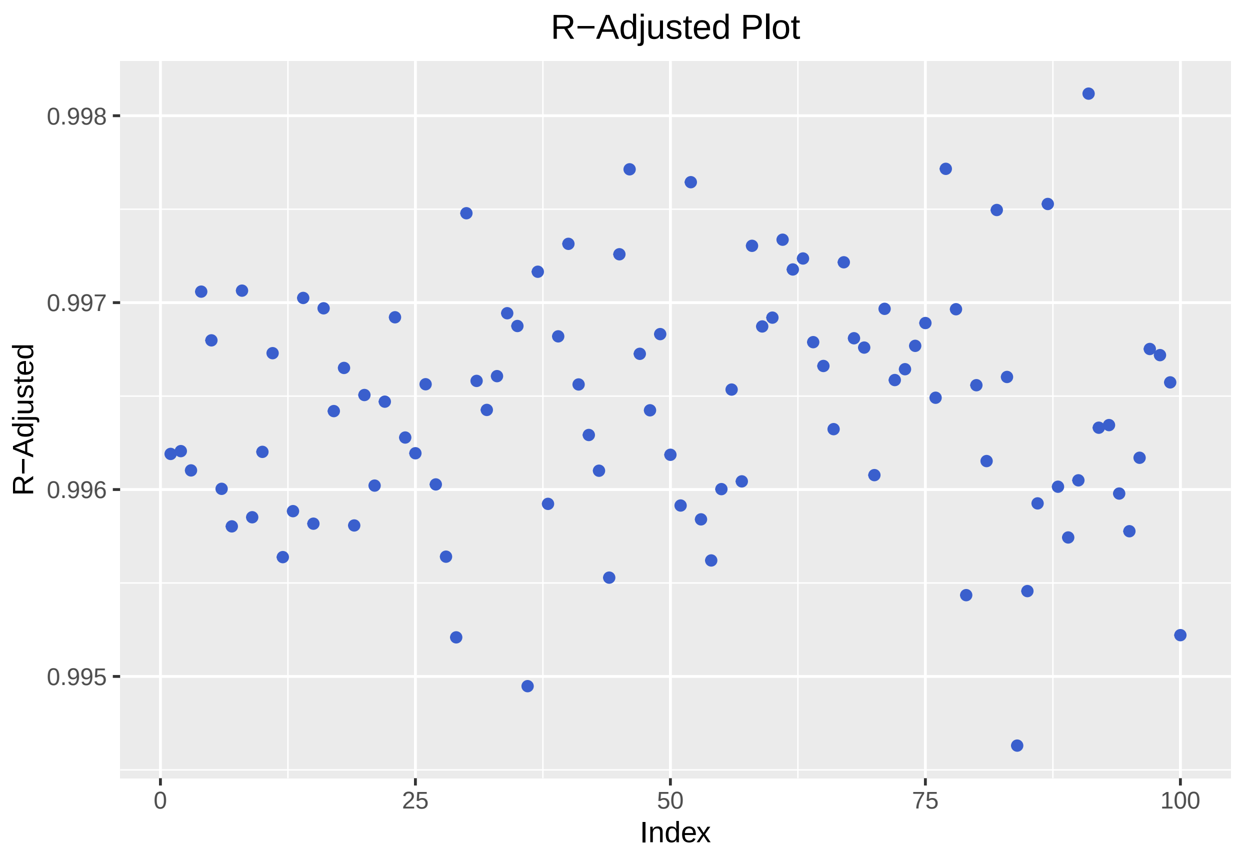
We are going to repeat the same computation 100 times, in order to have significant values for different training and testing data sets, where the training and the testing data set represents respectively 75% and 25% of the initial data set.

At each iteration we are going to compute the RMSD for the train and test data set.

```r
# vector to store R-Squared Adjusted
R_Squared_Adj <- vector(length = 100)
# vector to store RMSD computed with train data set
RMSD_Train <- vector(length = 100)
# vector to store RMSD computed with test data set
RMSD_Test <- vector(length = 100)
```

