21205992 Predictive Analytics Assignment1 Solution

Vivek Bulani 21205992

11/1/2021

Explotory Data Analysis

Q1]

```
df=read.csv("/Users/vivekbulani/Documents/UCD Sem 1 Modules/UCD Predictive Analytics/
data_data_set.csv")
summary(df)                                #returns statistical summary for all column
s
```

```
##      Price           Area              bathroom         bedroom
##  Min.   :  480   Length:2413        Min.   : 0.000   Min.   :0.000
##  1st Qu.: 1600   Class :character   1st Qu.: 1.000   1st Qu.:1.000
##  Median : 1900   Mode  :character   Median : 2.000   Median :1.000
##  Mean   : 2069                      Mean   : 1.869   Mean   :1.442
##  3rd Qu.: 2295                      3rd Qu.: 2.000   3rd Qu.:2.000
##  Max.   :33600                      Max.   :12.000   Max.   :9.000
##  property.type
##  Length:2413
##  Class :character
##  Mode  :character
##
##
##
```

```
str(df)                                    #returns datatype of each column along with
values
```

```
## 'data.frame':    2413 obs. of  5 variables:
##  $ Price        : int  1650 1924 2100 1800 1200 1800 2051 1650 1800 1650 ...
##  $ Area         : chr  "South Dublin City" "West Co. Dublin" "South Dublin City"
"South Dublin City" ...
##  $ bathroom     : int  2 3 3 1 0 2 2 1 2 1 ...
##  $ bedroom      : int  1 2 1 1 0 1 2 1 2 1 ...
##  $ property.type: chr  "Apartment" "House" "House" "Apartment" ...
```

Hence Price, bathroom and bedroom are numeric variables. Whereas Area and property type are categorical variables. Hence we need to convert Area and property type into factors using as.factor()

```
dim(df)                                    #returns number of rows and columns in data
set
```

```
## [1] 2413    5
```

```
colnames(df)                                #returns names of all columns
```

```
## [1] "Price"         "Area"          "bathroom"      "bedroom"
## [5] "property.type"
```

```
df$Area=as.factor(df$Area)                #converts non numeric Area column into fact
ors having 7 levels
summary(df)
```

```
##      Price                       Area        bathroom         bedroom
##  Min.   :  480   Co. Dublin        :  3   Min.   : 0.000   Min.   :0.000
##  1st Qu.: 1600   Dublin City Centre:631   1st Qu.: 1.000   1st Qu.:1.000
##  Median : 1900   North Co. Dublin  : 99   Median : 2.000   Median :1.000
##  Mean   : 2069   North Dublin City :430   Mean   : 1.869   Mean   :1.442
##  3rd Qu.: 2295   South Co. Dublin  :205   3rd Qu.: 2.000   3rd Qu.:2.000
##  Max.   :33600   South Dublin City :893   Max.   :12.000   Max.   :9.000
##                  West Co. Dublin   :152
##  property.type
##  Length:2413
##  Class :character
##  Mode  :character
##
##
##
##
```

```
str(df)
```

```
## 'data.frame':    2413 obs. of  5 variables:
##  $ Price        : int  1650 1924 2100 1800 1200 1800 2051 1650 1800 1650 ...
##  $ Area         : Factor w/ 7 levels "Co. Dublin","Dublin City Centre",..: 6 7 6 6
6 6 7 2 7 2 ...
##  $ bathroom     : int  2 3 3 1 0 2 2 1 2 1 ...
##  $ bedroom      : int  1 2 1 1 0 1 2 1 2 1 ...
##  $ property.type: chr  "Apartment" "House" "House" "Apartment" ...
```

```
df$property.type=as.factor(df$property.type)
#converts non numeric Property_type column into factors having 4 levels

summary(df)
```

```
##      Price                          Area        bathroom         bedroom      
##  Min.   :  480   Co. Dublin         :  3   Min.   : 0.000   Min.   :0.000  
##  1st Qu.: 1600   Dublin City Centre:631   1st Qu.: 1.000   1st Qu.:1.000  
##  Median : 1900   North Co. Dublin  : 99   Median : 2.000   Median :1.000  
##  Mean   : 2069   North Dublin City :430   Mean   : 1.869   Mean   :1.442  
##  3rd Qu.: 2295   South Co. Dublin  :205   3rd Qu.: 2.000   3rd Qu.:2.000  
##  Max.   :33600   South Dublin City :893   Max.   :12.000   Max.   :9.000  
##                  West Co. Dublin   :152                                   
##    property.type
##  Apartment:1748
##  Flat     :  74
##  House    : 429
##  Studio   : 162
##                
##                
##
```

```
str(df)
```

```
## 'data.frame':    2413 obs. of  5 variables:
##  $ Price        : int  1650 1924 2100 1800 1200 1800 2051 1650 1800 1650 ...
##  $ Area         : Factor w/ 7 levels "Co. Dublin","Dublin City Centre",..: 6 7 6 6
6 6 7 2 7 2 ...
##  $ bathroom     : int  2 3 3 1 0 2 2 1 2 1 ...
##  $ bedroom      : int  1 2 1 1 0 1 2 1 2 1 ...
##  $ property.type: Factor w/ 4 levels "Apartment","Flat",..: 1 3 3 1 4 1 1 1 1 1
...
```

Now categorical variables are converted to factors.

Q2]

```
df[rowSums(is.na(df))>0,]
```

```
## [1] Price         Area          bathroom      bedroom       property.type
## <0 rows> (or 0-length row.names)
```

```
#prints rows having at least 1 NA. But in my data set, there are no NA values
```

```
print(paste("The number of rows having missing values are",nrow(df[rowSums(is.na(df))
>0,])))
```

```
## [1] "The number of rows having missing values are 0"
```

```
#return the number of rows having missing values in the data set.
```

```
df=df[!rowSums(is.na(df))>0,]
```

```
dim(df)
```

```
## [1] 2413    5
```

```
#As there are no NA values, hence number of rows remains same as before.
```

Q3]

```
#df[duplicated(df),]                        #prints rows having duplicate values
```

```
print(paste("The number of duplicate rows in the dataset is",nrow(df[duplicated(d
f),])))
```

```
## [1] "The number of duplicate rows in the dataset is 1288"
```

```
print(paste("The total number of rows in the dataset before removing duplicate rows i
s",nrow(df)))
```

```
## [1] "The total number of rows in the dataset before removing duplicate rows is 241
3"
```

```
df=df[!duplicated(df),]                     #removes all duplicate rows
```

```
print(paste("The total number of rows left in the dataset after removing duplicate ro
ws is",nrow(df)))
```

```
## [1] "The total number of rows left in the dataset after removing duplicate rows is
1125"
```

Q4]

1. There might be some situations when the range of data is very large and it becomes difficult to handle such data. as an example suppose some variables has values ranging from -10000000 to +10000000. Analyzing and visualizing such a variable would be very complex. In such situation it would be useful to transform it into a new range which is easy to handle.
2. Sometimes the distribution of a numeric variable can be very complex or it might have distribution which we are not familiar of and have no idea about its statistical formulas. Hence performing statistical analysis on such distribution could be very challenging and complex. In such case, it would be better if we transform this numeric variable into a new range such that its distribution becomes easy and familiar.
3. Also it is a usual case that variables that are having different scales/ranges may contribute differently to the model. Hence in such cases it might be necessary to transform some numeric columns into desired range for proper model building.

Examples of Data Transformations :-

a. Log Transform :-

1. Normal distribution is one of the most popular distribution and easy to work with. Suppose some distribution is not following the bell curve structure. Instead it is either a left or right skewed data.

Working with such data would be very difficult.
2. In such case, we can perform log operation on the data such that the final distribution almost follows a bell curve.
3. But one of the requirement to apply this transformation is that original data should follow log-normal distribution. Otherwise the result would not be helpful.

b. Min-Max Transform :-

1. In his method, we usually scale the original data such that it has new minimum as 0 and new maximum as 1. By doing so, it becomes a lot easier for handling multiple such variables and results in a good model building.
2. By scaling the data to a range of [0,1], it also helps to develop more informative and clean visualizations which can further help in analysis.
3. It should be noted that this method is sensitive to outliers.

Q5]

1. Consider the example of Age variable. For reporting purposes, it is always beneficial that we convert this continuous data into discrete values/categories (such as less than 20, 20-50, 50-80, more than 80).
2. This would help us to group data and identify characteristics of each group and accordingly draw conclusions.
3. Also for such categories, we can then use plots such as histograms, bar plots, etc for clear understanding and better explanation.
4. Hence, in general, for reporting purposes, it is always beneficial to have discrete data rather than continuous data.

Q6]

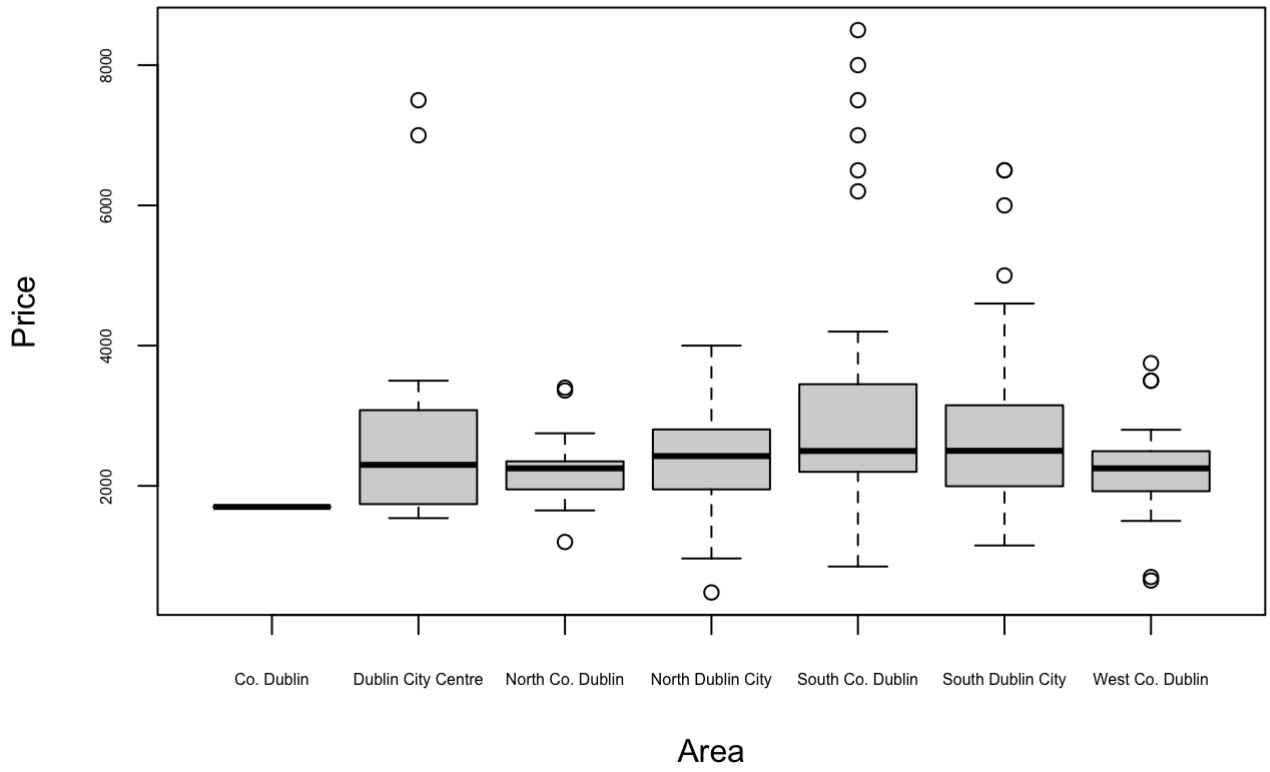
```
df_House1=df$Price[df$property.type=="House"]             #create a new data frame of
Price column for just Houses.
df_House2=df$Area[df$property.type=="House"]              #create a new data frame of
Area column for just Houses.

by(df_House1,df_House2,summary)
```

```
## df_House2: Co. Dublin
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1700    1700    1700    1700    1700    1700
## ------------------------------------------------------------
## df_House2: Dublin City Centre
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1540    1749    2300    2970    3060    7500
## ------------------------------------------------------------
## df_House2: North Co. Dublin
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1200    1956    2250    2252    2350    3400
## ------------------------------------------------------------
## df_House2: North Dublin City
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     480    1950    2425    2453    2802    4000
## ------------------------------------------------------------
## df_House2: South Co. Dublin
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     850    2200    2498    3203    3225    8500
## ------------------------------------------------------------
## df_House2: South Dublin City
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1150    1996    2500    2726    3150    6500
## ------------------------------------------------------------
## df_House2: West Co. Dublin
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     650    1924    2250    2220    2495    3750
```

1. Mean is the value which can represent the entire data values. Hence in some cases, when there are missing values, we replace them by the mean because mean is the best representation of data.
2. Median is the value which divides the data approximately into 2 halves, that is it divides the data into 50% each.
3. 1st Quartile indicates that 25% of data lies below it when data is sorted in increasing order.
4. 3rd Quartile indicates that 75% of data lies below it when data is sorted in increasing order.
5. From above summary, we can infer that Prices are relatively high in South Co. Dublin and Dublin City Centre.
6. Whereas the Price of houses is cheaper in Co. Dublin and West Co. Dublin.
7. Also we can see that there is no any information about houses in Co. Dublin.
8. Mean of Prices of houses in almost areas lies roughly between 2000 and 3000.

