Exercise 3

Question 1: (50 points)

As explained in Lesson 5, data exploration through visualization is important because statistics alone might not tell the entire story. This is best shown by the French statistician Francis Anscombe in 1973 when he presented four sets of data. This data is shown here

Data I		Data II		Data III		Data IV	
x	y	x	y	x	y	x	y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

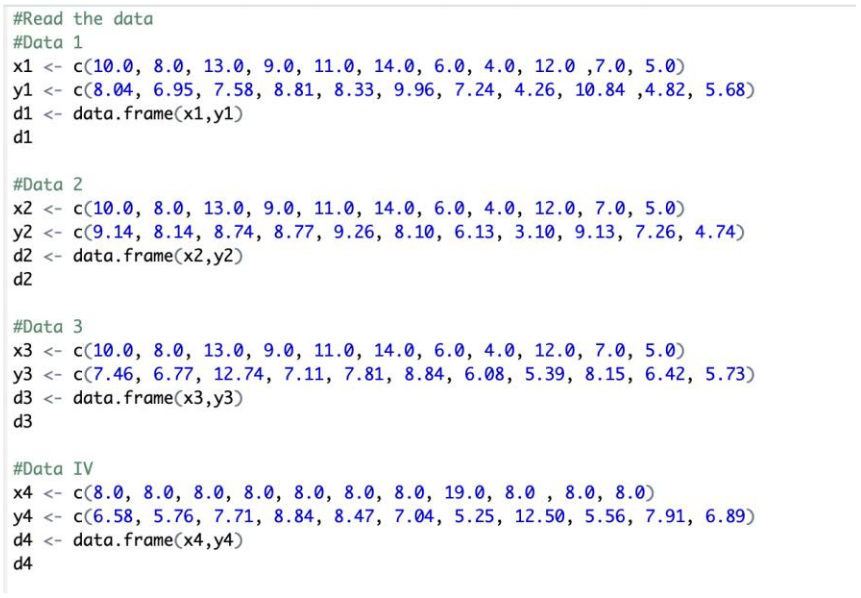
Question 1 (40)

Calculate the mean, variance, correlation, and a linear regression for each data set. Using base R or ggplot2, create a visual representation of this data. What does this visualization show?

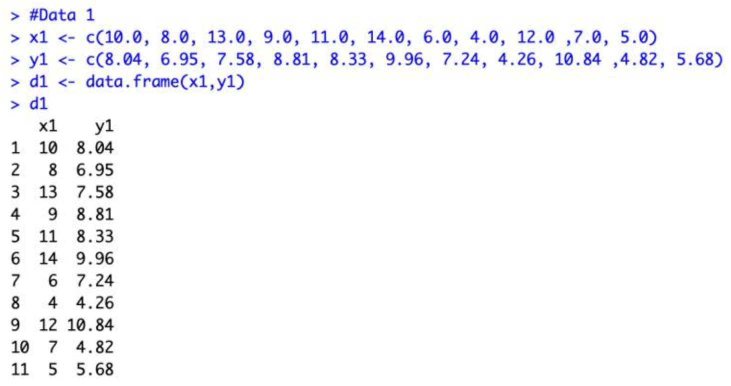
References:

1. Anscombe, F. J. (1973). "Graphs in Statistical Analysis". American Statistician 27 (1): 17–21.
2. [Anscombe's quartet (Links to an external site.)Links to an external site.]

Question 1:



```
#Read the data
#Data 1
x1 <- c(10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0 ,7.0, 5.0)
y1 <- c(8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84 ,4.82, 5.68)
d1 <- data.frame(x1,y1)
d1

#Data 2
x2 <- c(10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0, 7.0, 5.0)
y2 <- c(9.14, 8.14, 8.74, 8.77, 9.26, 8.10, 6.13, 3.10, 9.13, 7.26, 4.74)
d2 <- data.frame(x2,y2)
d2

#Data 3
x3 <- c(10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0, 7.0, 5.0)
y3 <- c(7.46, 6.77, 12.74, 7.11, 7.81, 8.84, 6.08, 5.39, 8.15, 6.42, 5.73)
d3 <- data.frame(x3,y3)
d3

#Data IV
x4 <- c(8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 19.0, 8.0 , 8.0, 8.0)
y4 <- c(6.58, 5.76, 7.71, 8.84, 8.47, 7.04, 5.25, 12.50, 5.56, 7.91, 6.89)
d4 <- data.frame(x4,y4)
d4
```



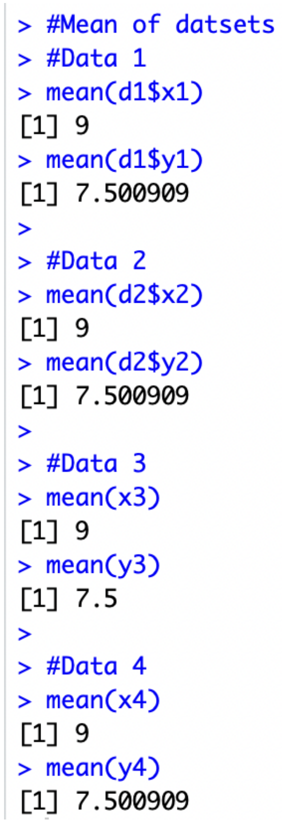
```
> #Data 1
> x1 <- c(10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0 ,7.0, 5.0)
> y1 <- c(8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84 ,4.82, 5.68)
> d1 <- data.frame(x1,y1)
> d1
   x1    y1
1  10  8.04
2   8  6.95
3  13  7.58
4   9  8.81
5  11  8.33
6  14  9.96
7   6  7.24
8   4  4.26
9  12 10.84
10  7  4.82
11  5  5.68
```

```
> #Data 2
> x2 <- c(10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0, 7.0, 5.0)
> y2 <- c(9.14, 8.14, 8.74, 8.77, 9.26, 8.10, 6.13, 3.10, 9.13, 7.26, 4.74)
> d2 <- data.frame(x2,y2)
> d2
   x2   y2
1  10 9.14
2   8 8.14
3  13 8.74
4   9 8.77
5  11 9.26
6  14 8.10
7   6 6.13
8   4 3.10
9  12 9.13
10  7 7.26
11  5 4.74
```

```
> #Data 3
> x3 <- c(10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0, 7.0, 5.0)
> y3 <- c(7.46, 6.77, 12.74, 7.11, 7.81, 8.84, 6.08, 5.39, 8.15, 6.42, 5.73)
> d3 <- data.frame(x3,y3)
> d3
   x3    y3
1  10  7.46
2   8  6.77
3  13 12.74
4   9  7.11
5  11  7.81
6  14  8.84
7   6  6.08
8   4  5.39
9  12  8.15
10  7  6.42
11  5  5.73
```

```
> x4 <- c(8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 19.0, 8.0 , 8.0, 8.0)
> y4 <- c(6.58, 5.76, 7.71, 8.84, 8.47, 7.04, 5.25, 12.50, 5.56, 7.91, 6.89)
> d4 <- data.frame(x4,y4)
> d4
   x4    y4
1   8  6.58
2   8  5.76
3   8  7.71
4   8  8.84
5   8  8.47
6   8  7.04
7   8  5.25
8  19 12.50
9   8  5.56
10  8  7.91
11  8  6.89
```

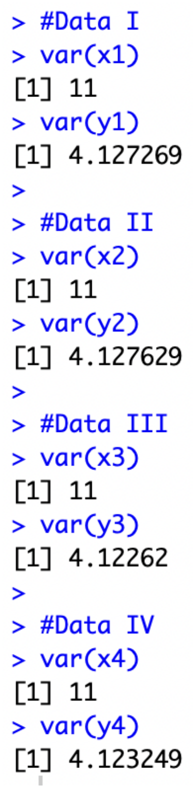
Mean:



```
> #Mean of datsets
> #Data 1
> mean(d1$x1)
[1] 9
> mean(d1$y1)
[1] 7.500909
>
> #Data 2
> mean(d2$x2)
[1] 9
> mean(d2$y2)
[1] 7.500909
>
> #Data 3
> mean(x3)
[1] 9
> mean(y3)
[1] 7.5
>
> #Data 4
> mean(x4)
[1] 9
> mean(y4)
[1] 7.500909
```

From this we can observe that the mean of all the four datasets are nearly identical. Dataset 3 has mean 7.5 whereas the rest have 7.500909 for y-axis. Mean of 9 for x-axis among all datasets. With mean we cannot conclude the difference or know the distribution of the data.

