SIMPLE LINEAR REGRESSION

1. SELECTION OF DATASET:

The dataset used here is 'Glass Identification Database'. This dataset is used to predict the glass type from chemical properties.

Dimension: 214 rows and 10 columns

```
> library(mlbench)
> data(Glass)
> head(Glass)          #To see the first six rows of the dataset
       RI    Na   Mg   Al    Si    K   Ca Ba   Fe Type
1 1.52101 13.64 4.49 1.10 71.78 0.06 8.75  0 0.00    1
2 1.51761 13.89 3.60 1.36 72.73 0.48 7.83  0 0.00    1
3 1.51618 13.53 3.55 1.54 72.99 0.39 7.78  0 0.00    1
4 1.51766 13.21 3.69 1.29 72.61 0.57 8.22  0 0.00    1
5 1.51742 13.27 3.62 1.24 73.08 0.55 8.07  0 0.00    1
6 1.51596 12.79 3.61 1.62 72.97 0.64 8.07  0 0.26    1
```

```
> summary(Glass)      #To see the summary of the dataset
       RI              Na              Mg              Al              Si               K                Ca        
 Min.   :1.511   Min.   :10.73   Min.   :0.000   Min.   :0.290   Min.   :69.81   Min.   :0.0000   Min.   : 5.430  
 1st Qu.:1.517   1st Qu.:12.91   1st Qu.:2.115   1st Qu.:1.190   1st Qu.:72.28   1st Qu.:0.1225   1st Qu.: 8.240  
 Median :1.518   Median :13.30   Median :3.480   Median :1.360   Median :72.79   Median :0.5550   Median : 8.600  
 Mean   :1.518   Mean   :13.41   Mean   :2.685   Mean   :1.445   Mean   :72.65   Mean   :0.4971   Mean   : 8.957  
 3rd Qu.:1.519   3rd Qu.:13.82   3rd Qu.:3.600   3rd Qu.:1.630   3rd Qu.:73.09   3rd Qu.:0.6100   3rd Qu.: 9.172  
 Max.   :1.534   Max.   :17.38   Max.   :4.490   Max.   :3.500   Max.   :75.41   Max.   :6.2100   Max.   :16.190  
       Ba              Fe            Type  
 Min.   :0.000   Min.   :0.00000   1:70  
 1st Qu.:0.000   1st Qu.:0.00000   2:76  
 Median :0.000   Median :0.00000   3:17  
 Mean   :0.175   Mean   :0.05701   5:13  
 3rd Qu.:0.000   3rd Qu.:0.10000   6: 9  
 Max.   :3.150   Max.   :0.51000   7:29  
```

```
> str(Glass)           #To see the structure of the dataset
'data.frame':   214 obs. of  10 variables:
 $ RI  : num  1.52 1.52 1.52 1.52 1.52 ...
 $ Na  : num  13.6 13.9 13.5 13.2 13.3 ...
 $ Mg  : num  4.49 3.6 3.55 3.69 3.62 3.61 3.6 3.61 3.58 3.6 ...
 $ Al  : num  1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...
 $ Si  : num  71.8 72.7 73 72.6 73.1 ...
 $ K   : num  0.06 0.48 0.39 0.57 0.55 0.64 0.58 0.57 0.56 0.57 ...
 $ Ca  : num  8.75 7.83 7.78 8.22 8.07 8.07 8.17 8.24 8.3 8.4 ...
 $ Ba  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Fe  : num  0 0 0 0 0 0.26 0 0 0 0.11 ...
 $ Type: Factor w/ 6 levels "1","2","3","5",..: 1 1 1 1 1 1 1 1 1 1 ...
```

The structure of the data set gives a detailed description of the data types and the levels for the variables which have factors.

```
> any(is.na(Glass))   #To check whether there are any null values in the dataset
[1] FALSE
```