Q7]

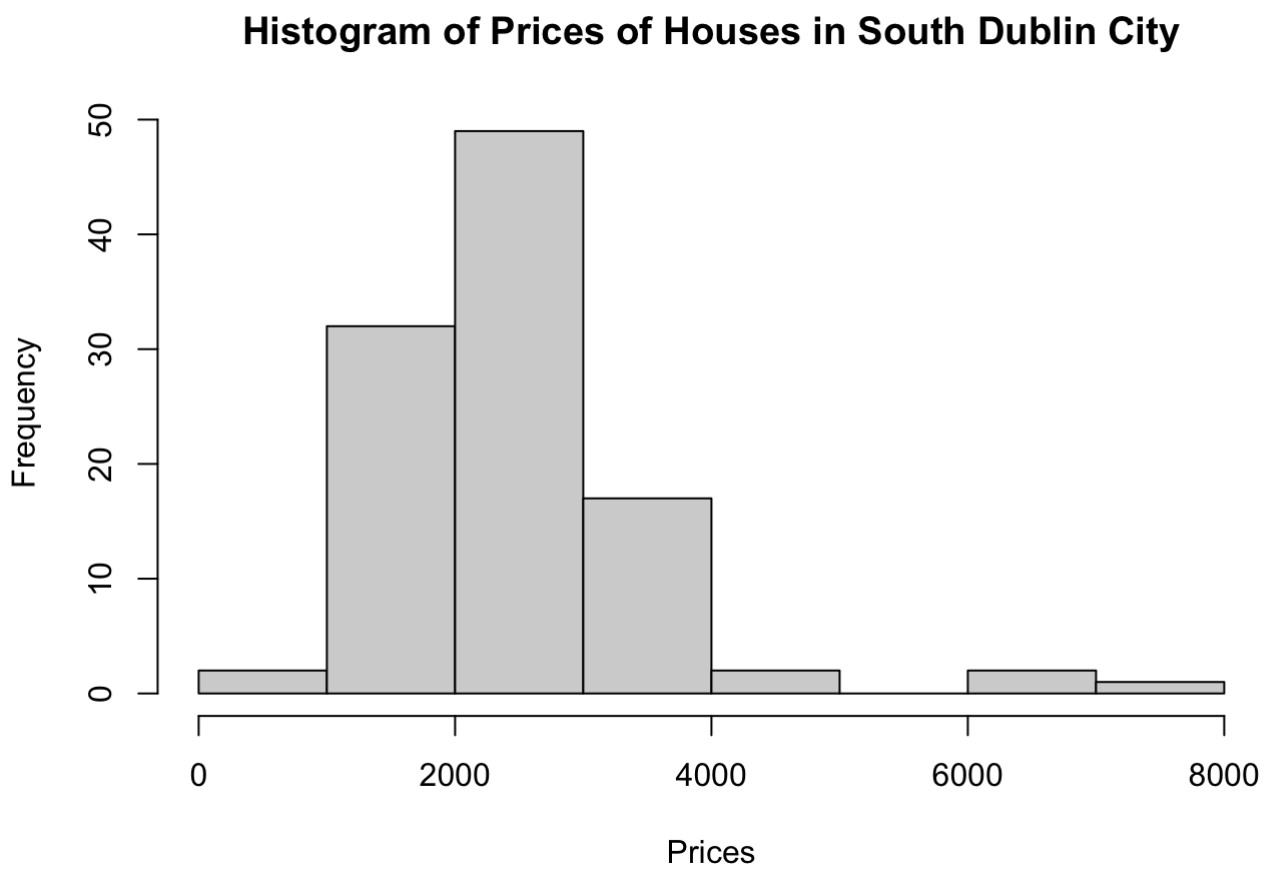
```
boxplot(df_House1~df_House2 , xlab="Area" , ylab="Price" , main="Price of Houses vs A
rea", cex.axis=0.5)
```



1)

From the histogram we can see that no information is there about houses in Co. Dublin. 2) Almost for all areas, median of Prices lies roughly between 2000 and 3000. 3) There are relatively many outliers (points outside whiskers) for Dublin City Centre, South Co. Dublin and South Dublin City. These outliers should be removed for better and accurate analysis. 4) There are many houses in Dublin City Centre, South Co. Dublin and South Dublin City as compared to other areas.

```
hist(df_House1[df$Area=="South Dublin City"], main = "Histogram of Prices of Houses i
n South Dublin City", xlab = "Prices")              #histogram of prices of houses i
n South Dublin City
```




```
hist(df_House1[df$Area=="Dublin City Centre"], main = "Histogram of Prices of Houses
 in Dublin City Centre", xlab = "Prices")              #histogram of prices of house
s in Dublin City Centre
```

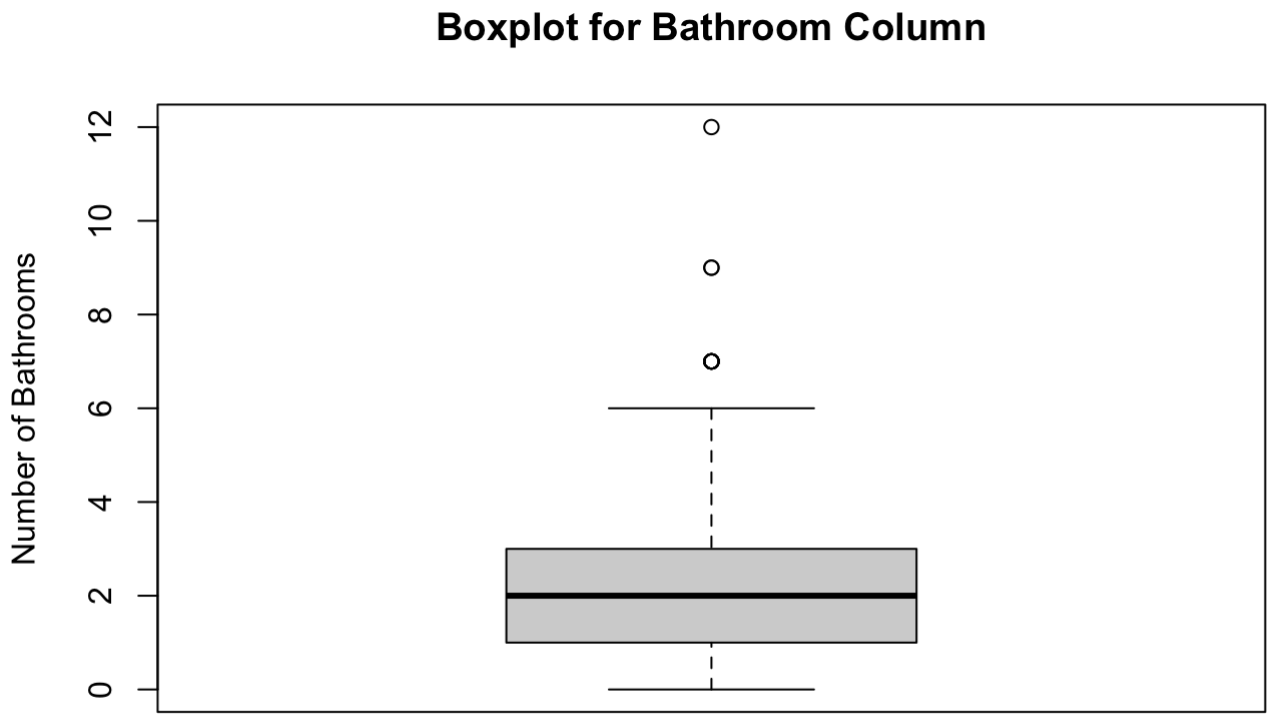
Similarly we can draw histograms for all Areas.

Q8]

1. A histogram can present data to a user which is misleading. For example, if we use too many bins/blocks, it makes the analysis difficult, whereas if we use too few bins/blocks, it can leave out important information.
2. As we change the bin size/width, the histograms change drastically, that is the y-values change a lot. Hence it becomes difficult to present proper and accurate histogram to the end user/client.
3. Different people can come up with different histograms based on bin size/widths. This may lead to confusion and finally hamper statistical analysis.
4. Histograms are useful only when there are two variables under observation. If there are more variables to be considered simultaneously then histograms cannot be used.
5. Histograms break the data into bins. This may lead to loss of meaningful information.

Q9]

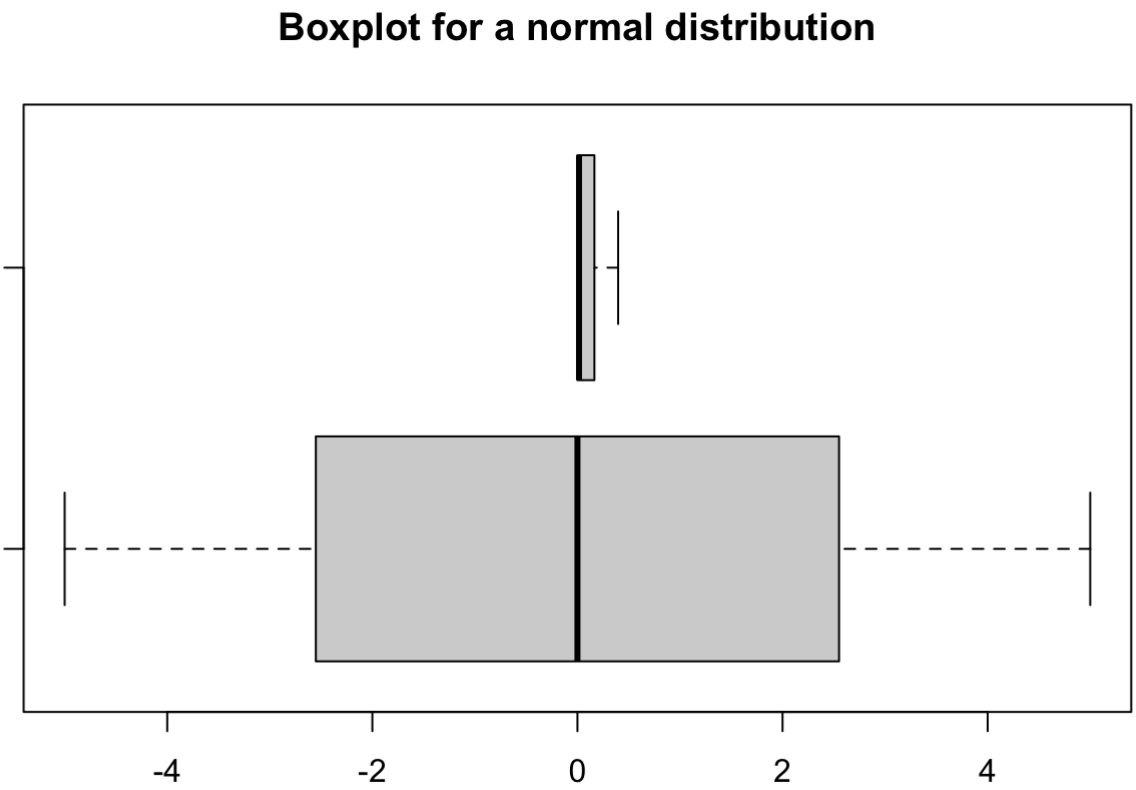
```
boxplot(df$bathroom, main="Boxplot for Bathroom Column", ylab = "Number of Bathrooms"
)
```