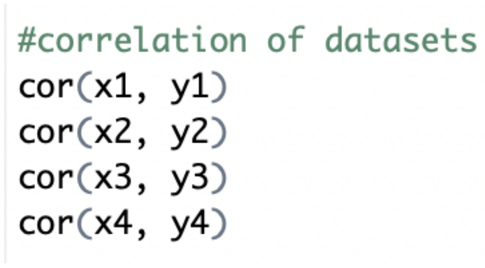
Variance:



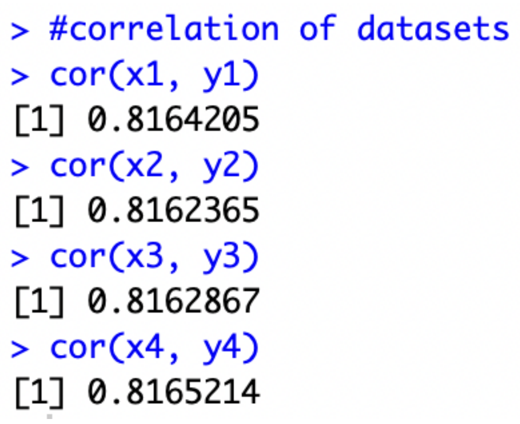
```
> #Data I
> var(x1)
[1] 11
> var(y1)
[1] 4.127269
>
> #Data II
> var(x2)
[1] 11
> var(y2)
[1] 4.127629
>
> #Data III
> var(x3)
[1] 11
> var(y3)
[1] 4.12262
>
> #Data IV
> var(x4)
[1] 11
> var(y4)
[1] 4.123249
```

Again we can notice that the variance among all four data sets are exactly similar. This shows that simple descriptive statistics of different data sets can have similar values. (4.123 in our case for y-axis and 11 for x-axis)

Correlation



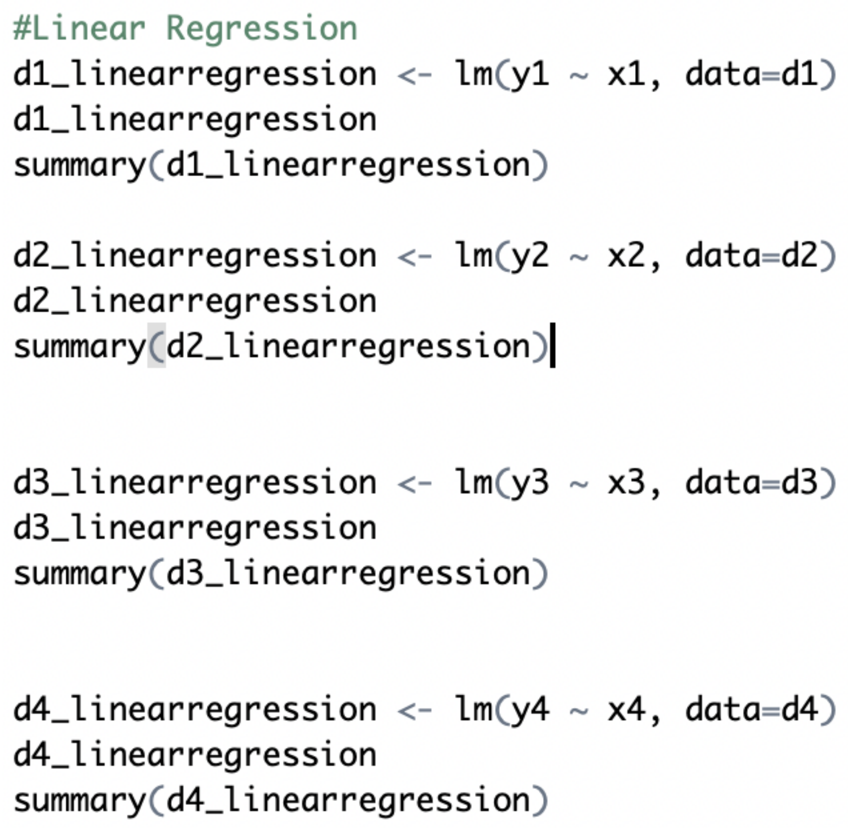
```
#correlation of datasets
cor(x1, y1)
cor(x2, y2)
cor(x3, y3)
cor(x4, y4)
```



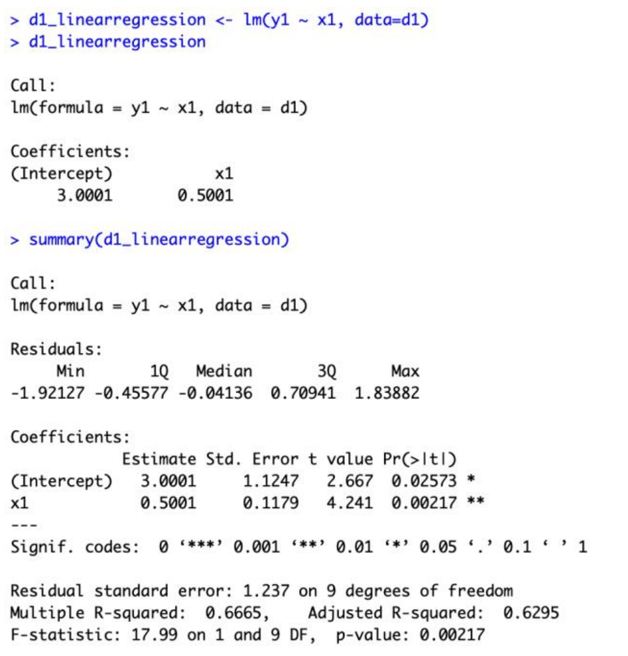
```
> #correlation of datasets
> cor(x1, y1)
[1] 0.8164205
> cor(x2, y2)
[1] 0.8162365
> cor(x3, y3)
[1] 0.8162867
> cor(x4, y4)
[1] 0.8165214
```

From this descriptive statistics also we can observe that it outputs the same values for correlation between x and y for all datasets. Hence we can conclude that descriptive statistics alone is not the best way to understand the distribution of data. It is always best to do descriptive statistics first and then some visualization to see the distribution of data and general trends. Now we will be doing some visualizations for the same.

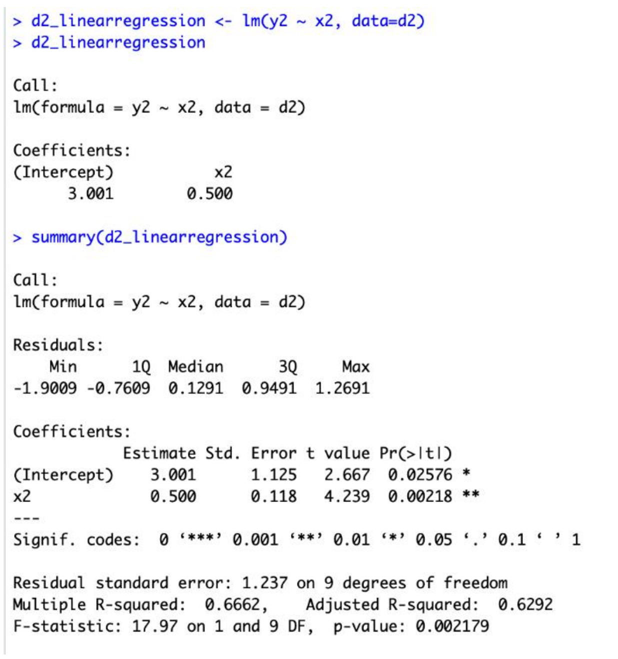
Linear regression:



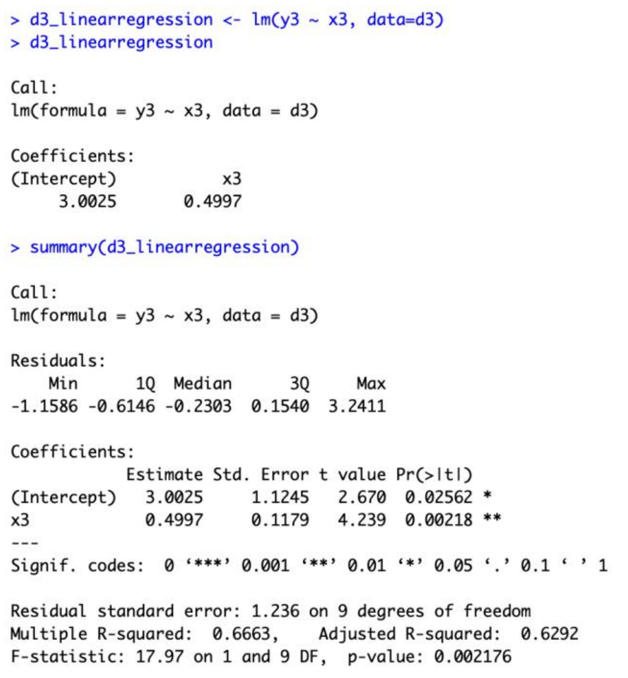
```
#Linear Regression
d1_linearregression <- lm(y1 ~ x1, data=d1)
d1_linearregression
summary(d1_linearregression)

d2_linearregression <- lm(y2 ~ x2, data=d2)
d2_linearregression
summary(d2_linearregression)


d3_linearregression <- lm(y3 ~ x3, data=d3)
d3_linearregression
summary(d3_linearregression)


d4_linearregression <- lm(y4 ~ x4, data=d4)
d4_linearregression
summary(d4_linearregression)
```



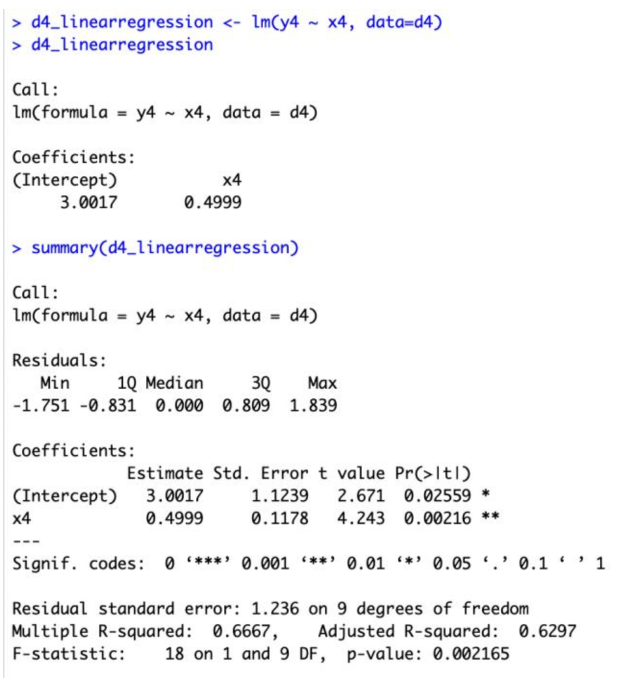
```
> d1_linearregression <- lm(y1 ~ x1, data=d1)
> d1_linearregression

Call:
lm(formula = y1 ~ x1, data = d1)

Coefficients:
(Intercept)           x1
     3.0001       0.5001

> summary(d1_linearregression)

Call:
lm(formula = y1 ~ x1, data = d1)

Residuals:
     Min       1Q   Median       3Q      Max
-1.92127 -0.45577 -0.04136  0.70941  1.83882

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.0001     1.1247   2.667  0.02573 *
x1            0.5001     0.1179   4.241  0.00217 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.237 on 9 degrees of freedom
Multiple R-squared:  0.6665,	Adjusted R-squared:  0.6295
F-statistic: 17.99 on 1 and 9 DF,  p-value: 0.00217
```



```
> d2_linearregression <- lm(y2 ~ x2, data=d2)
> d2_linearregression

Call:
lm(formula = y2 ~ x2, data = d2)

Coefficients:
(Intercept)           x2
      3.001        0.500

> summary(d2_linearregression)

Call:
lm(formula = y2 ~ x2, data = d2)

Residuals:
    Min      1Q  Median      3Q     Max
-1.9009 -0.7609  0.1291  0.9491  1.2691

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    3.001      1.125   2.667  0.02576 *
x2             0.500      0.118   4.239  0.00218 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.237 on 9 degrees of freedom
Multiple R-squared:  0.6662,	Adjusted R-squared:  0.6292
F-statistic: 17.97 on 1 and 9 DF,  p-value: 0.002179
```



```
> d3_linearregression <- lm(y3 ~ x3, data=d3)
> d3_linearregression

Call:
lm(formula = y3 ~ x3, data = d3)

Coefficients:
(Intercept)           x3
     3.0025       0.4997

> summary(d3_linearregression)

Call:
lm(formula = y3 ~ x3, data = d3)

Residuals:
    Min      1Q  Median      3Q     Max
-1.1586 -0.6146 -0.2303  0.1540  3.2411

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.0025     1.1245   2.670  0.02562 *
x3            0.4997     0.1179   4.239  0.00218 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.236 on 9 degrees of freedom
Multiple R-squared:  0.6663,	Adjusted R-squared:  0.6292
F-statistic: 17.97 on 1 and 9 DF,  p-value: 0.002176
```



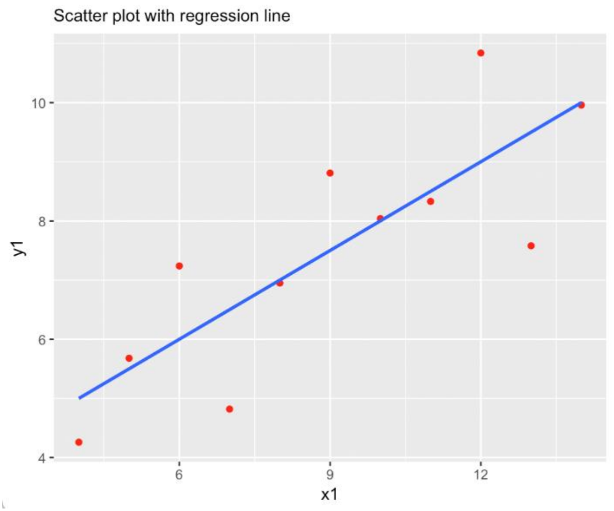
```
> d4_linearregression <- lm(y4 ~ x4, data=d4)
> d4_linearregression

Call:
lm(formula = y4 ~ x4, data = d4)

Coefficients:
(Intercept)           x4
     3.0017       0.4999

> summary(d4_linearregression)

Call:
lm(formula = y4 ~ x4, data = d4)

Residuals:
   Min     1Q Median     3Q    Max
-1.751 -0.831  0.000  0.809  1.839

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.0017     1.1239   2.671  0.02559 *
x4            0.4999     0.1178   4.243  0.00216 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.236 on 9 degrees of freedom
Multiple R-squared:  0.6667,	Adjusted R-squared:  0.6297
F-statistic:    18 on 1 and 9 DF,  p-value: 0.002165
```

Here we can observe that the residuals are different for all datasets. Each dataset has an intercept value of 3, which is nearly identical. For all datasets, the r squared value is about 0.67, the r squared value shows that the data does not fit well to the linear regression models.