2. CORRELATION ANALYSIS OF THE DATASET:

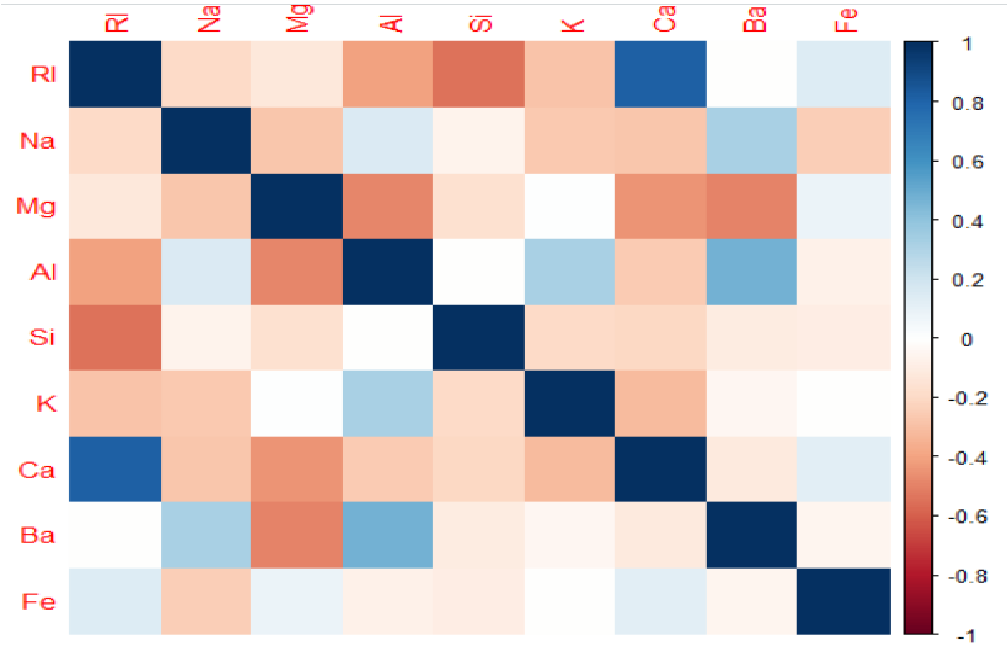
```
> #The following libraries are used for Exploratory Data Analysis
> library(ggplot2)
> library(ggthemes)
> library(dplyr)
```

```
> #Grabbing only the numeric columns as we can't see correlation between the categorical variables
> num.cols<- sapply(Glass,is.numeric)
> #Filtering to numeric columns for correlation
> cor.data<-cor(Glass[ , num.cols])
> cor.data
```

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
RI	1.0000000000	-0.19188538	-0.122274039	-0.40732603	-0.54205220	-0.289832711	0.8104027	-0.0003860189	0.143009609
Na	-0.1918853790	1.00000000	-0.273731961	0.15679367	-0.06980881	-0.266086504	-0.2754425	0.3266028795	-0.241346411
Mg	-0.1222740393	-0.27373196	1.000000000	-0.48179851	-0.16592672	0.005395667	-0.4437500	-0.4922621178	0.083059529
Al	-0.4073260341	0.15679367	-0.481798509	1.00000000	-0.00552372	0.325958446	-0.2595920	0.4794039017	-0.074402151
Si	-0.5420521997	-0.06980881	-0.165926723	-0.00552372	1.00000000	-0.193330854	-0.2087322	-0.1021513105	-0.094200731
K	-0.2898327111	-0.26608650	0.005395667	0.32595845	-0.19333085	1.000000000	-0.3178362	-0.0426180594	-0.007719049
Ca	0.8104026963	-0.27544249	-0.443750026	-0.25959201	-0.20873215	-0.317836155	1.0000000	-0.1128409671	0.124968219
Ba	-0.0003860189	0.32660288	-0.492262118	0.47940390	-0.10215131	-0.042618059	-0.1128410	1.0000000000	-0.058691755
Fe	0.1430096093	-0.24134641	0.083059529	-0.07440215	-0.09420073	-0.007719049	0.1249682	-0.0586917554	1.000000000