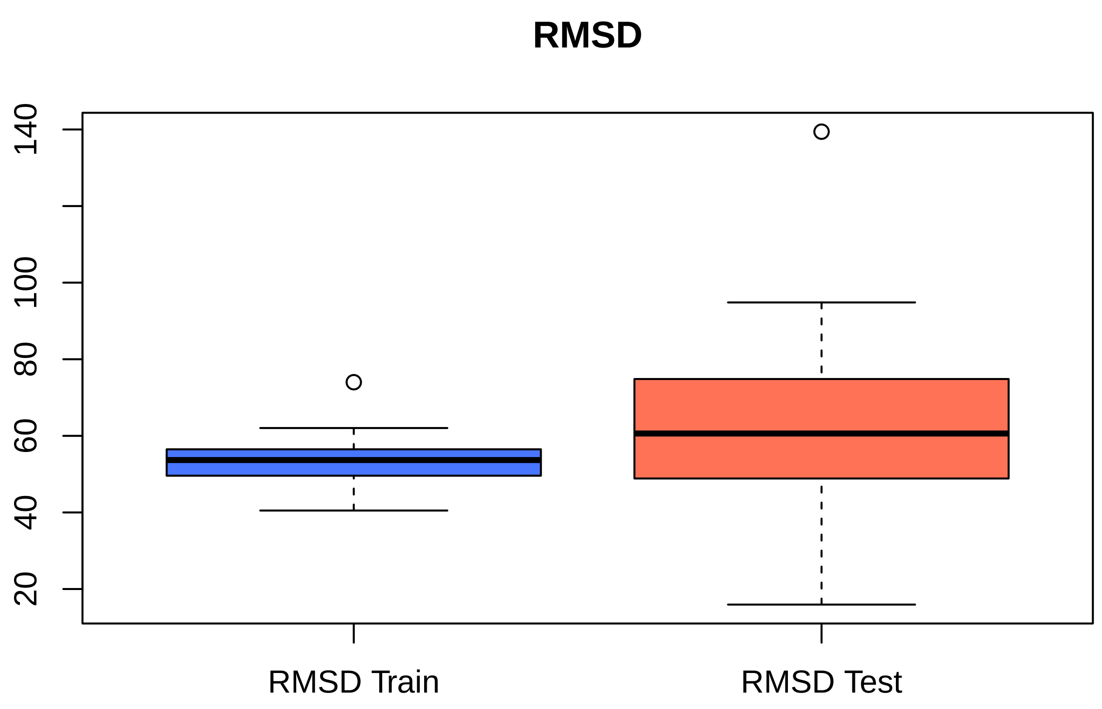
```r
for (i in 1:100)
{
  # train and test split
  sample <- sample(c(TRUE, FALSE), nrow(facebook_data), replace = TRUE, prob = c(0.75, 0.25))
  Train <- facebook_data[sample, ]
  Test <- facebook_data[!sample, ]
  # model training
  model <- lm(formula = users ~ months, data = Train)
  # compute R Adjusted
  R_Squared_Adj[i] <- summary(model)["adj.r.squared"]$adj.r.squared
  # compute RMSD Train
  Sum_Of_Square_Train <- sum((Train$users - predict(model, newdata = Train))^2)
  RMSD_Train[i] <- sqrt(Sum_Of_Square_Train / length(Train$users))
  # compute RMSD Test
  Sum_Of_Square_Test <- sum((Test$users - predict(model, newdata = Test))^2)
  RMSD_Test[i] <- sqrt(Sum_Of_Square_Test / length(Test$users))
}
```

```r
ggplot(data.frame(x = 1:length(R_Squared_Adj), y = R_Squared_Adj), aes(x, y)) +
  geom_point(size = 1.5, color = "royalblue3") +
  labs(title = "R-Adjusted Plot", x = "Index", y = "R-Adjusted") +
  theme(plot.title = element_text(hjust = 0.5))
```



After plotting the R-Adjusted we observe that all the values are in between 0.994 and 0.998.

```
boxplot(RMSD_Train, RMSD_Test,
        main = "RMSD", names = c("RMSD Train", "RMSD Test"),
        col = c("royalblue1", "coral1"))
```



We notice that the RMSD computed on the test data set is larger than the RMSD computed on the train data set. It's because the coefficients are estimated based on the train data set which means they perfectly fit those data compared to the test data. Even though, we can see that the two medians are quite similar. This plot help us to identify if our model is overfitting the train data set.

Computation by hand

Finally, let's compute by hand the OLS estimated coefficients using the formula:

$$\hat{\beta}_{OLS} = \left(X^\top X\right)^{-1} X^\top Y$$

```
X <- as.matrix(cbind(rep(1, nrow(facebook_data)), facebook_data$months))
Y <- as.matrix(facebook_data$users)
inv(t(X)%*%X)%*%t(X)%*%Y
```

```
         [,1]
[1,] 90.16677
[2,] 18.29366
```

We can notice that we have nearly the same values as the R regression function gave us earlier.