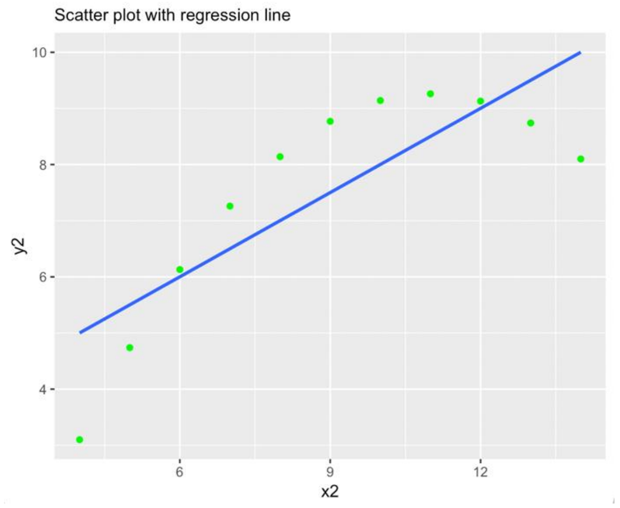
Plotting:

```
#Data Visualization
library(ggplot2)
ggplot(data=d1, aes(x1, y1)) + geom_point(color = "red") + geom_smooth(method= "lm", se = FALSE) + xlab("x1") + ylab("y1") + labs(subtitle = "Scatter plot with regression line")
ggplot(data=d2, aes(x2, y2)) + geom_point(color = "green") + geom_smooth(method= "lm", se = FALSE) + xlab("x2") + ylab("y2")+labs(subtitle = "Scatter plot with regression line")
ggplot(data=d3, aes(x3, y3)) + geom_point(color = "yellow") + geom_smooth(method= "lm", se = FALSE) + xlab("x3") + ylab("y3")+labs(subtitle = "Scatter plot with regression line")
ggplot(data=d4, aes(x4, y4)) + geom_point(color = "black") + geom_smooth(method= "lm", se = FALSE) + xlab("x4") + ylab("y4")+ labs(subtitle = "Scatter plot with regression line")
```

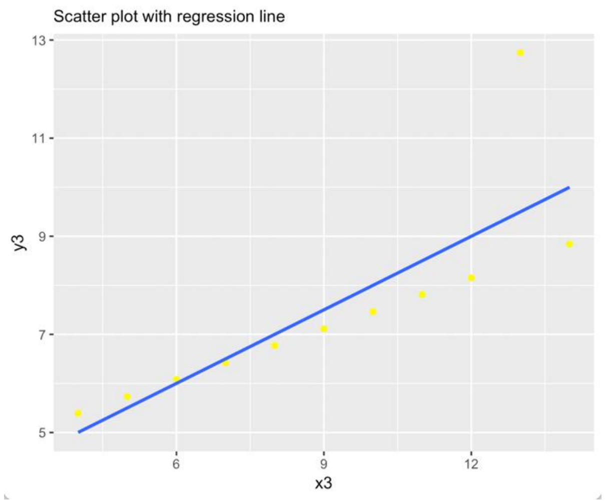
.



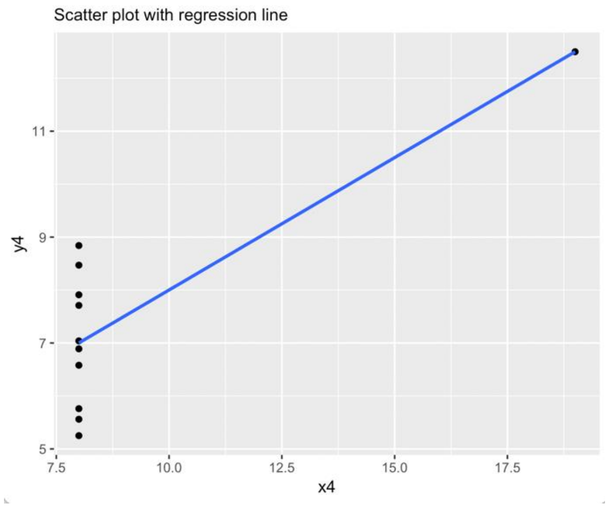
Scatterplot of dataset 1



Scatterplot of dataset 2



Scatterplot of dataset 3



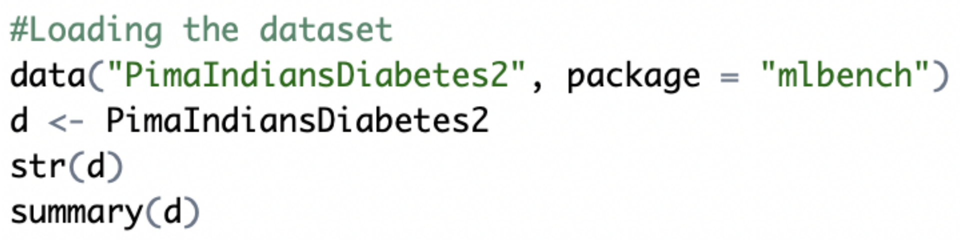
Scatterplot of dataset 4

From these we scatterplots we can observe that the first dataset fits the linear model, from second plot the data doesn't seem to be linear hence it does not fit the linear regression model well. The third model shows that linear regression model cannot compensate for the outliers that are present and fourth data does not have linear relationship and cannot be represented well with linear regression.

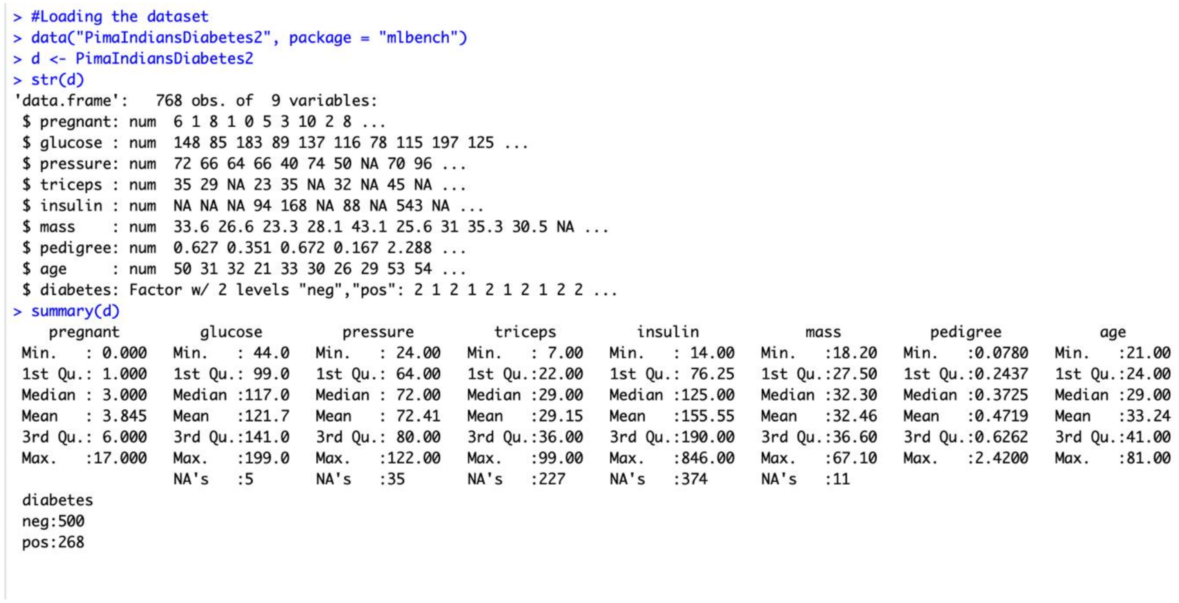
Since not all the data have linear relationship, they are not a good fit for linear regression. After these models are shown on a scatter plot, each dataset produces a unique pattern that regression algorithm cannot explain. From this we can conclude that to truly understand the data, data visualization is very important as it will help you see the distribution of the data and identify outliers, diversity of the data, linear separability of the data.

Question 2

1) (30) TakeData set “PimaIndiansDiabetes2” available in “mlbench” package and remove NAs. Partition the data into 80:20 fashion. Find the coefficients, maximum likelihood, and odds ratio for the predictors using logistic regression on training data set.
   Loading the dataset:



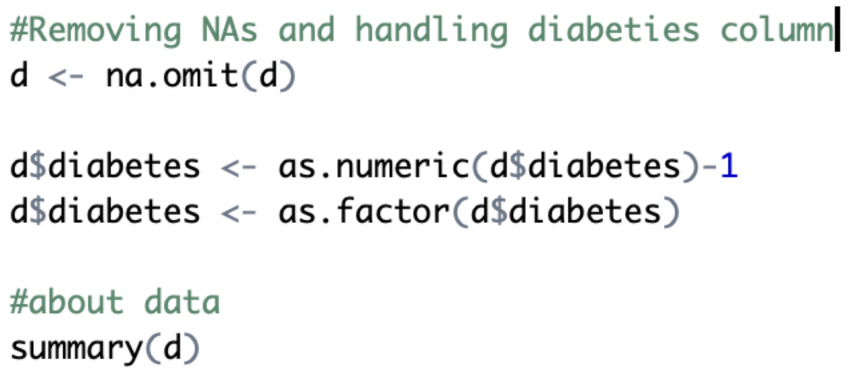
```
#Loading the dataset
data("PimaIndiansDiabetes2", package = "mlbench")
d <- PimaIndiansDiabetes2
str(d)
summary(d)
```