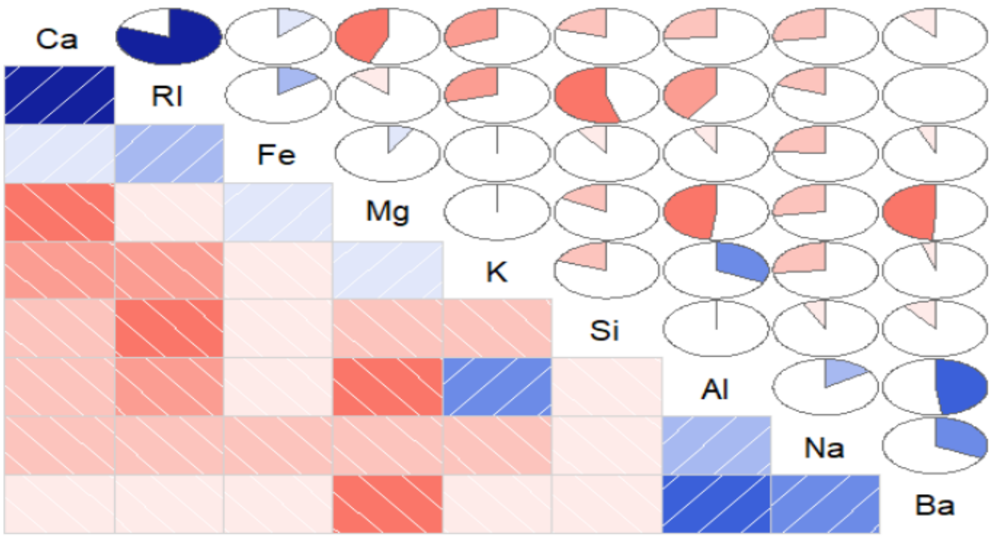
```
> #For proper data visualization we use the 'corrgram' package and the 'corrplot' package
> library(corrgram)
> library(corrplot)
```

```
> #Now we perform the correlation plot and see what we can infer from that
> corrplot(cor.data, method='color')
```



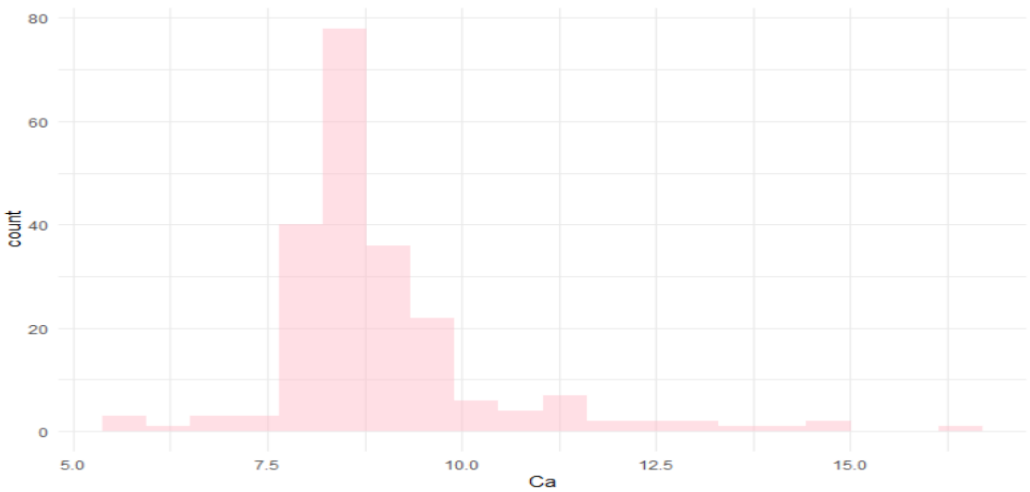
We can see from the above correlation plot that white indicates zero correlation, the shades from white to blue indicates positive correlation and from white to red indicates negative correlation. Positive correlation indicates that as the value of one variable increases, the value of the other variable also increases, i.e., directly proportional, while Negative correlation indicates that as the value of one variable increases the value of the other variable will decrease, i.e., indirectly proportional. We can see that Ca and RI are highly correlated to each other. This means that higher the refractive index(RI),higher is the amount of Calcium(Ca). We are going to predict the glass type with Ca based on RI. Here Ca is a dependent variable and RI is an independent variable.

```
> #Now we perform the correlogram and see what we can infer from that
> corrgram(Glass ,order=TRUE, lower.panel=panel.shade, upper.panel=panel.pie, text.panel=panel.txt)
```



In R, correlograms are implemented using the 'corrgram' function. It shows the graph of the correlation matrix and is very useful for highlighting the most correlated values. The results of this plot can be interpreted in the same way as 'corrplot'. Here blue is positive correlation and pink is negative correlation.

```
> #Histogram of the amount of Ca
> ggplot(Glass,aes(x=Ca))+ geom_histogram(bins=20, alpha=0.5, fill='pink') + theme_minimal()
>
```