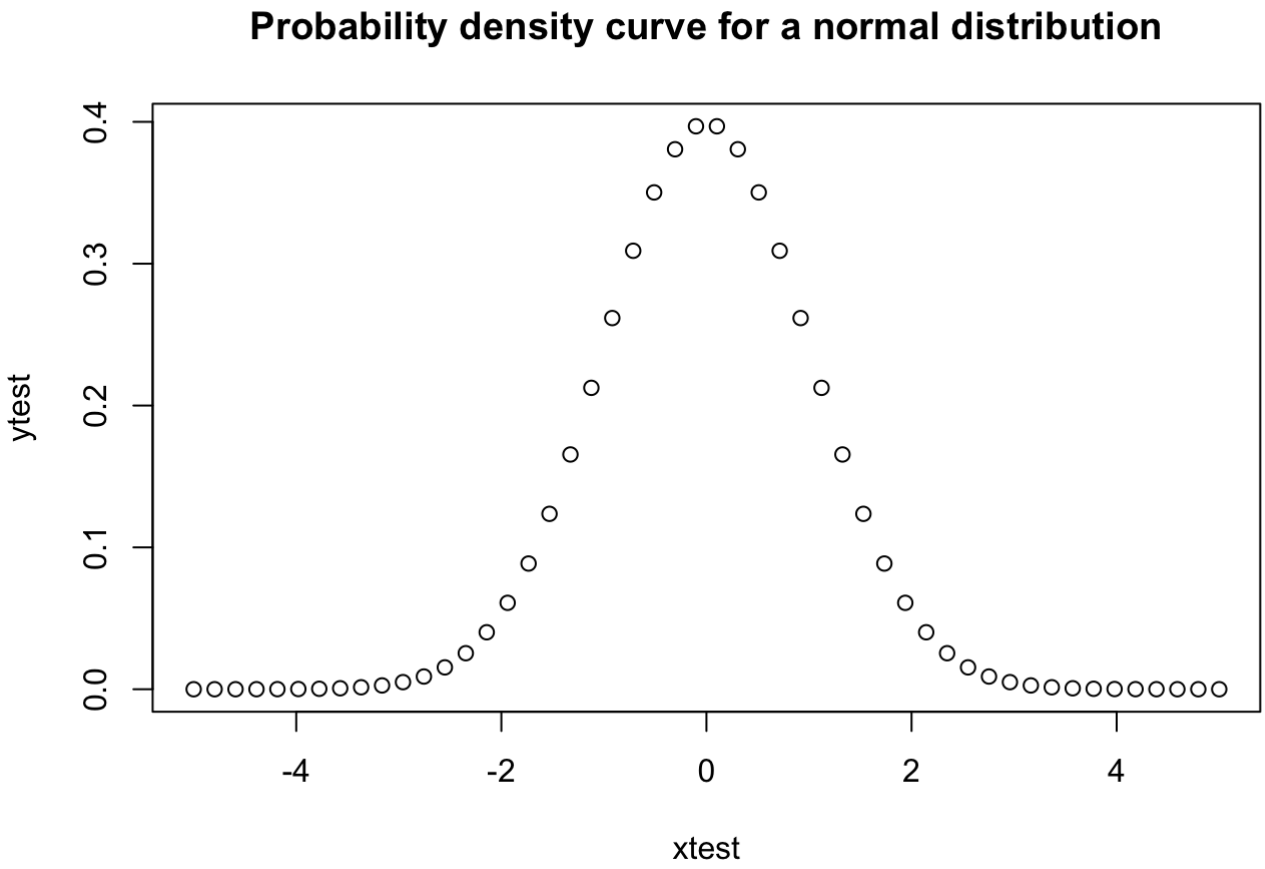
1. Here the bottom most line is the lower whisker, which is at around 0 in above plot.
2. Then comes the 1st quartile (the lower boundary of box) which is here at around 1. There are roughly 25% values between lower whisker and 1st quartile.
3. Then the dark line in the middle of the box is called the median. It indicates that 50% values lie under it when data is sorted in increasing order. Here the median is roughly at 2.
4. Then the upper boundary of box is called 3rd quartile. Here it is roughly at 3. It indicates that 75% values lie under it. Hence total 50% values lie between 1st quartile and 3rd quartile.
5. Then comes the upper line, called upper whisker, which is at 6.
6. Points below the lower whisker and above the upper whisker are called outliers and need to be removed for proper and accurate data analysis.

Q10]

```
xtest = seq(-5,5,length=50)
ytest = dnorm(xtest,0,1)
boxplot(xtest,ytest,horizontal=TRUE, main = "Boxplot for a normal distribution")
#Plot the boxplot
```




```
plot(xtest,ytest,main = "Probability density curve for a normal distribution")
#Plot the probability density curve
```



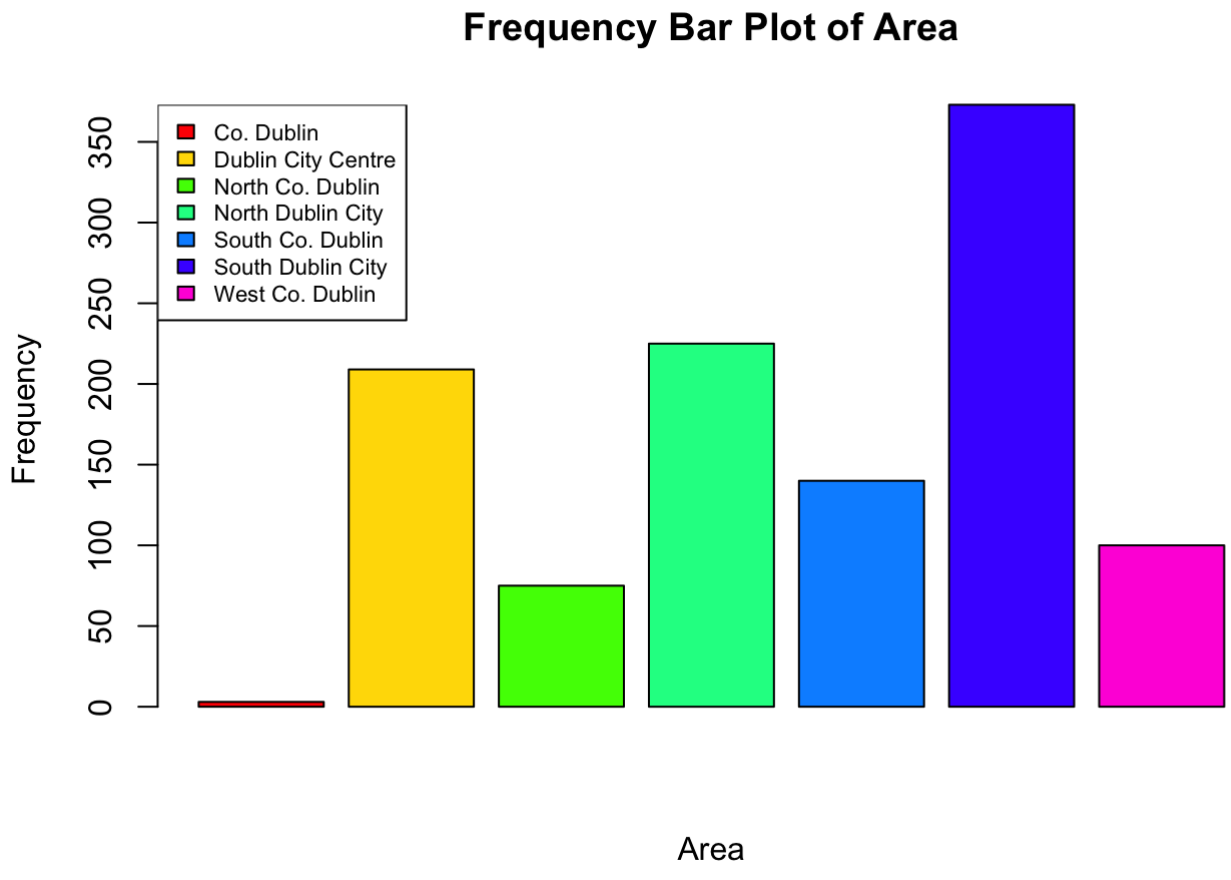
1. After comparing the 2 graphs, we see that 1st quartile is at approximately -2.25. At around the same value, the probability density curve goes to zero on left hand side.
2. Similarly, the 3rd quartile is at approximately +2.25. At around the same value, the probability density curve goes to zero on right hand side.
3. Hence we can say that 50% values lie between 1st quartile and 3rd quartile.
4. The lower whisker is roughly at around -5 and upper whisker is approximately around +5.
5. 25% values lie between 1st quartile and lower whisker, and 25% values lie between 3rd quartile and upper whisker.
6. Also we can say that for a normal distribution, the probability density curve as well as box plot are symmetric.

Q11]

```
table(df$Area)                                    #Frequencies based on different Are
as.
```

```
##
##          Co. Dublin Dublin City Centre   North Co. Dublin  North Dublin City
##                   3                 209                 75                 225
##   South Co. Dublin  South Dublin City    West Co. Dublin
##                 140                 373                 100
```

```
barplot(table(df$Area),xlab = "Area", ylab = "Frequency", main = "Frequency Bar Plot
 of Area", col = rainbow(7), xaxt='n')
legend("topleft", legend=rownames(table(df$Area)), fill = rainbow(7),cex=0.7)
```

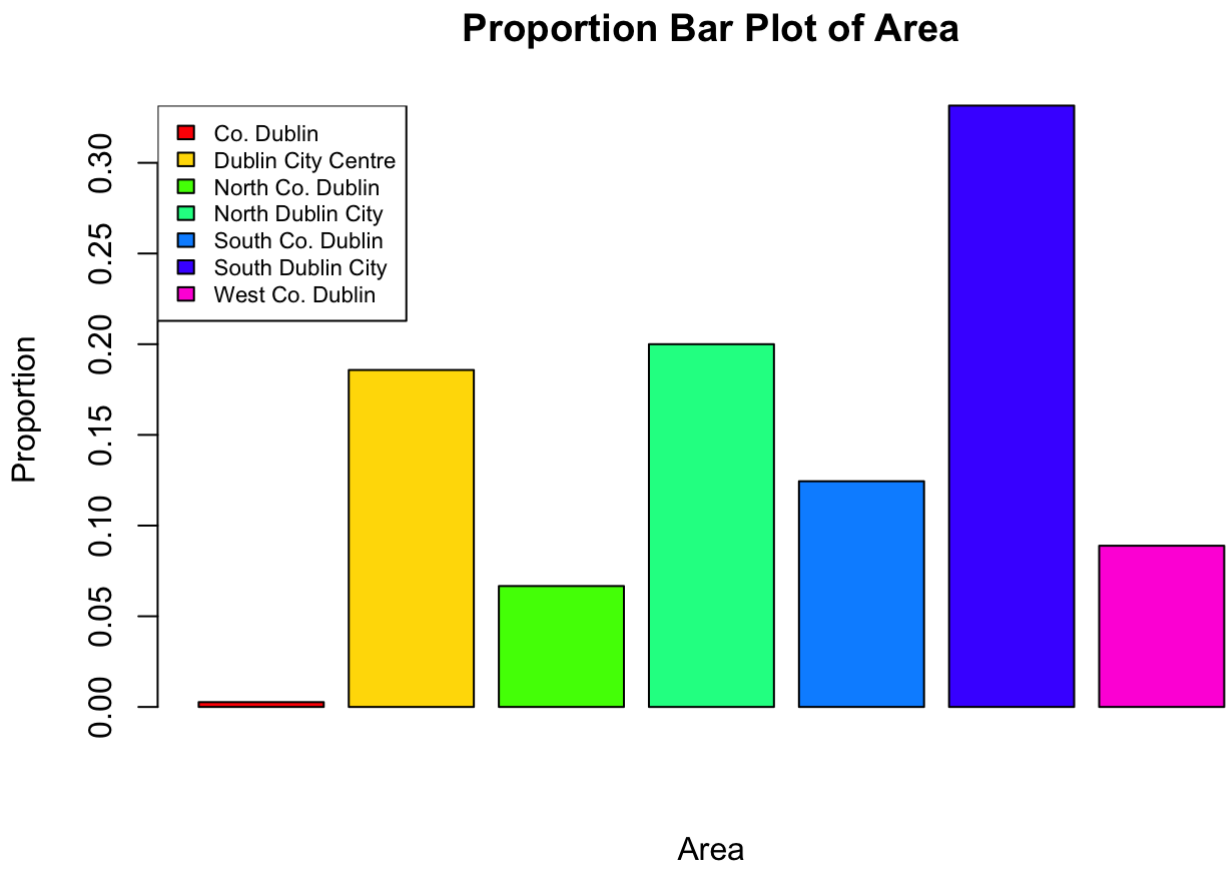


Here we can see that there are highest number of rows for South Dublin City, followed by North Dublin City and Dublin City Centre. Least/Negligible data is present for Co. Dublin.

```
prop.table(table(df$Area))                    #Proportion of data based on different
 Areas.
```

```
##
##          Co. Dublin Dublin City Centre   North Co. Dublin  North Dublin City
##         0.002666667        0.185777778        0.066666667        0.200000000
##    South Co. Dublin  South Dublin City    West Co. Dublin
##         0.124444444        0.331555556        0.088888889
```

```
barplot(prop.table(table(df$Area)) , xlab = "Area" , ylab = "Proportion", main = "Pro
portion Bar Plot of Area" , col = rainbow(7), xaxt='n')
legend("topleft", legend=rownames(table(df$Area)), fill = rainbow(7),cex=0.7)
```