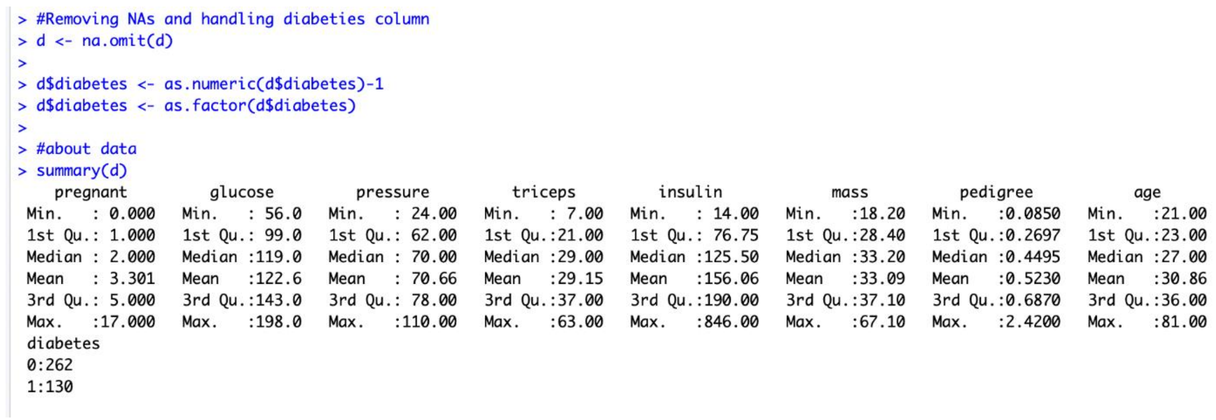
```
> #Loading the dataset
> data("PimaIndiansDiabetes2", package = "mlbench")
> d <- PimaIndiansDiabetes2
> str(d)
'data.frame':   768 obs. of  9 variables:
 $ pregnant: num  6 1 8 1 0 5 3 10 2 8 ...
 $ glucose : num  148 85 183 89 137 116 78 115 197 125 ...
 $ pressure: num  72 66 64 66 40 74 50 NA 70 96 ...
 $ triceps : num  35 29 NA 23 35 NA 32 NA 45 NA ...
 $ insulin : num  NA NA NA 94 168 NA 88 NA 543 NA ...
 $ mass    : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 NA ...
 $ pedigree: num  0.627 0.351 0.672 0.167 2.288 ...
 $ age     : num  50 31 32 21 33 30 26 29 53 54 ...
 $ diabetes: Factor w/ 2 levels "neg","pos": 2 1 2 1 2 1 2 1 2 2 ...
> summary(d)
    pregnant         glucose         pressure         triceps          insulin            mass          pedigree           age       
 Min.   : 0.000   Min.   : 44.0   Min.   : 24.00   Min.   : 7.00   Min.   : 14.00   Min.   :18.20   Min.   :0.0780   Min.   :21.00  
 1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 64.00   1st Qu.:22.00   1st Qu.: 76.25   1st Qu.:27.50   1st Qu.:0.2437   1st Qu.:24.00  
 Median : 3.000   Median :117.0   Median : 72.00   Median :29.00   Median :125.00   Median :32.30   Median :0.3725   Median :29.00  
 Mean   : 3.845   Mean   :121.7   Mean   : 72.41   Mean   :29.15   Mean   :155.55   Mean   :32.46   Mean   :0.4719   Mean   :33.24  
 3rd Qu.: 6.000   3rd Qu.:141.0   3rd Qu.: 80.00   3rd Qu.:36.00   3rd Qu.:190.00   3rd Qu.:36.60   3rd Qu.:0.6262   3rd Qu.:41.00  
 Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00   Max.   :846.00   Max.   :67.10   Max.   :2.4200   Max.   :81.00  
                  NA's   :5       NA's   :35       NA's   :227     NA's   :374      NA's   :11                                       
 diabetes
 neg:500 
 pos:268 
```

The above screenshot shows that the dataset contains 968 values in 9 different attributes. In those attributes eight are numeric variables and one is factor. The diabetes column contains data as negative and positive. The summary function also states the number of NA values which are to be dealt with in each columns.

Missing values:



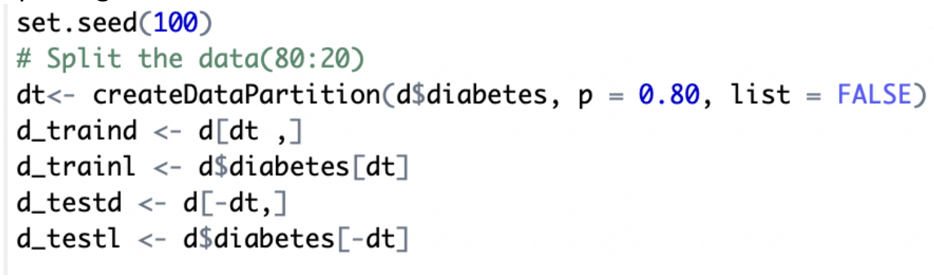
```
#Removing NAs and handling diabeties column
d <- na.omit(d)

d$diabetes <- as.numeric(d$diabetes)-1
d$diabetes <- as.factor(d$diabetes)

#about data
summary(d)
```



```
> #Removing NAs and handling diabeties column
> d <- na.omit(d)
>
> d$diabetes <- as.numeric(d$diabetes)-1
> d$diabetes <- as.factor(d$diabetes)
>
> #about data
> summary(d)
    pregnant         glucose         pressure          triceps          insulin            mass          pedigree            age
 Min.   : 0.000   Min.   : 56.0   Min.   : 24.00   Min.   : 7.00   Min.   : 14.00   Min.   :18.20   Min.   :0.0850   Min.   :21.00
 1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.:21.00   1st Qu.: 76.75   1st Qu.:28.40   1st Qu.:0.2697   1st Qu.:23.00
 Median : 2.000   Median :119.0   Median : 70.00   Median :29.00   Median :125.50   Median :33.20   Median :0.4495   Median :27.00
 Mean   : 3.301   Mean   :122.6   Mean   : 70.66   Mean   :29.15   Mean   :156.06   Mean   :33.09   Mean   :0.5230   Mean   :30.86
 3rd Qu.: 5.000   3rd Qu.:143.0   3rd Qu.: 78.00   3rd Qu.:37.00   3rd Qu.:190.00   3rd Qu.:37.10   3rd Qu.:0.6870   3rd Qu.:36.00
 Max.   :17.000   Max.   :198.0   Max.   :110.00   Max.   :63.00   Max.   :846.00   Max.   :67.10   Max.   :2.4200   Max.   :81.00
 diabetes
 0:262
 1:130
```

Once we remove the NA values using omit() function the summary states there are no NA values. Here we also assign the diabities column as a factor.