V. Medical data

Data Set Exploration

```
diabetes_data <- read.table(file = "diabetes.txt",header = TRUE)
dim(diabetes_data)
```

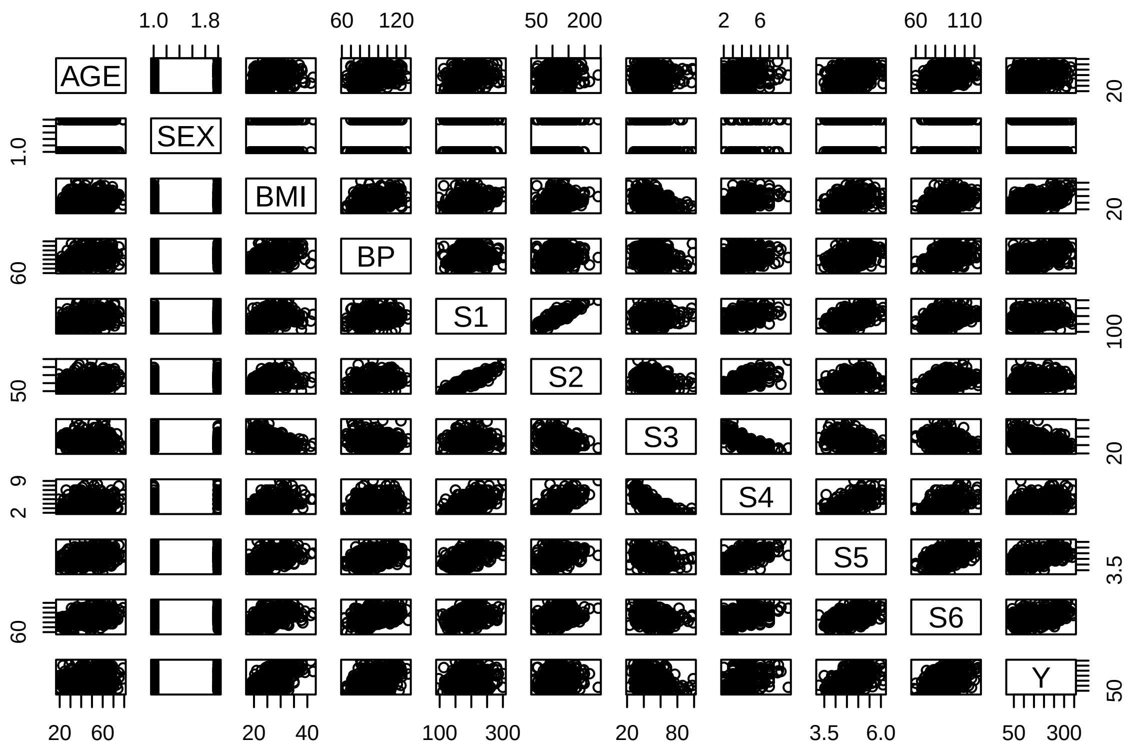
```
[1] 442  11
```

```
head(diabetes_data)
```

```
  AGE SEX  BMI  BP  S1    S2 S3 S4     S5 S6   Y
1  59   2 32.1 101 157  93.2 38  4 4.8598 87 151
2  48   1 21.6  87 183 103.2 70  3 3.8918 69  75
3  72   2 30.5  93 156  93.6 41  4 4.6728 85 141
4  24   1 25.3  84 198 131.4 40  5 4.8903 89 206
5  50   1 23.0 101 192 125.4 52  4 4.2905 80 135
6  23   1 22.6  89 139  64.8 61  2 4.1897 68  97
```

First of all, let's try to visualize our data in order to see how does it looks like.

```
plot(diabetes_data)
```



Regression Model Application

In order to compute the values of our coefficients using the data set we are going to use the OLS estimated values which minimizes the quadratic error on the diabetes data set.

$$E(\beta) = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1}))^2$$

$$\hat{\beta}_{OLS} = \left(X^\top X\right)^{-1} X^\top Y$$

```r
diabetes_model <- lm(formula = Y ~ ., data = diabetes_data)
summary(diabetes_model)
```

```
Call:
lm(formula = Y ~ ., data = diabetes_data)

Residuals:
     Min       1Q   Median       3Q      Max
-155.827  -38.536   -0.228   37.806  151.353

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -334.56714   67.45462  -4.960 1.02e-06 ***
AGE           -0.03636    0.21704  -0.168 0.867031
SEX          -22.85965    5.83582  -3.917 0.000104 ***
BMI            5.60296    0.71711   7.813 4.30e-14 ***
BP             1.11681    0.22524   4.958 1.02e-06 ***
S1            -1.09000    0.57333  -1.901 0.057948 .
S2             0.74645    0.53083   1.406 0.160390
S3             0.37200    0.78246   0.475 0.634723
S4             6.53383    5.95864   1.097 0.273459
S5            68.48312   15.66972   4.370 1.56e-05 ***
S6             0.28012    0.27331   1.025 0.305990
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 54.15 on 431 degrees of freedom
Multiple R-squared:  0.5177,    Adjusted R-squared:  0.5066
F-statistic: 46.27 on 10 and 431 DF,  p-value: < 2.2e-16
```

From the summary of our regression we can tell that:

- The residuals are quite symmetrically distributed around their median;
- The intercept of our line is equal to -334.56714, we also notice the influence of each co-variable on Y;
- The standard error and the t-value are provided to show how the p-values were calculated;
- Some p-values are very low but other are large which means that some co-variables aren't significant;
- The R^2 tells us that the p co-variables can explain 51.77% of the variation in the target variable Y;
- The first degree of freedom corresponds to $p - 1$ with $p = 11$ the number of variables of the model;
- The second degree of freedom corresponds to $n - p$ with $n = 442$ the number of data points;
- Some estimated p-values are very high which means that the corresponding co-variables doesn't have that much influence on the quantitative measure of disease progression one year after baseline.