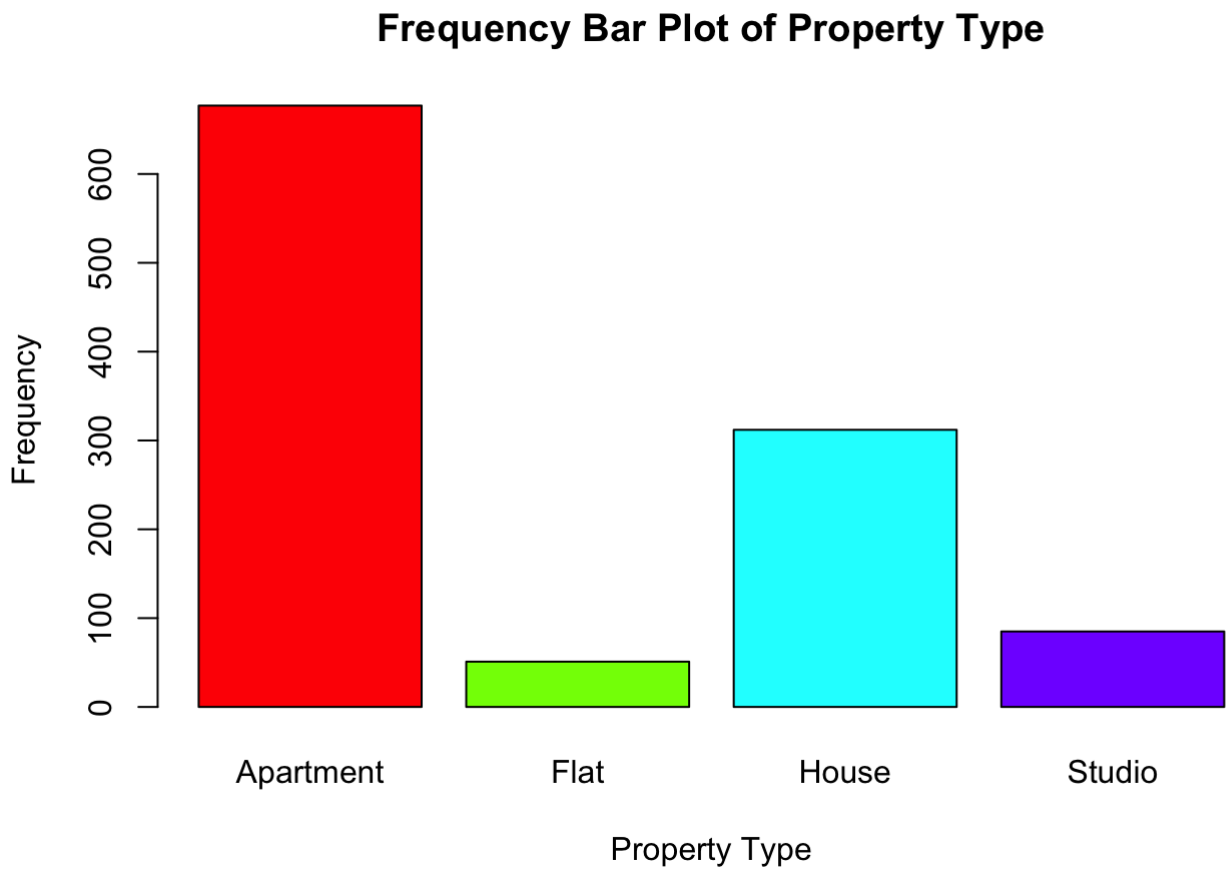
Proportion scales the frequency values between 0 and 1 for better comparison. We can relatively compare using proportion values. From above graph, we can see that approximately more than 30% data is for South Dublin City.

Q12]

```
table(df$property.type)                                #Frequencies based on differen
t Property Types
```

```
##
## Apartment      Flat     House    Studio
##       677        51       312        85
```

```
barplot(table(df$property.type),xlab = "Property Type", ylab = "Frequency", main = "F
requency Bar Plot of Property Type", col = rainbow(4))
```

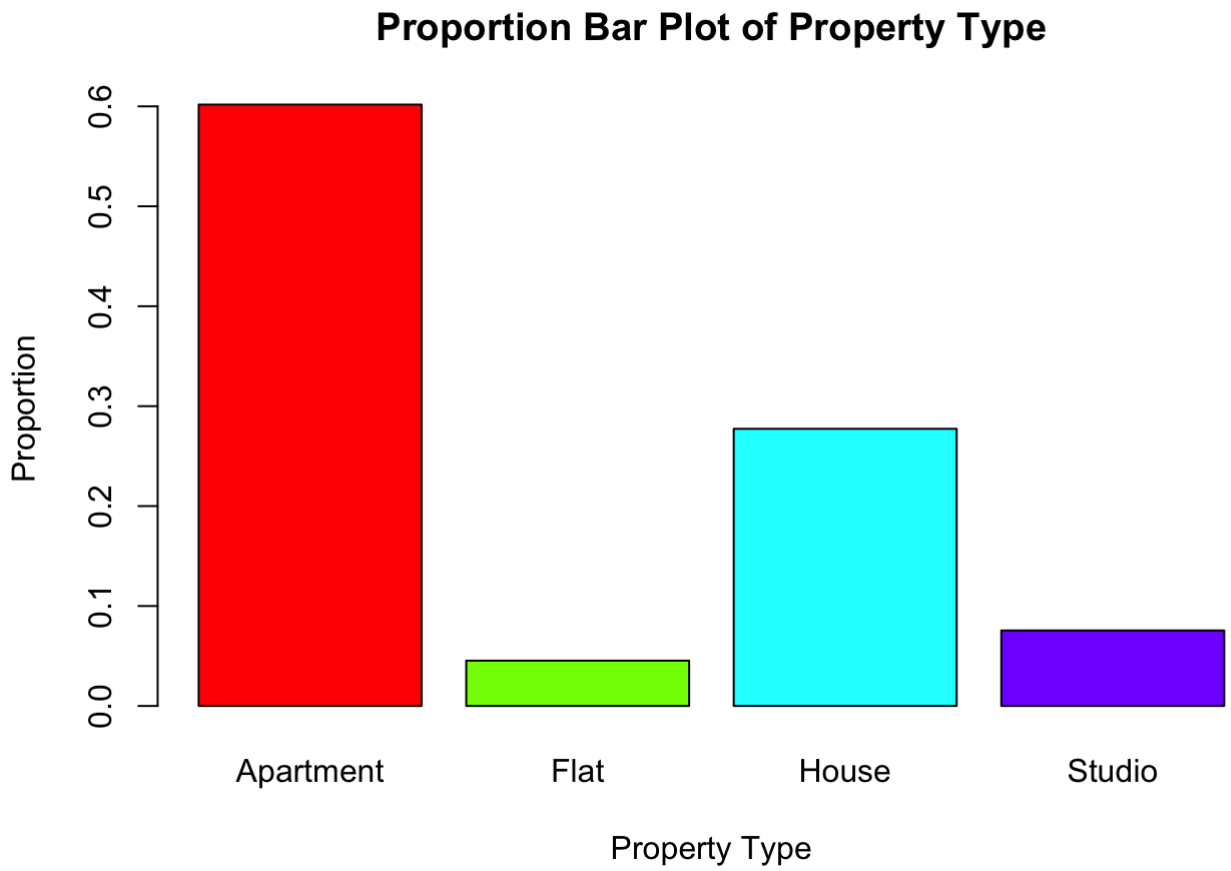


Here we can see that in the data, most of the property are of type Apartment. There are only a few Flats and Studio type of property.

```
prop.table(table(df$property.type))                    #Proportion of data based on d
ifferent Property Types
```

```
##
##  Apartment       Flat      House     Studio
## 0.60177778 0.04533333 0.27733333 0.07555556
```

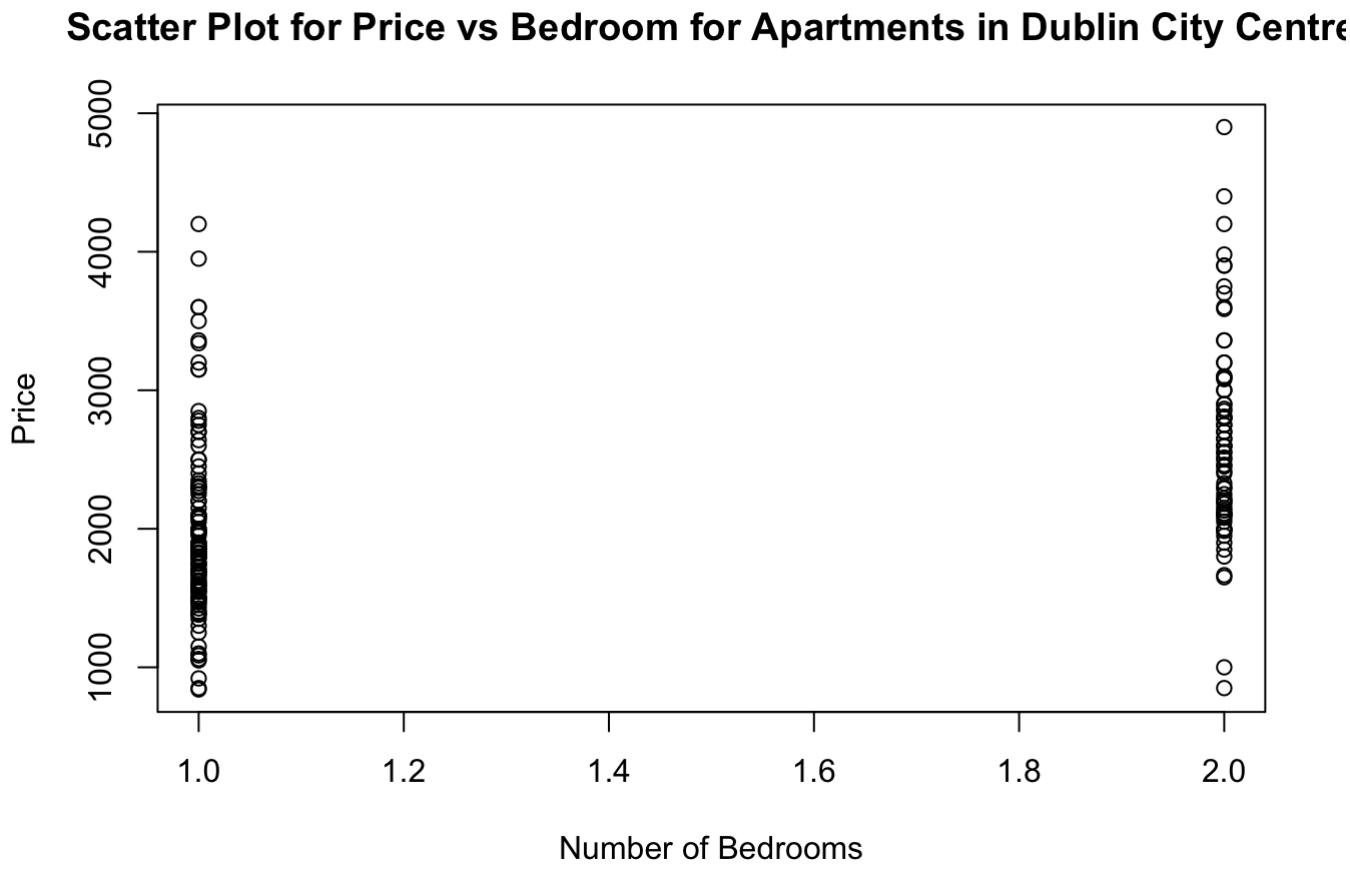
```
barplot(prop.table(table(df$property.type)),xlab = "Property Type", ylab = "Proportio
n", main = "Proportion Bar Plot of Property Type", col = rainbow(4))
```



From above graph, we can infer that approximately 60% data is for Apartment type of property, followed by 30% of House type of property. Flat and Studio together combine to form 10% of data.