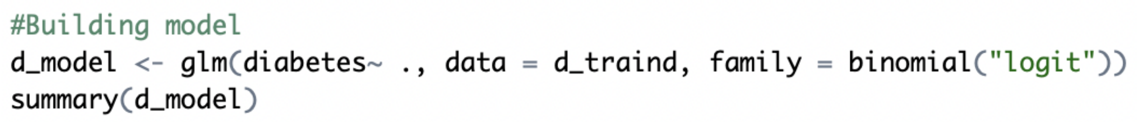
Splitting:



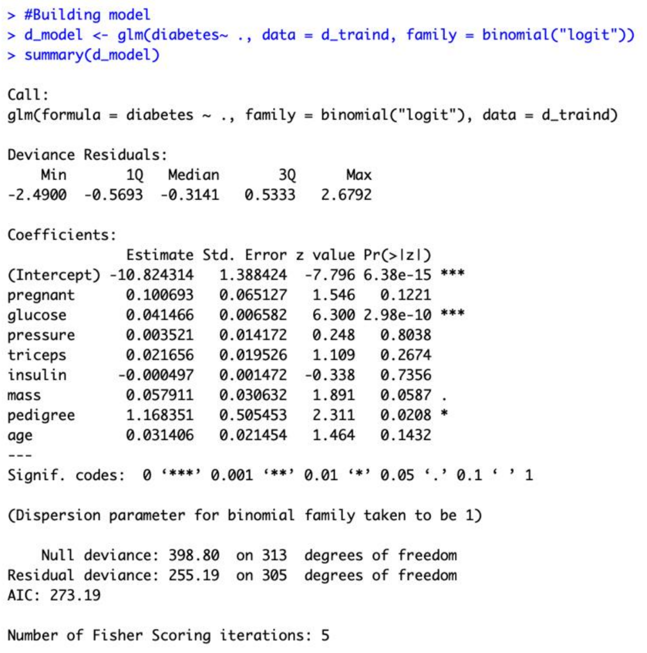
```
set.seed(100)
# Split the data(80:20)
dt<- createDataPartition(d$diabetes, p = 0.80, list = FALSE)
d_traind <- d[dt ,]
d_trainl <- d$diabetes[dt]
d_testd <- d[-dt,]
d_testl <- d$diabetes[-dt]
```

We split the dataset with diabetes as the dependent variable in 80:20 fashion. Here I set the seed to make sure the split is performed in similar fashion for all the times the code is run.

Building Model:



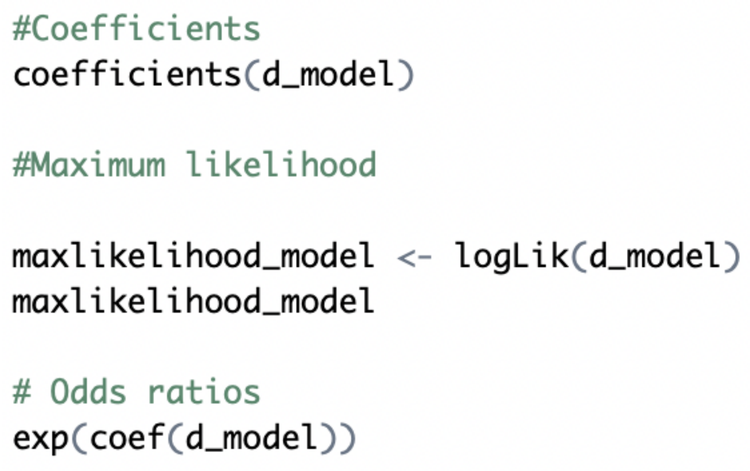
```
#Building model
d_model <- glm(diabetes~ ., data = d_traind, family = binomial("logit"))
summary(d_model)
```



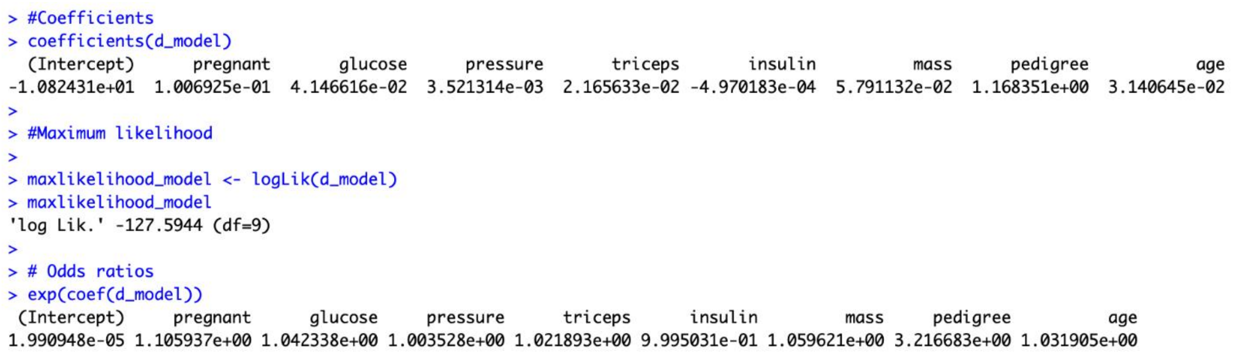
```
> #Building model
> d_model <- glm(diabetes~ ., data = d_traind, family = binomial("logit"))
> summary(d_model)

Call:
glm(formula = diabetes ~ ., family = binomial("logit"), data = d_traind)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4900  -0.5693  -0.3141   0.5333   2.6792

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -10.824314   1.388424  -7.796 6.38e-15 ***
pregnant      0.100693   0.065127   1.546   0.1221
glucose       0.041466   0.006582   6.300 2.98e-10 ***
pressure      0.003521   0.014172   0.248   0.8038
triceps       0.021656   0.019526   1.109   0.2674
insulin      -0.000497   0.001472  -0.338   0.7356
mass          0.057911   0.030632   1.891   0.0587 .
pedigree      1.168351   0.505453   2.311   0.0208 *
age           0.031406   0.021454   1.464   0.1432
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 398.80  on 313  degrees of freedom
Residual deviance: 255.19  on 305  degrees of freedom
AIC: 273.19

Number of Fisher Scoring iterations: 5
```

Looking at the p-value we can conclude that some of the variables are not significant as their p value is low (glucose, pedigree and intercept). Since for the other variables the p value is greater than 0.5 they are significant.

Coefficients, Max likelihood, odds ratio:



```
#Coefficients
coefficients(d_model)

#Maximum likelihood

maxlikelihood_model <- logLik(d_model)
maxlikelihood_model

# Odds ratios
exp(coef(d_model))
```

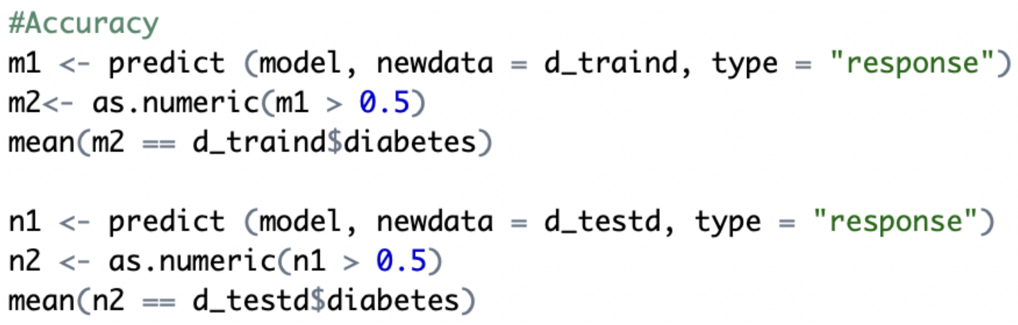


```
> #Coefficients
> coefficients(d_model)
  (Intercept)      pregnant       glucose      pressure       triceps       insulin          mass      pedigree           age 
-1.082431e+01  1.006925e-01  4.146616e-02  3.521314e-03  2.165633e-02 -4.970183e-04  5.791132e-02  1.168351e+00  3.140645e-02 
> 
> #Maximum likelihood
> 
> maxlikelihood_model <- logLik(d_model)
> maxlikelihood_model
'log Lik.' -127.5944 (df=9)
> 
> # Odds ratios
> exp(coef(d_model))
 (Intercept)     pregnant      glucose     pressure      triceps      insulin         mass     pedigree          age 
1.990948e-05 1.105937e+00 1.042338e+00 1.003528e+00 1.021893e+00 9.995031e-01 1.059621e+00 3.216683e+00 1.031905e+00 
```

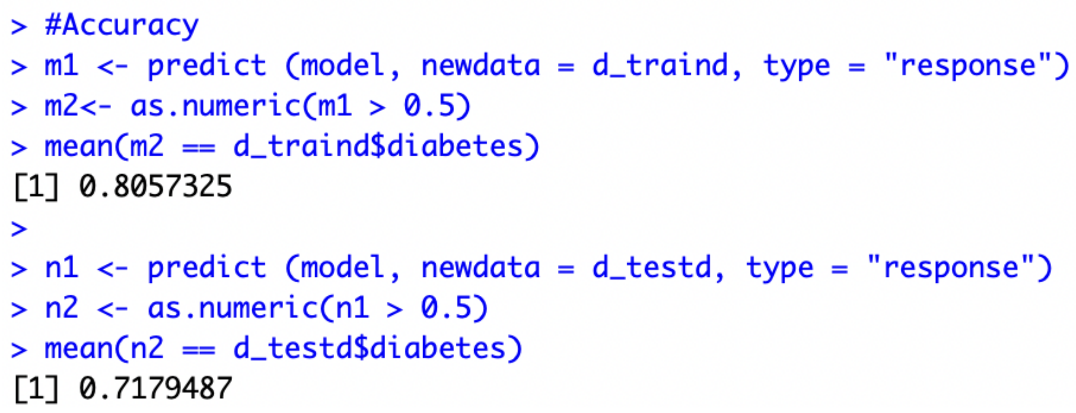
Looking that coefficients, we can conclude that all the coefficients are positive whereas insulin is not and hence it has a negative effect.
Maximum likelihood value is usually calculated to compare this value with other models to see which model is a better fit. The higher the maximum likelihood value the better the model for the dataset.
The odds ratios show which variables contribute to the chance of diabetes occurring. We can see that all the attributes have positive odds ratio hence they contribute to the dependent variable (diabetes).