Study of the Normality

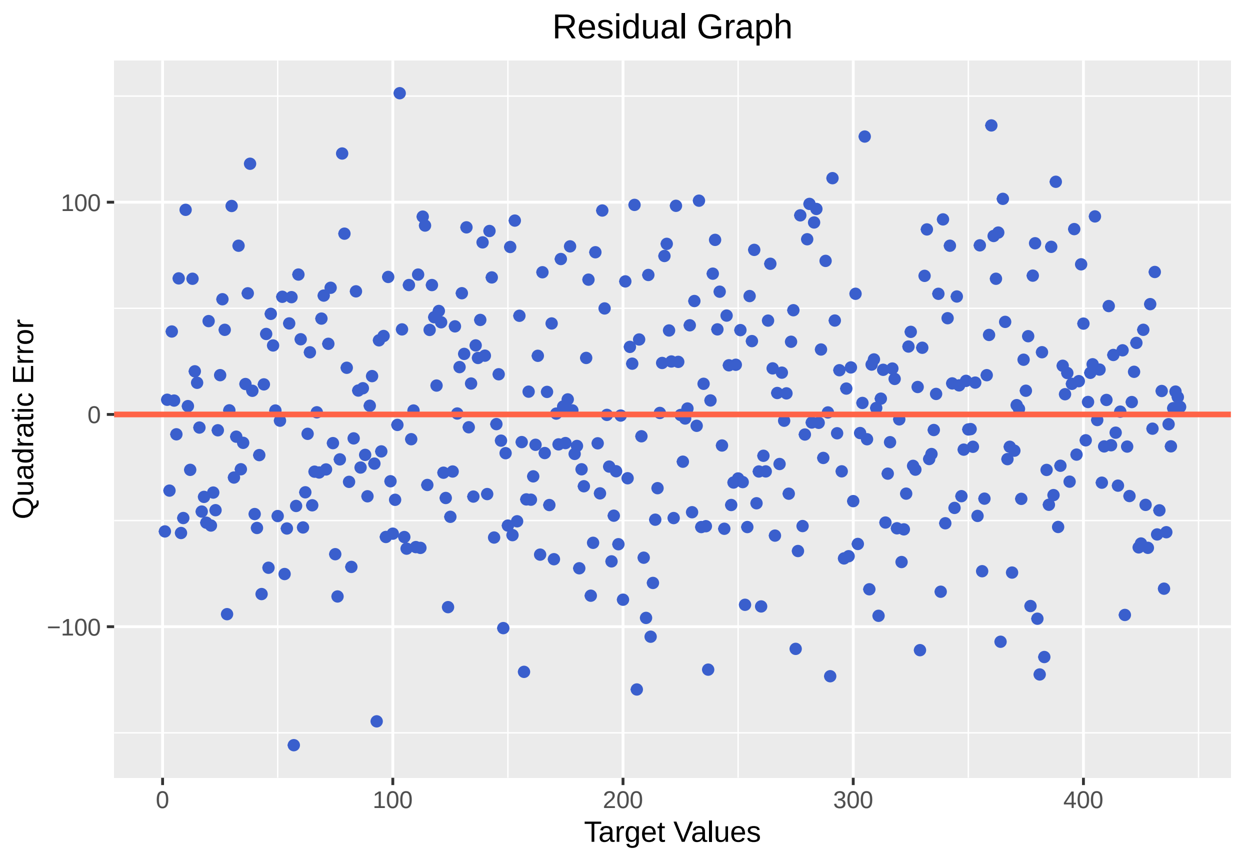
```
shapiro.test(diabetes_data$Y)
```

```

	Shapiro-Wilk normality test

data:  diabetes_data$Y
W = 0.94906, p-value = 3.364e-11
```

Here we can see that the p-value is equal to $3.364.10^{-11}$ so we can reject the null hypothesis with a very low alpha level around 10^{-10}. Therefore, we can assume that our data is not normally distributed.