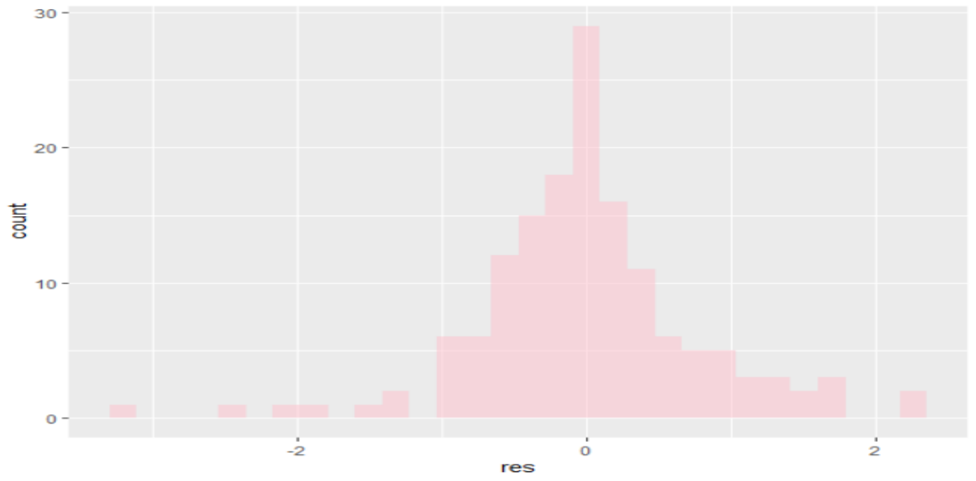
As we have to predict the amount of Ca so we draw a histogram of it. From the graph we can see that the high amount of Ca is below the mean of the Ca and there are much more values or observations beyond the mean value. So we can say that this graph is positively skewed.

3. SIMPLE LINEAR REGRESSION:

```
> #We need to split our data into a training set and a testing set in order to test our accuracy, so we can do this using the caTools library
> library(caTools)
>
> #Spliting up the sample for training and testing and assigns boolean values to a new column
> sample <- sample.split(Glass$Ca, SplitRatio = 0.70)
>
> #Training data
> train_data = subset(Glass, sample == TRUE)
>
> #Testing data
> test_data= subset(Glass, sample == FALSE)
>
> #Training the model
> model <- lm(Ca ~ RI, train_data)
>
> summary(model)

Call:
lm(formula = Ca ~ RI, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.1264 -0.4039 -0.0320  0.3379  2.3436 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  -619.77      33.32  -18.60   <2e-16 ***
RI            414.05      21.94   18.87   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7743 on 147 degrees of freedom
Multiple R-squared:  0.7078,	Adjusted R-squared:  0.7058 
F-statistic: 356.1 on 1 and 147 DF,  p-value: < 2.2e-16
```

The residuals are the difference between the actual values of the variable we are predicting and predicted values from our regression.
The stars are for significance levels, with the number of asterisks displayed according to the p-value computed. *** indicates high significance. In this case, *** indicates that there is high significance between RI and Ca.
The estimated coefficient is the value of slope calculated by the regression.
Standard Error of the Coefficient Estimate is the measure of the variability in the estimate for the coefficient.
t-value shows how many standard deviations the coefficient is far away from zero. Further it is away from zero, stronger the relationship between the variables.
p-value shows whether the overall model is significant or not. Coefficients having p-value less than the level of significance are said to be statistically significant. In this case we can say that the variable RI is statistically significant for the prediction of amount of Ca.
R-squared is an overall measure of the strength of association and adjusted R-squared gives a more proper estimate of the R-squared value for the population. Its value shows that 70.58 % of the variance in the amount of Ca can be predicted from RI.

```
> #Visualizing the model
>
> res<- residuals(model)       #Grabbing the residuals
> res<- as.data.frame(res)     #Converting the residuals to dataframe for ggplot
> head(res)
          res
1  -1.2553089
3  -0.2254548
4  -0.3982465
6   0.1556359
7  -0.3530154
10 -0.1727012
>
> #Histogram of residuals
> ggplot(res, aes(res)) + geom_histogram(fill='pink', alpha=0.5)
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We have to remove the negative values from here.