Q13]

```
df1=df[df$Area=="Dublin City Centre" & df$property.type=="Apartment" , ]
plot(df1$bedroom,df1$Price,main = "Scatter Plot for Price vs Bedroom for Apartments i
n Dublin City Centre", xlab = "Number of Bedrooms", ylab = "Price")
```



From the plot, we can see that as number of bedrooms increases, price also increases. Hence there is a significant positive correlation between the 2 variables.

```
cor_value=cor(df1$bedroom,df1$Price)
print(paste("The correlation value between Price and number of Bedrooms for Apartment
s in Dublin City Centre is",cor_value))
```

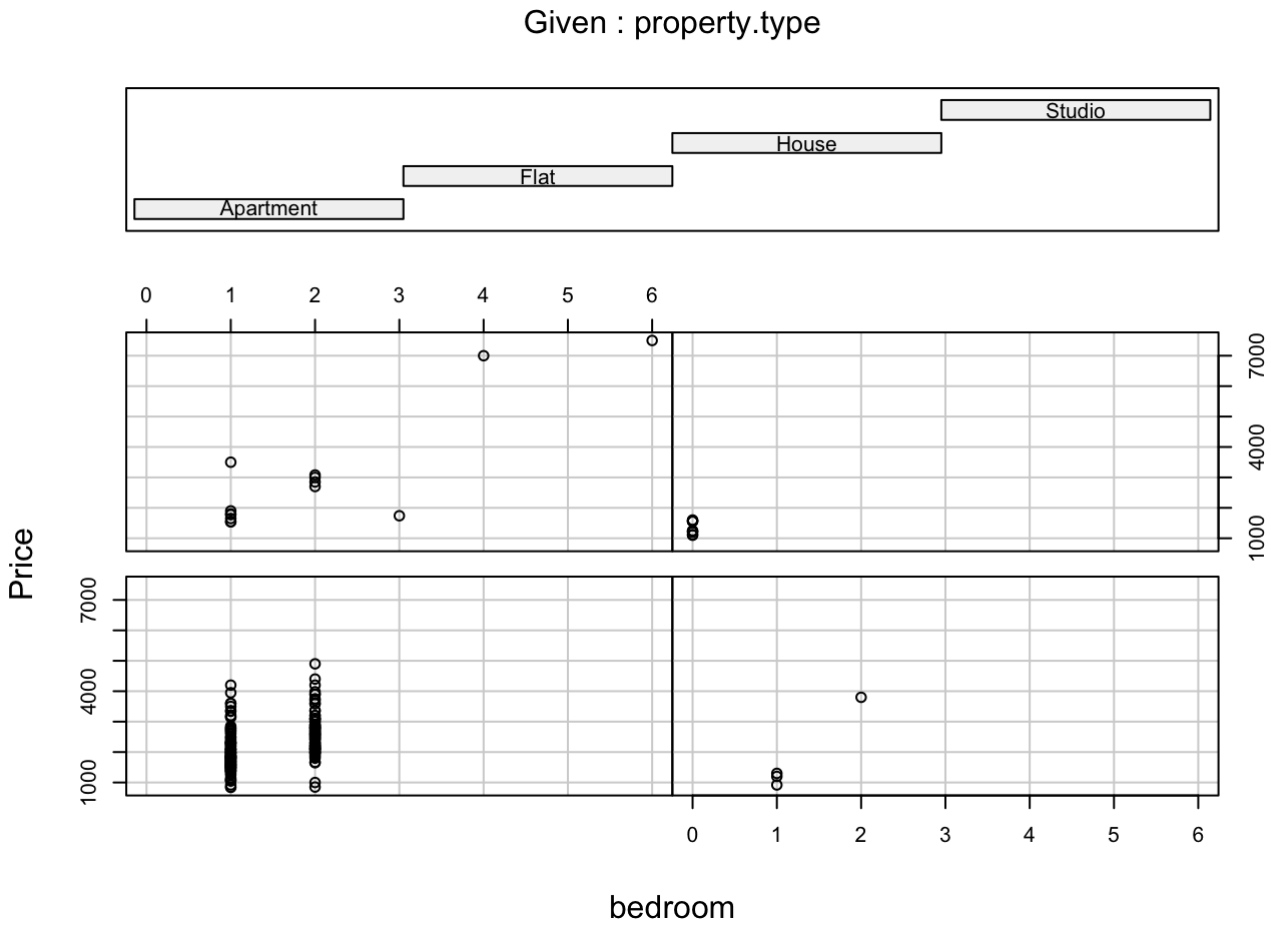
```
## [1] "The correlation value between Price and number of Bedrooms for Apartments in
Dublin City Centre is 0.43053265707973"
```

As the correlation value is approximately near to 0.5, it means there is significant correlation between the Price and bedroom variables.

Q14]

```
df2=df[df$Area=="Dublin City Centre", ]                #data for Dublin City Centre
 Area.
#df2
```

```
coplot(Price~bedroom | property.type , data=df2)
```



Above is the conditioning plot between Price and number of bedrooms for different types of property in Dublin City Centre. Here we see that for Apartments, as bedrooms increases, the price also increases a litte. For rest of the property types, there is not much data available.

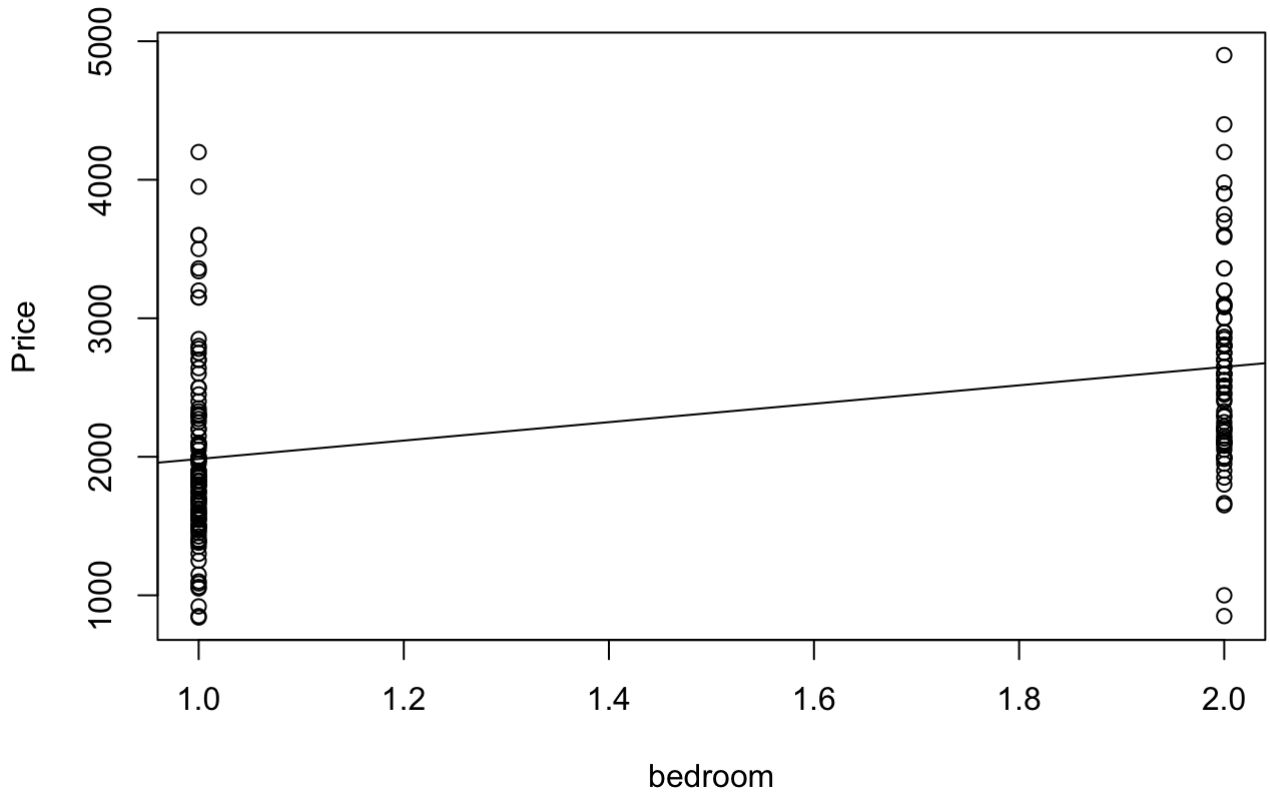
Simple Linear Regression

Q1]

```
linear_model=lm(df1$Price ~ df1$bedroom)
print(linear_model)
```

```
##
## Call:
## lm(formula = df1$Price ~ df1$bedroom)
##
## Coefficients:
## (Intercept)  df1$bedroom
##      1318.8        665.3
```

```
with(df1,plot(bedroom, Price))
abline(linear_model)
```



Q2]

```
estimated_beta0=summary(linear_model)$coefficients[1,1]
estimated_beta1=summary(linear_model)$coefficients[2,1]
print(paste("Price =",estimated_beta0,"+ Bedroom *",estimated_beta1))
```

```
## [1] "Price = 1318.81387457585 + Bedroom * 665.3053914792"
```

Here the Y variable is Price, X variable is Bedroom, beta0 (also called intercept) is 1318.814 and beta1 (also called slope of the line) is 665.305. Intercept means when X value is 0, the line cuts the Y axis at point 1318.814. Slope indicates that if X value increases by 1, the Y value increases by a factor of 665.305.

Q3]

1. The main aim of fitting the model/line is that we want to find a line that minimizes the error values, that is it minimizes the vertical distance between the line and the observed points on the scatter plot.
2. This can be done by taking adding absolute values of errors and then minimizing it. This method is called least-absolute-value (LAV) regression.
3. But in mathematics, absolute values are pretty difficult to work with.
4. Hence we need a different method to make all the errors positive. This can be done by squaring all the error values and if needed, take a square root at the end. This method is called least-squares regression.
5. So we want to find the regression line that minimizes the sum of the areas of these error squares.

6. Using absolute values yields a regression line that is more robust than what we get from least squares method. But it is not efficient.

7. The advantages of least squares method are:

a. It is easy to find the best-fitting regression line
b. It is much easier to work with mathematically
c. It is easy to minimize the error
d. We are sure that we will find only one best fitting line unlike absolute valueswhere we may find more than one best fit lines

7. Note that least squares regression is much sensitive to outliers.

Q4]

There are four assumptions associated with a linear regression model:

1. Linearity: The relationship between X and Y is linear.

2. Homoscedasticity: The variance of residual is the same for any value of X i.e it does not depend on X value. Mathematically, Var(ei) = sigma^2 for all i

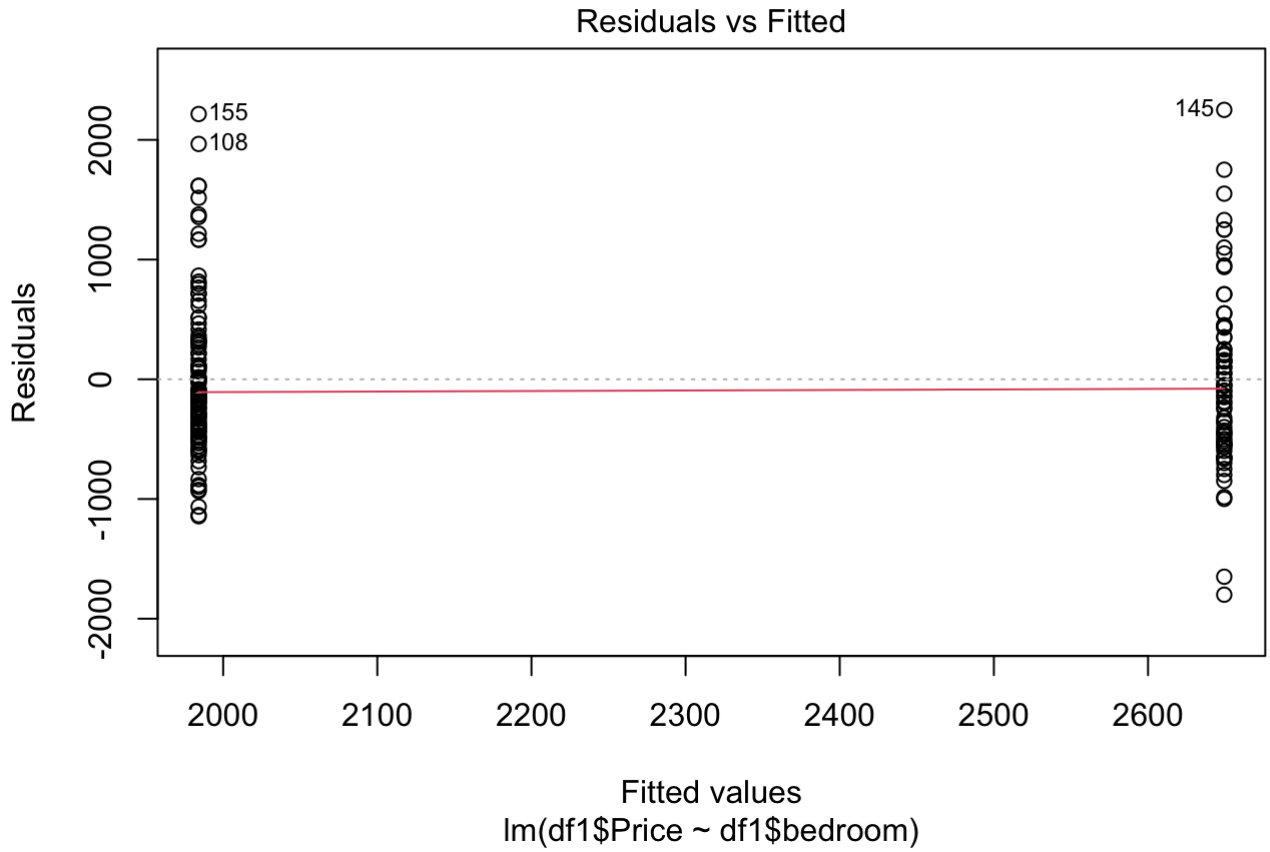
3. Independence: Observations are independent of each other. In other words, Yi or ei values are not correlated with each other i.e Cov(ei,ej)=0, for i!=j

4. Normality of residuals. The residual errors are assumed to be normally distributed.

5. Zero conditional mean :- The error term conditioned on the independent variable should average to 0. In other words, error term is unrelated to independent variables.