2) (30) Find accuracy, precision, and recall of the predicted outcomes. Plot the ROC curve. What is the area under the curve?

Accuracy. of the predicted outcomes:



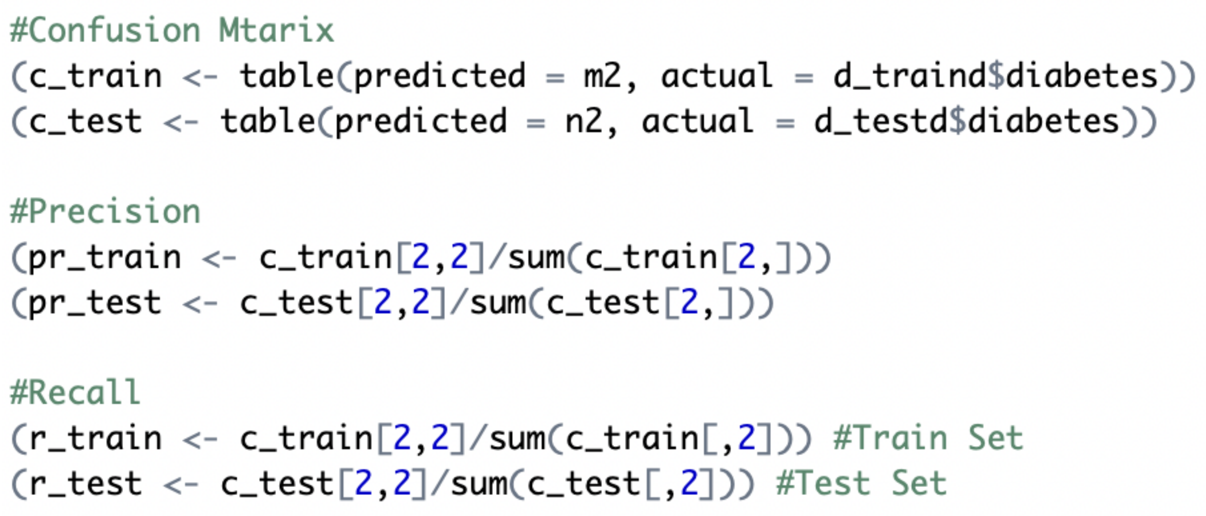
```
#Accuracy
m1 <- predict (model, newdata = d_traind, type = "response")
m2<- as.numeric(m1 > 0.5)
mean(m2 == d_traind$diabetes)

n1 <- predict (model, newdata = d_testd, type = "response")
n2 <- as.numeric(n1 > 0.5)
mean(n2 == d_testd$diabetes)
```



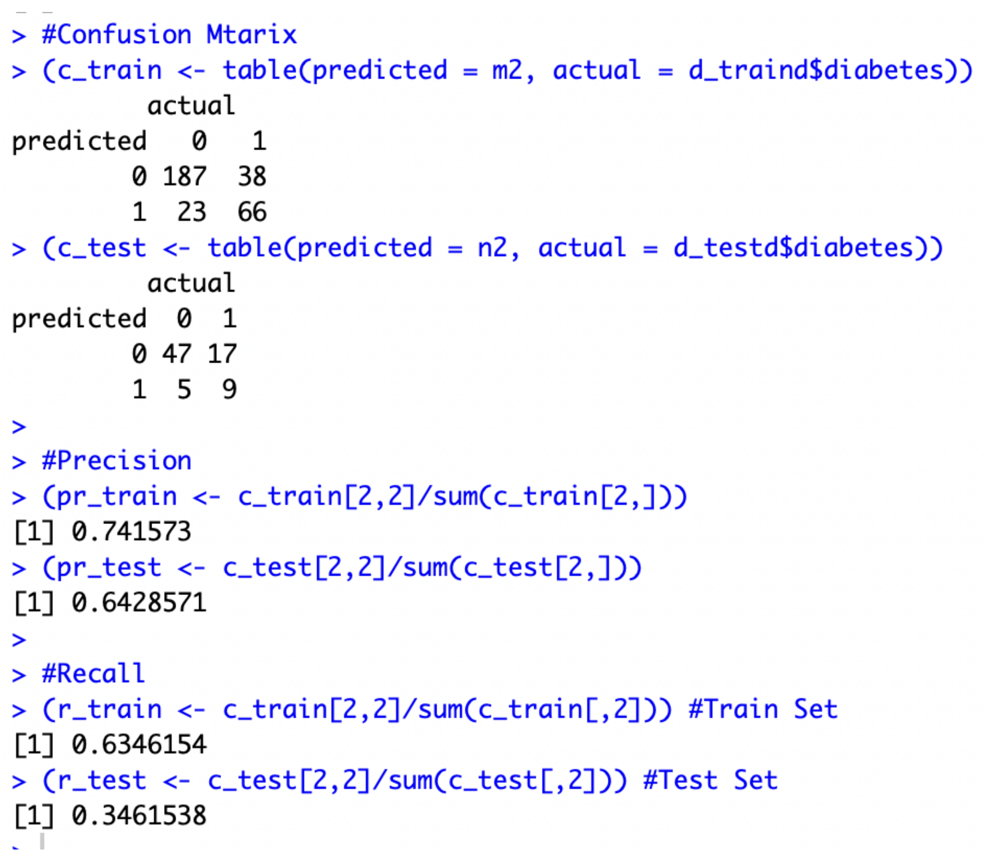
```
> #Accuracy
> m1 <- predict (model, newdata = d_traind, type = "response")
> m2<- as.numeric(m1 > 0.5)
> mean(m2 == d_traind$diabetes)
[1] 0.8057325
> 
> n1 <- predict (model, newdata = d_testd, type = "response")
> n2 <- as.numeric(n1 > 0.5)
> mean(n2 == d_testd$diabetes)
[1] 0.7179487
```

Here we calculate the accuracy which is nothing but the number of correctly predicted data points out of all the data points. The accuracy of test data (71.79%) is quite lower than that of train (80.5%)

Confusion Matrix ,Precision, recall of the predicted outcomes:



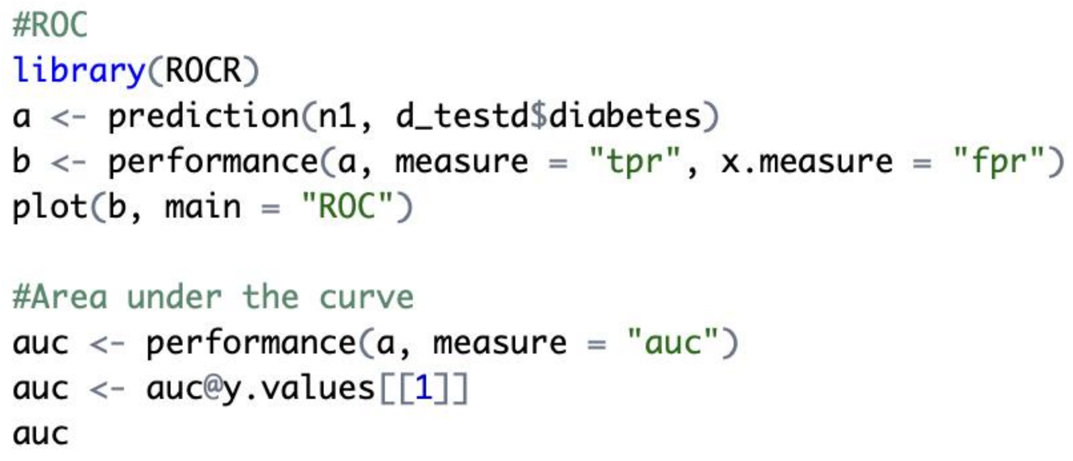
```
#Confusion Mtarix
(c_train <- table(predicted = m2, actual = d_traind$diabetes))
(c_test <- table(predicted = n2, actual = d_testd$diabetes))

#Precision
(pr_train <- c_train[2,2]/sum(c_train[2,]))
(pr_test <- c_test[2,2]/sum(c_test[2,]))

#Recall
(r_train <- c_train[2,2]/sum(c_train[,2])) #Train Set
(r_test <- c_test[2,2]/sum(c_test[,2])) #Test Set
```