Study of the Residuals

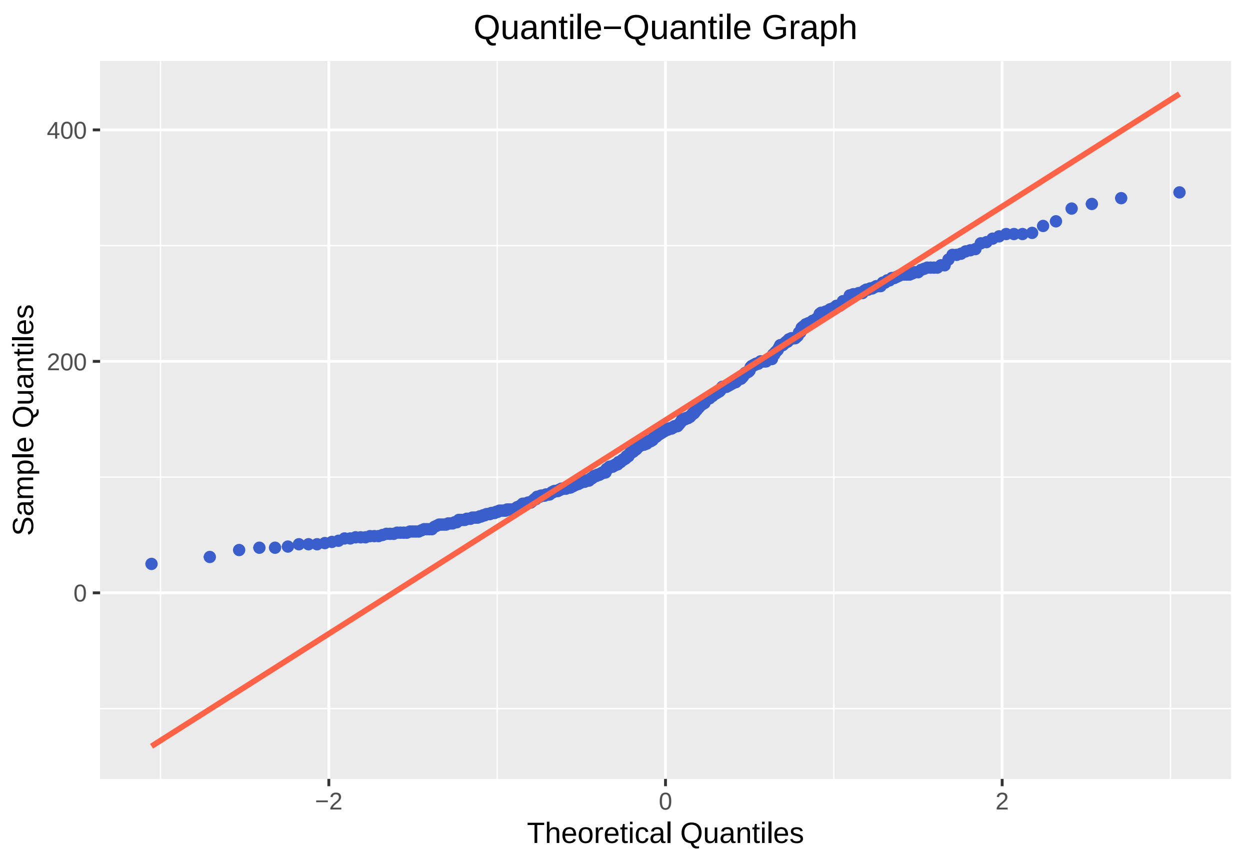
```
residuals <- as.data.frame(diabetes_model$residuals)
ggplot(data = residuals,
       mapping = aes(x = seq(1, nrow(residuals)), y = diabetes_model$residuals)) +
  geom_point(size = 1.5, color = "royalblue3") +
  geom_hline(yintercept = 0, size = 1, color = "tomato1") +
  labs(title = "Residual Graph", x = "Target Values", y = "Quadratic Error") +
  theme(plot.title = element_text(hjust = 0.5))
```



The distribution of points is random, so there is no more information to capture from the residuals.

Study of the Quantiles

```
ggplot(data = diabetes_data, mapping = aes(sample = Y)) +
  stat_qq(size = 1.5, color = "royalblue3") +
  stat_qq_line(size = 1, color = "tomato1") +
  labs(title = "Quantile-Quantile Graph",
       x = "Theoretical Quantiles", y = "Sample Quantiles") +
  theme(plot.title = element_text(hjust = 0.5))
```



Here we are comparing two probability distributions by plotting their quantiles against each other.

They fit the y = x line so we can assume that the two distributions are similar for the middle values.

So, the linear model is the right model to use in this situation.

However, for extreme values the points are a little bit far from the first bisector which means that the distributions may not be as similar as we assume, especially on extreme values.