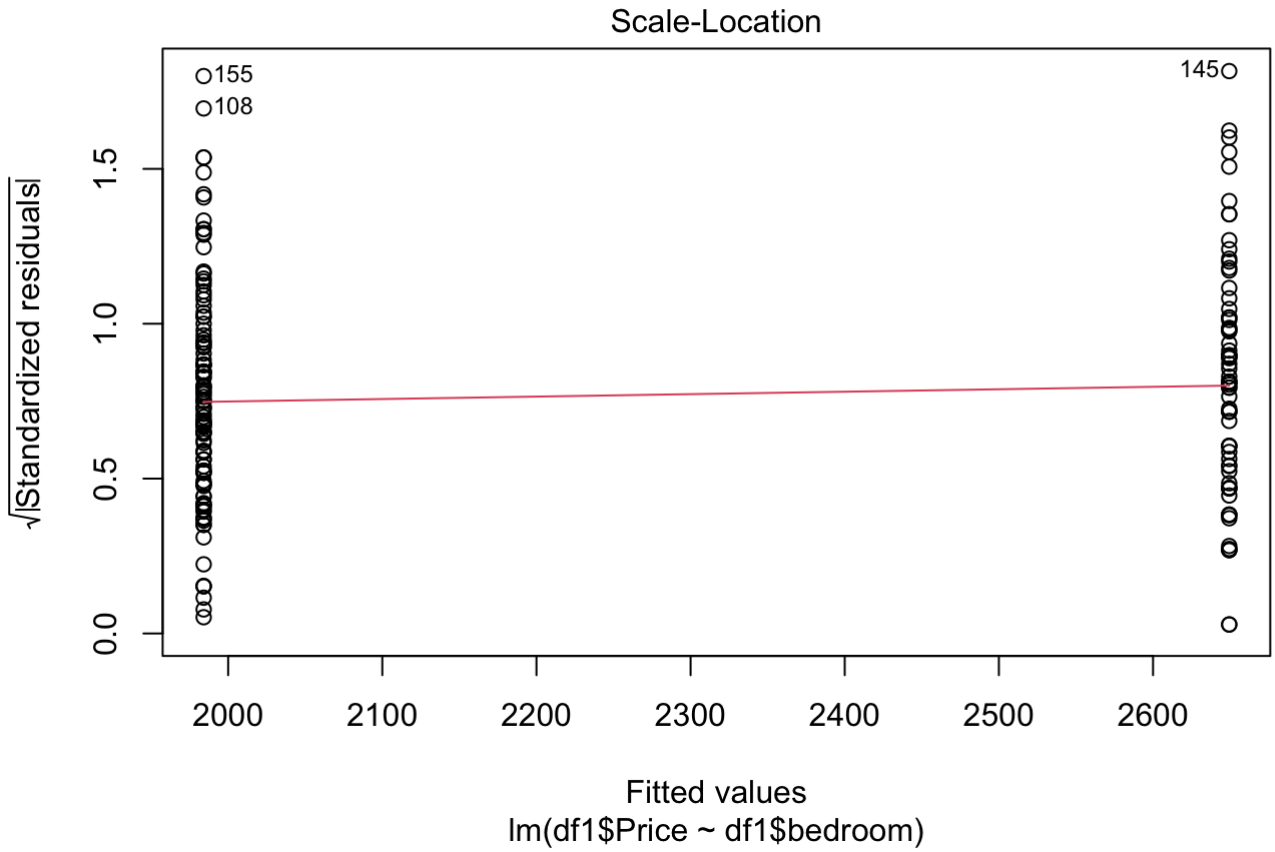
Q5]

```
#Check for linearity of the data
plot(linear_model,1)
```



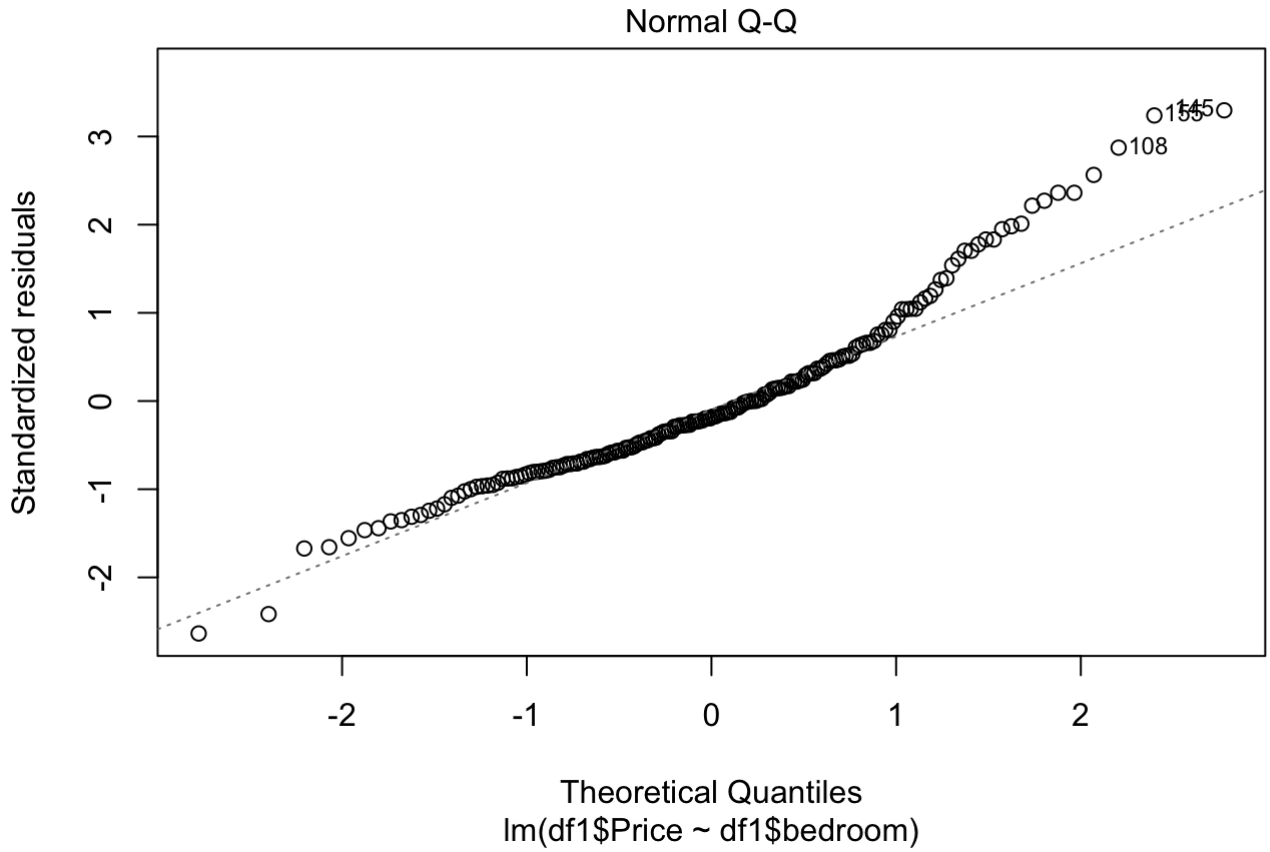
Ideally, the residual plot should show no fitted pattern. That is, the red line should be approximately horizontal at zero, which is satisfied in our case. If there is any presence of a pattern then red line may follow some pattern.

```
#Check for Homogeneity of variance
plot(linear_model,3)
```



Ideally, we should get see a horizontal line with equally spread points. This is also satisfied in our case.

```
#Check for Normality of residuals
plot(linear_model,2)
```



The QQ plot of residuals can be used to visually check the normality assumption. Ideally, the plot should approximately follow a straight line. But in our case, at the higher end, it gets a little deviated from the line.

Q6]

1. If the model is not linear, then our model would be biased and hence make poor predictions on unseen data.
2. If there exists collinearity between variables then as we change one variable, it also changes other variable. Also these 2 variables add up their effect twice, which is not good.
3. If any error term is correlated with one of the independent variables then it causes variable bias.

Q7]

```
X <- df1$bedroom
Y <- df1$Price

SXY = sum(X*Y) - length(X)*mean(X)*mean(Y)
SXY
```

```
## [1] 29087.01
```

```
SXX = sum(X^2) - length(X)*mean(X)^2
SXX
```

```
## [1] 43.71978
```

```
beta1_hat = SXY/SXX
print(paste("Beta 1 :-",beta1_hat))
```

```
## [1] "Beta 1 :- 665.305391479201"
```

```
beta0_hat = mean(Y) - beta1_hat*mean(X)
print(paste("Beta 0 :-",beta0_hat))
```

```
## [1] "Beta 0 :- 1318.81387457585"
```

Hence if X is 0, y is 1318.8138.

Q8]

```
beta1_hat = SXY/SXX
print(paste("Beta 1 :-",beta1_hat))
```

```
## [1] "Beta 1 :- 665.305391479201"
```

Hence change in X by 1 unit causes change in Y by 665.305.

Q9]

beta1_hat = SXY/SXX SXY is sample covariance between X and Y, SXX is sample variance of X

SXY = RXY * SX * SY RXY is sample correlation coefficient between x and y, SX is sample standard deviation of X, SY is sample standard deviation of Y

Hence sample correlation coefficient between x and y is directly proportional to estimate of slope(beta1).

Q10]

```
Yhat = beta0_hat + beta1_hat*X
SSE = sum((Y-Yhat)^2)
SSE
```

```
## [1] 85050027
```

```
MSE = SSE/(length(Y)-2)
MSE
```

```
## [1] 472500.2
```

```
Var_slope = MSE/SXX
print(paste("Variance in estimated slope is",Var_slope))
```

```
## [1] "Variance in estimated slope is 10807.468592224"
```

```
Var_intercept = MSE*(1/length(Y) + mean(X)^2/SXX)
print(paste("Variance in estimated intercept is",Var_intercept))
```

```
## [1] "Variance in estimated intercept is 23812.0599202297"
```

Q11]

```
SE_slope = sqrt(Var_slope)
print(paste("Standard error in estimated slope is",SE_slope))
```

```
## [1] "Standard error in estimated slope is 103.958975525079"
```

```
SE_intercept = sqrt(Var_intercept)
print(paste("Standard error in estimated intercept is",SE_intercept))
```

```
## [1] "Standard error in estimated intercept is 154.3115676812"
```

Hence there is quite large standard error in estimated slope and variance.

Q12]

```
alpha=0.05
c(beta0_hat - qt(1-alpha/2,length(Y)-2)*sqrt(Var_intercept),
  beta0_hat + qt(1-alpha/2,length(Y)-2)*sqrt(Var_intercept))
```

```
## [1] 1014.322 1623.306
```

Hence beta0 lies between 1014 and 1623.

Q13]

```
alpha=0.05
c(beta1_hat - qt(1-alpha/2,length(Y)-2)*sqrt(Var_slope),
  beta1_hat + qt(1-alpha/2,length(Y)-2)*sqrt(Var_slope))
```

```
## [1] 460.1703 870.4404
```

Hence beta1 lies between 460 and 870.