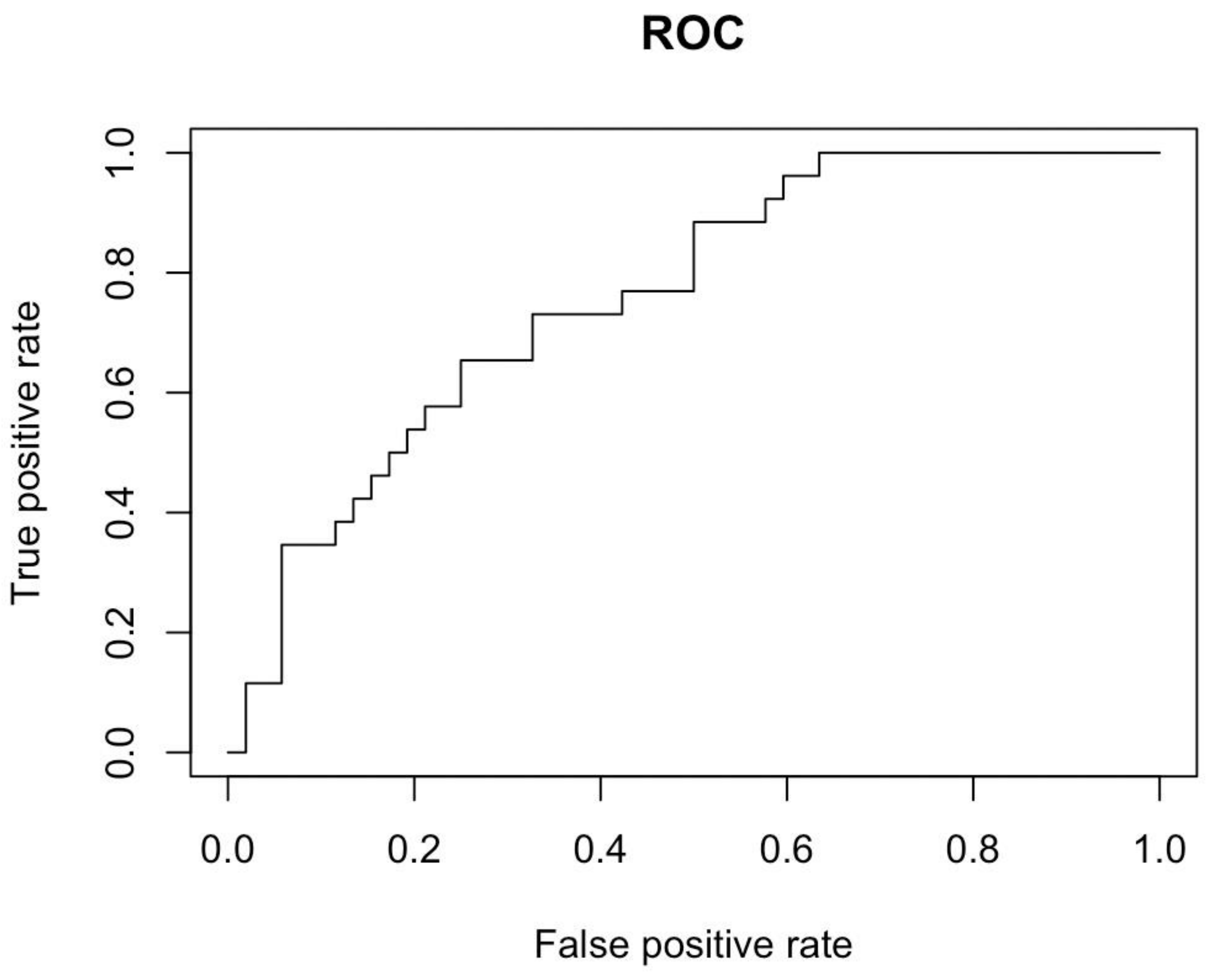
```
> #Confusion Mtarix
> (c_train <- table(predicted = m2, actual = d_traind$diabetes))
         actual
predicted   0   1
        0 187  38
        1  23  66
> (c_test <- table(predicted = n2, actual = d_testd$diabetes))
         actual
predicted  0  1
        0 47 17
        1  5  9
>
> #Precision
> (pr_train <- c_train[2,2]/sum(c_train[2,]))
[1] 0.741573
> (pr_test <- c_test[2,2]/sum(c_test[2,]))
[1] 0.6428571
>
> #Recall
> (r_train <- c_train[2,2]/sum(c_train[,2])) #Train Set
[1] 0.6346154
> (r_test <- c_test[2,2]/sum(c_test[,2])) #Test Set
[1] 0.3461538
```

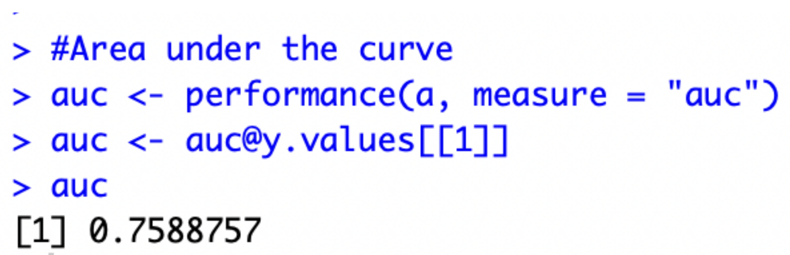
The confusion matrix is 2*2 matrix it is used for evaluating the performance of a classification model. It is used to compare the actual target values with predicted values. Here for train data, we have 187 true negative (predicted and actual are 0), 66 true positive (predicted and actual are 1), 23 false positive (actual value was 0 but the model predicted 1), and 38 false negatives (actual value was 1 but the model predicted 0). Similarly, we can do the same analysis for confusion matrix on test data.
Since our data was imbalanced accuracy would not be a great evaluation matrix as it shows how many times it correctly predicted the diabetes cases, instead of telling how many of the correctly predicted cases actually turned out to be diabetic (1) (for this we will be using precision) or how many of the actual positive cases we were able to predict correctly with our model (for this we will be calculation recall).
For test data:
We have precision as 64% (percentage of predicted diabetes actually turned to be diabetic)
And recall of 34% (percentage of actual diabetic customers that were predicted correctly)
For our case recall is more important as false alarm isn't a bigger concern, but the actual positive cases should not go undetected. With recall of just 34% our model is not performing well on test data.

ROC and AUC of the predicted outcomes:



```
#ROC
library(ROCR)
a <- prediction(n1, d_testd$diabetes)
b <- performance(a, measure = "tpr", x.measure = "fpr")
plot(b, main = "ROC")

#Area under the curve
auc <- performance(a, measure = "auc")
auc <- auc@y.values[[1]]
auc
```





```
> #Area under the curve
> auc <- performance(a, measure = "auc")
> auc <- auc@y.values[[1]]
> auc
[1] 0.7588757
```

The ROC curve is plotted with true positive rate against the false positive rate (TPR is on the y-axis and FPR is on the x-axis).
The AUC - ROC curve is an evaluation metric for classifying at various thresholds. The better the model predicts 0 courses as 0 and 1 classes as 1, the higher the AUC. For our model we have AUC as 75.8%, which indicates our model can differentiate between negatives and positives for diabetes with 75.88% prediction power.