```
> #Predictions
> Ca.predictions <- predict(model,test_data)
> results<- cbind(Ca.predictions,test_data$Ca)
> colnames(results) <- c('pred','real')
> results<- as.data.frame(results)
> results

         pred   real
2    8.597544  7.83
5    8.518875  8.07
8    8.576842  8.24
9    9.247600  8.30
11   7.810852  8.09
23   8.494032  8.70
30   8.692775  8.49
33   8.655511  8.56
43   8.672073  8.59
44  10.456622  9.74
46   9.173071  9.00
48  12.348823  9.82
57   6.336839  8.35
62   9.491889  8.67
68  10.216474  9.85
69  10.216474  9.82
71   7.823273  7.36
73   7.901943  7.83
79   7.984752  7.94
83   8.121388  8.10
84   7.906083  8.05
86   8.034438  8.21
87   7.802571  8.03
90   8.096545  8.08
91   8.928783  8.96
97   8.928783  9.13
98   8.523015  8.90
102  8.469189  9.23
```

```
106 11.553850 13.24
107 14.245165 13.30
109 10.506308 11.52
110  8.833552 10.99
120  8.146231  7.96
121  8.941204  8.42
127  8.208339  8.60
129  9.868673  9.57
142  8.970188  8.41
143  8.187636  8.23
145  8.179355  8.81
148  7.972331  8.33
150  8.108967  8.53
158 10.088119  9.65
160  8.742461  8.81
161  8.891518  8.99
163 10.460762  9.14
164  7.574844  5.87
166 10.295143 11.41
168  9.458765 11.53
169  8.204198 10.17
173  6.775731  6.93
174  9.765161 12.50
175  9.827268  9.70
177  9.193774  9.57
182  9.123386  9.95
185  5.922791  6.65
186  5.989039  5.43
190 11.098397  8.61
191  7.984752  8.67
192  7.939207  8.76
196  7.703199  9.07
197  7.748745  9.41
201  7.550001  8.34
203  7.574844  8.39
204  8.171074  8.28
206  8.477470  8.61

> #To remove negative predictions and replace it with 0
> to_zero <- function(x){
+   if(x <0) {
+     return(0)
+   }else{
+     return(x)
+   }
+ }
>
> results$pred <- sapply(results$pred,to_zero)
> results

         pred  real
2    8.597544  7.83
5    8.518875  8.07
8    8.576842  8.24
9    9.247600  8.30
11   7.810852  8.09
23   8.494032  8.70
30   8.692775  8.49
33   8.655511  8.56
43   8.672073  8.59
44  10.456622  9.74
46   9.173071  9.00
48  12.348823  9.82
57   6.336839  8.35
62   9.491889  8.67
68  10.216474  9.85
69  10.216474  9.82
71   7.823273  7.36
73   7.901943  7.83
79   7.984752  7.94
83   8.121388  8.10
84   7.906083  8.05
86   8.034438  8.21
87   7.802571  8.03
90   8.096545  8.08
91   8.928783  8.96
```

```
97    8.928783   9.13
98    8.523015   8.90
102   8.469189   9.23
106  11.553850  13.24
107  14.245165  13.30
109  10.506308  11.52
110   8.833552  10.99
120   8.146231   7.96
121   8.941204   8.42
127   8.208339   8.60
129   9.868673   9.57
142   8.970188   8.41
143   8.187636   8.23
145   8.179355   8.81
148   7.972331   8.33
150   8.108967   8.53
158  10.088119   9.65
160   8.742461   8.81
161   8.891518   8.99
163  10.460762   9.14
164   7.574844   5.87
166  10.295143  11.41
168   9.458765  11.53
169   8.204198  10.17
173   6.775731   6.93
174   9.765161  12.50
175   9.827268   9.70
177   9.193774   9.57
182   9.123386   9.95
185   5.922791   6.65
186   5.989039   5.43
190  11.098397   8.61
191   7.984752   8.67
192   7.939207   8.76
196   7.703199   9.07
197   7.748745   9.41
201   7.550001   8.34
203   7.574844   8.39
204   8.171074   8.28
206   8.477470   8.61
```

```
> #Evaluating the prediction values by the method of MSE(Mean Squared Error)
> mse <- mean((results$real - results$pred)^2)
> print(mse)
[1] 0.9650583
>
> #Evaluating the prediction values by the method of RMSE(Root Mean Squared Error)
> mse^0.5
[1] 0.9823738
>
> #Or we can just use the R-Squared Value for the model which gives the accuracy of the model
> SSE <- sum((results$pred - results$real)^2)      #Sum Square of Errors
> TSS <- sum( (mean(Glass$Ca) - results$real)^2)   #Total Sum of Squares
> R2 <- 1-SSE/TSS                                  #R-Squared
> R2
[1] 0.5165802
```

R-Squared (Coefficient of Determination) - This value lies between 0 and 1, and the higher it is, the better the model fits the data set.

Our main aim is to find those variables who give the lowest RMSE value and the highest R-Squared value.

The R-Squared for the training set is 51.65%. It means that the model can explain more than 51.65% of the variation.