Q14]

```
T = (beta0_hat-0)/SE_intercept
T
```

```
## [1] 8.546436
```

```
alpha =0.05
qt(1-alpha/2,  length(Y)-2)
```

```
## [1] 1.973231
```

```
pval <- 2*(1-pt(abs(T), length(Y)-2))
pval
```

```
## [1] 5.329071e-15
```

As abs(T)>qt , hence reject H0

At the 5% level of significance, the probability that beta0 = 0 is pval

Q15]

```
T = (beta1_hat-0)/SE_slope
T
```

```
## [1] 6.399692
```

```
alpha =0.05
qt(1-alpha/2, 3)
```

```
## [1] 3.182446
```

```
pval <- 2*(1-pt(abs(T), length(Y)-2))
pval
```

```
## [1] 1.308469e-09
```

As abs(T)>qt, hence reject H0

At the 5% level of significance, the probability that beta1 = 0 is pval

Q16]

```
summary(linear_model)
```

```
##
## Call:
## lm(formula = df1$Price ~ df1$bedroom)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1799.4  -449.4  -127.6   315.9  2250.6
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1318.8      154.3   8.546 5.31e-15 ***
## df1$bedroom    665.3      104.0   6.400 1.31e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 687.4 on 180 degrees of freedom
## Multiple R-squared:  0.1854, Adjusted R-squared:  0.1808
## F-statistic: 40.96 on 1 and 180 DF,  p-value: 1.308e-09
```

```
SSR = sum((Yhat - mean(Y))^2)
SSE = sum((Yhat - Y)^2)
MSR = SSR/1
MSE = SSE/(length(Y)-2)
F = MSR/MSE
F                                    #Same as that in the table
```

```
## [1] 40.95605
```

```
alpha =0.05
qf(1-alpha,1,length(Y)-2)
```

```
## [1] 3.89364
```

```
pf(F,1,length(Y)-2,lower.tail=FALSE)
```

```
## [1] 1.308469e-09
```

As F > rf , hence reject H0

The probability of the F-distribution taking the value 40.95605 is 1.308469e-09. As 1.308469e-09<0.05, reject H0

Q17]

```
summary(linear_model)
```

```
##
## Call:
## lm(formula = df1$Price ~ df1$bedroom)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1799.4  -449.4  -127.6   315.9  2250.6
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1318.8      154.3   8.546 5.31e-15 ***
## df1$bedroom    665.3      104.0   6.400 1.31e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 687.4 on 180 degrees of freedom
## Multiple R-squared:  0.1854, Adjusted R-squared:  0.1808
## F-statistic: 40.96 on 1 and 180 DF,  p-value: 1.308e-09
```

```
SSE = sum((Yhat - Y)^2)
SST = sum((Y - mean(Y))^2)
R2 = (SST -SSE)/SST
R2
```

```
## [1] 0.1853584
```

```
#or
R2 = (SSR)/SST
R2
```

```
## [1] 0.1853584
```

This R2 is same as adjusted R square in the summary() R2=0 means no relationship R2=1 implies perfect linear fit Higher R2 indicates better model

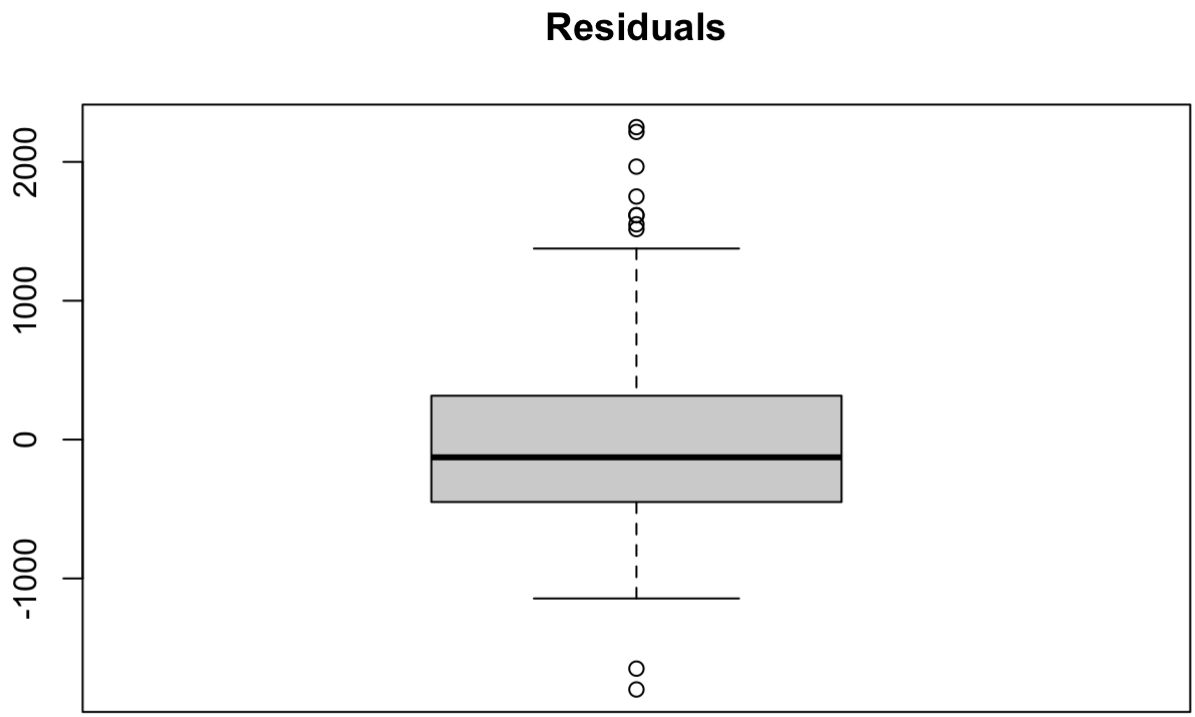
But in our case, it is very low. Hence our model is not that good.

Q18]

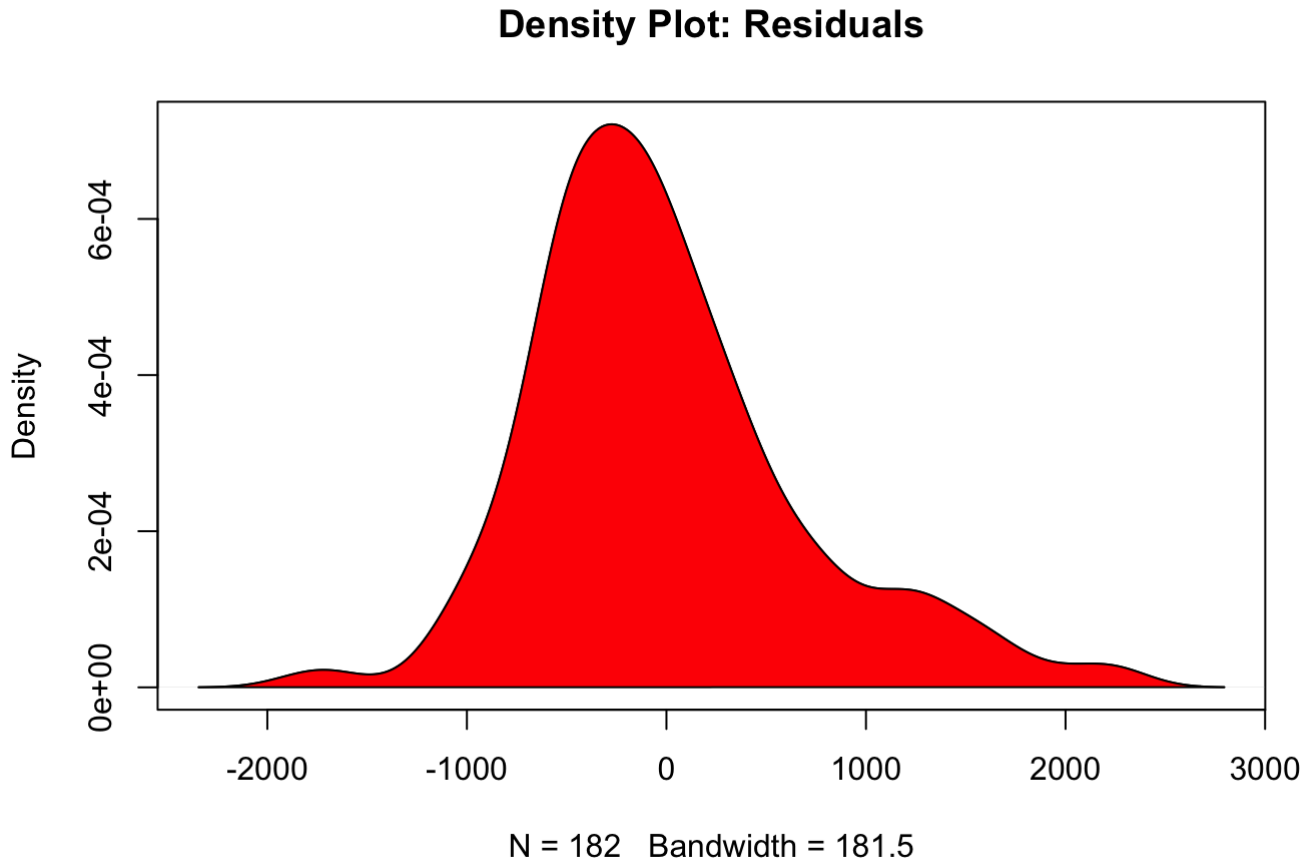
```
summary(residuals(linear_model))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1799.4  -449.4  -127.6     0.0   315.9  2250.6
```

```
boxplot(residuals(linear_model), main="Residuals")
```




```
plot(density(residuals(linear_model)),main="Density Plot: Residuals")
polygon(density(residuals(linear_model)), col="red")
```



```
RMSE = sqrt(SSE/(length(Y)-2))
RMSE
```

```
## [1] 687.3865
```

```
summary(linear_model)
```

```
##
## Call:
## lm(formula = df1$Price ~ df1$bedroom)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1799.4  -449.4  -127.6   315.9  2250.6
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1318.8      154.3   8.546 5.31e-15 ***
## df1$bedroom    665.3      104.0   6.400 1.31e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 687.4 on 180 degrees of freedom
## Multiple R-squared:  0.1854, Adjusted R-squared:  0.1808
## F-statistic: 40.96 on 1 and 180 DF,  p-value: 1.308e-09
```

Hence Bedroom predicts Price with about 687 units average error in price.

Q19]

```
SXX = sum((X - mean(X))^2)
VAR_Y = MSE*(1/length(Y) + (X-mean(X))^2/SXX)
cbind(Yhat- qt(1-alpha/2,length(Y)-2)*sqrt( VAR_Y),
      Yhat + qt(1-alpha/2,length(Y)-2)*sqrt( VAR_Y))
```