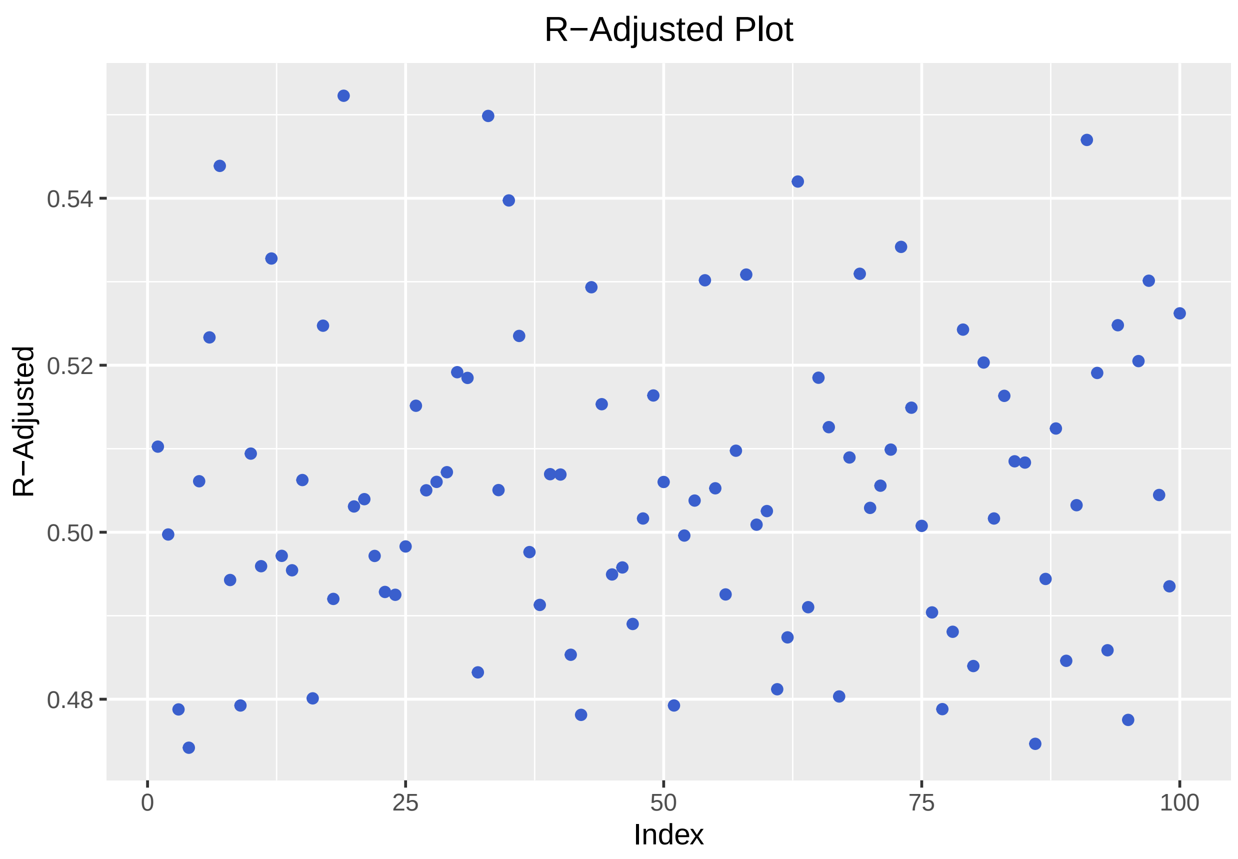
Random Partitionning

Now, let's split our data set into training and testing data set in order to study the predictive power of our model which refers to it's ability to compute good predictions for new data that he has never seen before.

```
R_Squared_Adj <- vector(length = 100)
RMSD_Train <- vector(length = 100)
RMSD_Test <- vector(length = 100)
```

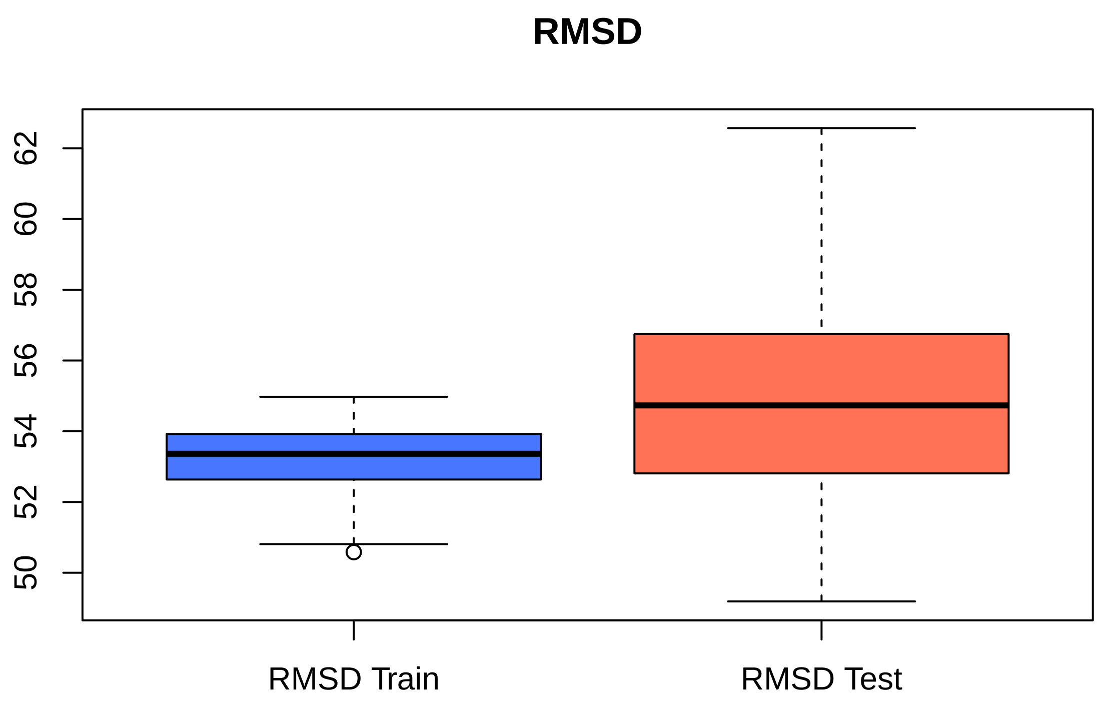
```r
for (i in 1:100)
{
  # train and test split + model training
  sample <- sample(c(TRUE, FALSE), nrow(diabetes_data), replace = TRUE, prob = c(0.75, 0.25))
  Train <- diabetes_data[sample, ]
  Test <- diabetes_data[!sample, ]
  model <- lm(formula = Y ~ ., data = Train)
  # compute R Adjusted
  R_Squared_Adj[i] <- summary(model)["adj.r.squared"]$adj.r.squared
  # compute RMSD Train
  Sum_Of_Square_Train <- sum((Train$Y - predict(model, newdata = Train))^2)
  RMSD_Train[i] <- sqrt(Sum_Of_Square_Train / length(Train$Y))
  # compute RMSD Test
  Sum_Of_Square_Test <- sum((Test$Y - predict(model, newdata = Test))^2)
  RMSD_Test[i] <- sqrt(Sum_Of_Square_Test / length(Test$Y))
}
```

```r
ggplot(data.frame(x = 1:length(R_Squared_Adj), y = R_Squared_Adj), aes(x, y)) +
  geom_point(size = 1.5, color = "royalblue3") +
  labs(title = "R-Adjusted Plot", x = "Index", y = "R-Adjusted") +
  theme(plot.title = element_text(hjust = 0.5))
```



After plotting the R-Adjusted we observe that all the values are in between 0.46 and 0.57.

```r
boxplot(RMSD_Train, RMSD_Test, main = "RMSD", names = c("RMSD Train", "RMSD Test"),
        col = c("royalblue1", "coral1"))
```



We notice that the RMSD computed on the test data set is larger than the RMSD computed on the train data set. It's because the coefficients are estimated based on the train data set which means they perfectly fit those data compared to the test data. Even though, we can see that the two medians are quite similar. This plot help us to identify if our model is overfitting the train data set.

Computation by hand

Finally, let's compute by hand the OLS estimated coefficients using the formula:

$$\hat{\beta}_{OLS} = \left(X^\top X\right)^{-1} X^\top Y$$

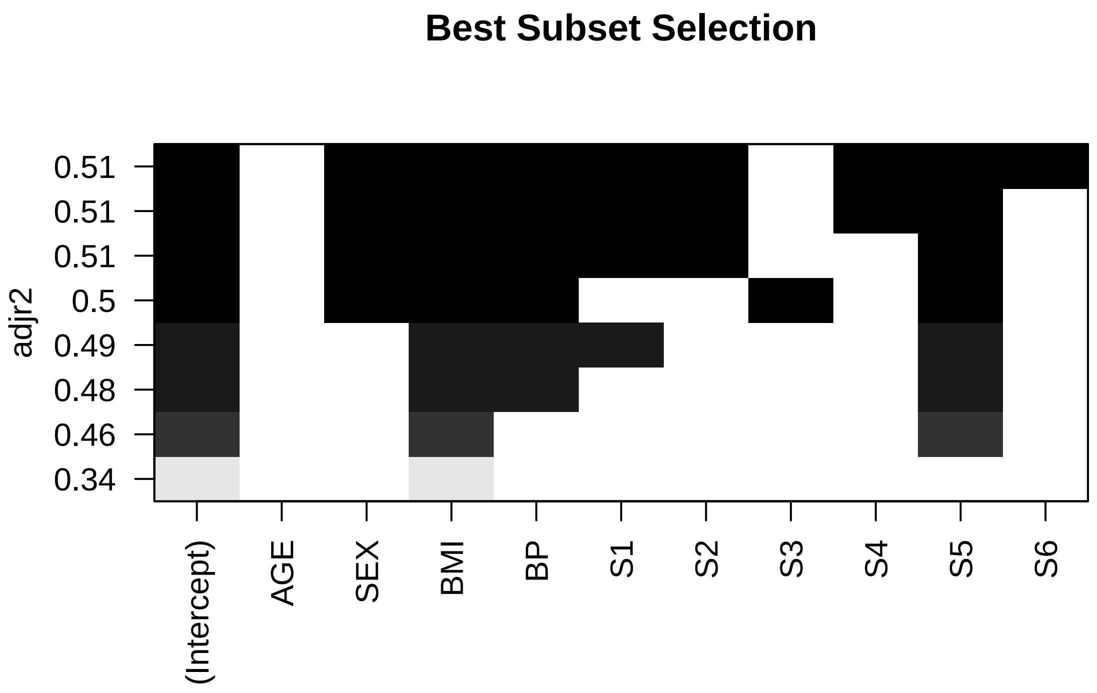
```r
X <- as.matrix(cbind(rep(1, nrow(diabetes_data)), diabetes_data[,1:10]))
Y <- as.matrix(diabetes_data$Y)
t(inv(t(X)%*%X)%*%t(X)%*%Y) # transpose is just for visual display
```

```
          [,1]       [,2]     [,3]     [,4]     [,5]      [,6]      [,7]
[1,] -334.5473 -0.0844378 -22.8566 5.653978 1.073965 -1.210768 0.6542097
          [,8]     [,9]    [,10]     [,11]
[1,] 0.2943411 6.575578 68.46167 0.2789234
```

We can notice that we have nearly the same values as the R regression function gave us.

Subset Method

```r
subsets <- regsubsets(Y ~ ., data = diabetes_data)
plot(subsets, scale = "adjr2", main = "Best Subset Selection")
```



In the plot above, we can see different models who takes in count different co-variables classified by descending order of R^2 adjusted. Which show us the best co-variables combination to maximize the R^2 adjusted.

Conclusion

To conclude, we can improve our model by making it the most simple possible which means having the highest predictive power using the less co-variables possible. In order to do so we need to select only the most relevant co-variables in the provided data set.

The easiest way to do so is to look at the p-values, the nearest ones to 1 correspond to the less influential co-variables in the model. For example, we can put aside the Age for example:

```r
diabetes_simple_model <- lm(formula = Y ~ SEX + BMI + BP + S1 + S2 + S3 + S4 + S5 + S6,
                            data = diabetes_data)
summary(diabetes_simple_model)
```

```
Call:
lm(formula = Y ~ SEX + BMI + BP + S1 + S2 + S3 + S4 + S5 + S6,
    data = diabetes_data)

Residuals:
     Min       1Q   Median       3Q      Max
-155.557  -38.259   -0.335   37.448  152.291

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -333.9514    67.2786  -4.964 9.97e-07 ***
SEX          -22.9564     5.8006  -3.958 8.85e-05 ***
BMI            5.6037     0.7163   7.823 3.99e-14 ***
BP             1.1096     0.2209   5.024 7.43e-07 ***
S1            -1.0882     0.5726  -1.901    0.058 .
S2             0.7427     0.5298   1.402    0.162
S3             0.3671     0.7810   0.470    0.639
S4             6.5484     5.9513   1.100    0.272
S5            68.3215    15.6224   4.373 1.54e-05 ***
S6             0.2741     0.2706   1.013    0.312
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 54.09 on 432 degrees of freedom
Multiple R-squared:  0.5177,    Adjusted R-squared:  0.5077
F-statistic: 51.53 on 9 and 432 DF,  p-value: < 2.2e-16
```

We obtain a similar R^2 which mean the Age isn't significantly involved in our model, but we should pay attention to the impact of dependence on testing coefficients.

Still there are better methods to select co-variables such as: Best subset method, Greedy method (forward, backward, step wise), Information criterion (AIC, BIC, C_p of Mallows)...

Even though, we need to be aware of co-variables collinearity, if it exists the co-variable significativity test is useless. We should also not look at the estimated value of the coefficient to determine co-variable significativity because even a very low estimated coefficient can become bigger at the end depending on the co-variable unit and magnitude.

Another important thing is to pay attention to high dimensional modeling, if there is a huge number of co-variables with a few number of observations ($p >> n$) then $X^T X$ is non-inversible.

Finally, we should keep in mind that we need to balance between a complex model with a lot of variables and a simple model with few variables, that's the trade-off between Bias and Variance.