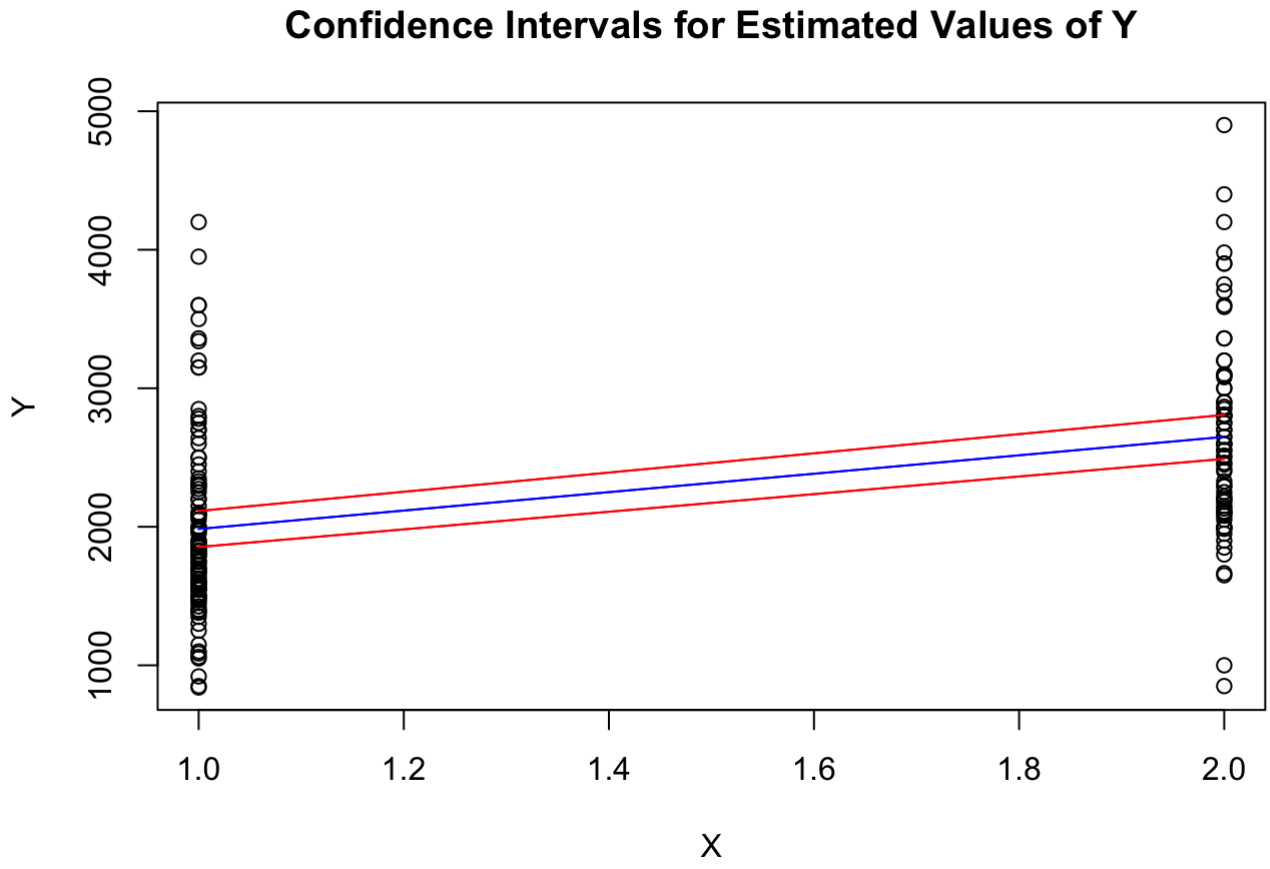
```
##            [,1]     [,2]
##   [1,] 1854.202 2114.036
##   [2,] 2490.673 2808.176
##   [3,] 1854.202 2114.036
##   [4,] 1854.202 2114.036
##   [5,] 1854.202 2114.036
##   [6,] 1854.202 2114.036
##   [7,] 2490.673 2808.176
##   [8,] 2490.673 2808.176
##   [9,] 2490.673 2808.176
##  [10,] 2490.673 2808.176
##  [11,] 2490.673 2808.176
##  [12,] 2490.673 2808.176
##  [13,] 2490.673 2808.176
##  [14,] 2490.673 2808.176
##  [15,] 1854.202 2114.036
##  [16,] 1854.202 2114.036
##  [17,] 2490.673 2808.176
##  [18,] 2490.673 2808.176
##  [19,] 2490.673 2808.176
##  [20,] 1854.202 2114.036
##  [21,] 1854.202 2114.036
##  [22,] 1854.202 2114.036
##  [23,] 2490.673 2808.176
##  [24,] 1854.202 2114.036
##  [25,] 1854.202 2114.036
##  [26,] 1854.202 2114.036
##  [27,] 1854.202 2114.036
##  [28,] 1854.202 2114.036
##  [29,] 2490.673 2808.176
##  [30,] 1854.202 2114.036
##  [31,] 2490.673 2808.176
##  [32,] 1854.202 2114.036
##  [33,] 1854.202 2114.036
##  [34,] 1854.202 2114.036
##  [35,] 1854.202 2114.036
##  [36,] 1854.202 2114.036
##  [37,] 1854.202 2114.036
##  [38,] 1854.202 2114.036
##  [39,] 2490.673 2808.176
##  [40,] 1854.202 2114.036
##  [41,] 1854.202 2114.036
##  [42,] 1854.202 2114.036
##  [43,] 1854.202 2114.036
##  [44,] 2490.673 2808.176
##  [45,] 1854.202 2114.036
##  [46,] 2490.673 2808.176
##  [47,] 1854.202 2114.036
##  [48,] 1854.202 2114.036
##  [49,] 1854.202 2114.036
##  [50,] 1854.202 2114.036
##  [51,] 1854.202 2114.036
##  [52,] 1854.202 2114.036
##  [53,] 1854.202 2114.036
##  [54,] 2490.673 2808.176
##  [55,] 1854.202 2114.036
##  [56,] 2490.673 2808.176
```

```
##  [57,] 1854.202 2114.036
##  [58,] 2490.673 2808.176
##  [59,] 1854.202 2114.036
##  [60,] 1854.202 2114.036
##  [61,] 2490.673 2808.176
##  [62,] 1854.202 2114.036
##  [63,] 1854.202 2114.036
##  [64,] 1854.202 2114.036
##  [65,] 1854.202 2114.036
##  [66,] 1854.202 2114.036
##  [67,] 2490.673 2808.176
##  [68,] 1854.202 2114.036
##  [69,] 1854.202 2114.036
##  [70,] 2490.673 2808.176
##  [71,] 2490.673 2808.176
##  [72,] 2490.673 2808.176
##  [73,] 1854.202 2114.036
##  [74,] 2490.673 2808.176
##  [75,] 1854.202 2114.036
##  [76,] 1854.202 2114.036
##  [77,] 1854.202 2114.036
##  [78,] 1854.202 2114.036
##  [79,] 1854.202 2114.036
##  [80,] 2490.673 2808.176
##  [81,] 1854.202 2114.036
##  [82,] 2490.673 2808.176
##  [83,] 2490.673 2808.176
##  [84,] 2490.673 2808.176
##  [85,] 2490.673 2808.176
##  [86,] 1854.202 2114.036
##  [87,] 2490.673 2808.176
##  [88,] 1854.202 2114.036
##  [89,] 1854.202 2114.036
##  [90,] 1854.202 2114.036
##  [91,] 2490.673 2808.176
##  [92,] 2490.673 2808.176
##  [93,] 1854.202 2114.036
##  [94,] 1854.202 2114.036
##  [95,] 2490.673 2808.176
##  [96,] 1854.202 2114.036
##  [97,] 1854.202 2114.036
##  [98,] 1854.202 2114.036
##  [99,] 2490.673 2808.176
## [100,] 2490.673 2808.176
## [101,] 2490.673 2808.176
## [102,] 2490.673 2808.176
## [103,] 2490.673 2808.176
## [104,] 1854.202 2114.036
## [105,] 1854.202 2114.036
## [106,] 2490.673 2808.176
## [107,] 1854.202 2114.036
## [108,] 1854.202 2114.036
## [109,] 2490.673 2808.176
## [110,] 2490.673 2808.176
## [111,] 1854.202 2114.036
## [112,] 2490.673 2808.176
## [113,] 1854.202 2114.036
## [114,] 2490.673 2808.176
```

```
## [115,] 1854.202 2114.036
## [116,] 2490.673 2808.176
## [117,] 1854.202 2114.036
## [118,] 1854.202 2114.036
## [119,] 1854.202 2114.036
## [120,] 2490.673 2808.176
## [121,] 1854.202 2114.036
## [122,] 1854.202 2114.036
## [123,] 1854.202 2114.036
## [124,] 1854.202 2114.036
## [125,] 1854.202 2114.036
## [126,] 2490.673 2808.176
## [127,] 2490.673 2808.176
## [128,] 2490.673 2808.176
## [129,] 2490.673 2808.176
## [130,] 2490.673 2808.176
## [131,] 2490.673 2808.176
## [132,] 1854.202 2114.036
## [133,] 1854.202 2114.036
## [134,] 1854.202 2114.036
## [135,] 2490.673 2808.176
## [136,] 1854.202 2114.036
## [137,] 1854.202 2114.036
## [138,] 2490.673 2808.176
## [139,] 2490.673 2808.176
## [140,] 2490.673 2808.176
## [141,] 1854.202 2114.036
## [142,] 1854.202 2114.036
## [143,] 2490.673 2808.176
## [144,] 2490.673 2808.176
## [145,] 2490.673 2808.176
## [146,] 1854.202 2114.036
## [147,] 1854.202 2114.036
## [148,] 1854.202 2114.036
## [149,] 2490.673 2808.176
## [150,] 1854.202 2114.036
## [151,] 1854.202 2114.036
## [152,] 1854.202 2114.036
## [153,] 1854.202 2114.036
## [154,] 1854.202 2114.036
## [155,] 1854.202 2114.036
## [156,] 2490.673 2808.176
## [157,] 1854.202 2114.036
## [158,] 1854.202 2114.036
## [159,] 1854.202 2114.036
## [160,] 1854.202 2114.036
## [161,] 2490.673 2808.176
## [162,] 1854.202 2114.036
## [163,] 1854.202 2114.036
## [164,] 2490.673 2808.176
## [165,] 2490.673 2808.176
## [166,] 2490.673 2808.176
## [167,] 2490.673 2808.176
## [168,] 1854.202 2114.036
## [169,] 1854.202 2114.036
## [170,] 2490.673 2808.176
## [171,] 1854.202 2114.036
## [172,] 2490.673 2808.176
```

```
## [173,] 2490.673 2808.176
## [174,] 1854.202 2114.036
## [175,] 1854.202 2114.036
## [176,] 1854.202 2114.036
## [177,] 1854.202 2114.036
## [178,] 1854.202 2114.036
## [179,] 1854.202 2114.036
## [180,] 2490.673 2808.176
## [181,] 1854.202 2114.036
## [182,] 2490.673 2808.176
```

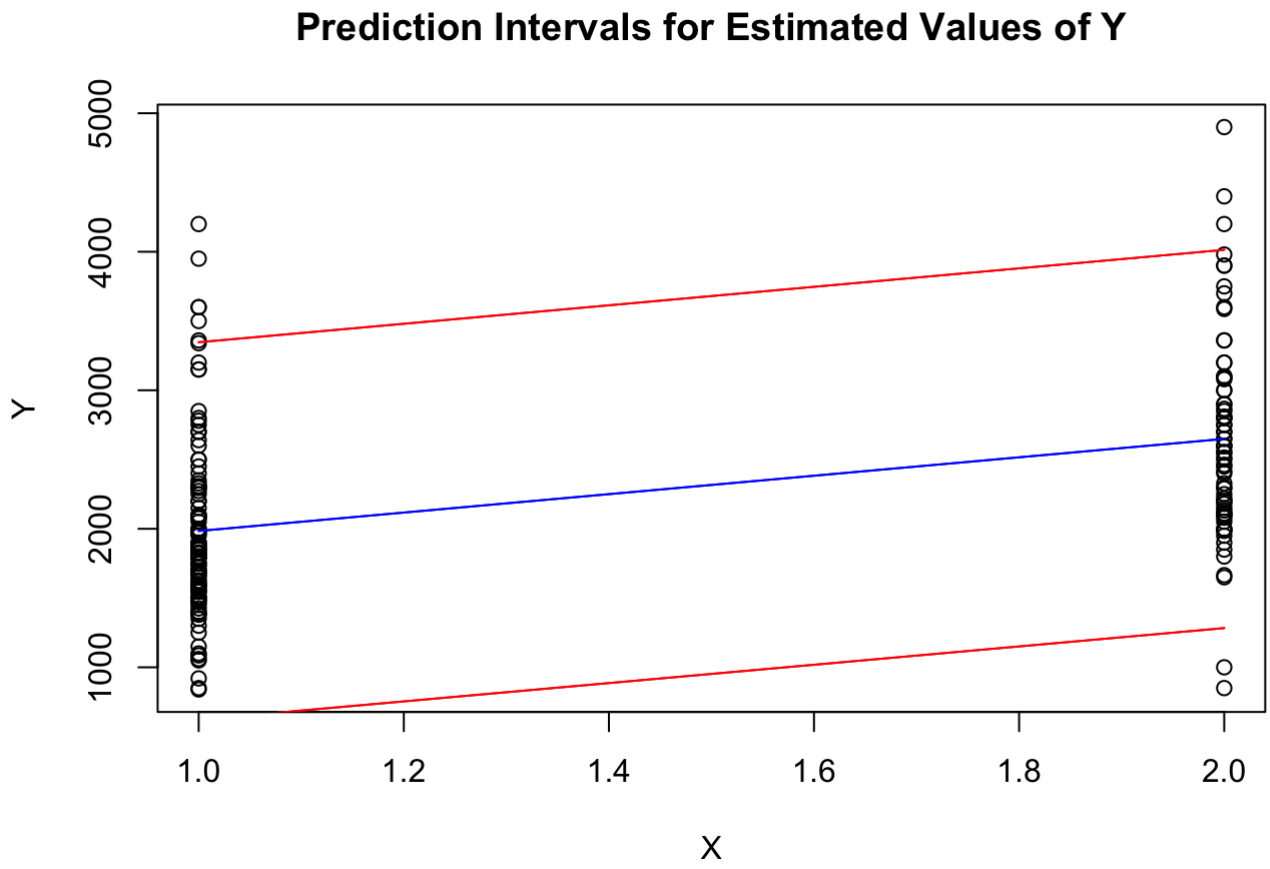
```
plot(X,Y,main = "Confidence Intervals for Estimated Values of Y")
lines(X,Yhat,col="blue")
VAR_Y = MSE*(1/length(X)+(X-mean(X))^2/SXX)
lines(X,Yhat+qt(1-alpha/2,length(X)-2)*sqrt(VAR_Y),col="red")
lines(X,Yhat-qt(1-alpha/2,length(X)-2)*sqrt(VAR_Y),col="red")
```



The confidence intervals are uniformly distanced over entire range of x. Also it is quite narrow.

Q20]

```
plot(X,Y,main = "Prediction Intervals for Estimated Values of Y")
lines(X,Yhat,col="blue")
Var_E = MSE*(1 + 1/length(X) + (X-mean(X))^2/SXX)
lines(X,Yhat+qt(1-alpha/2,length(X)-2)*sqrt(Var_E),col="red")
lines(X,Yhat-qt(1-alpha/2,length(X)-2)*sqrt(Var_E),col="red")
```



The prediction intervals are uniformly distanced over entire range of x. Also it is quite wide.