# STATISTICS FOR DATA ANALYTICS

## CA One

Dublin Business School
Module Code: B9DA101
Module Leader: Dr. Muhammad Alli
Dublin Business School
To be completed by date: Thursday, 31st October 2024, 11:55 PM

Sandeep Kumar
Student ID: 20049275 | Email: 20049275@mydbs.ie
Git:https://github.com/sandeepkumar-84/DBS.git

## *Necessary packages are installed, and libraries are imported at the beginning.*

***tidyverse*** – This core package makes available the packages like ggplot2, dplyr, etc which are extensively used in any project.

***ggplot2*** – It is used for data visualization like mapping, plotting, etc.

***dplyr*** – It provides a grammar of data manipulation, providing a consistent set of verbs that solve the most common data manipulation challenges.

***Fitdistrplus***- Help to Fit of a Parametric Distribution – In this project it will be used to for gamma distribution plotting.

***Ggcorrplot*** – It can be used to visualize easily a correlation matrix using ggplot2

***reshape2*** - aggregate data using just two functions: melt and 'dcast'

```
#Installing all the necessary packages and importing the libraries.

install.packages('tidyverse')
install.packages("fitdistrplus")
install.packages("ggcorrplot")
install.packages("reshape2")
library('tidyverse')
library(ggplot2)
library(dplyr)
library(fitdistrplus)
library(reshape2)
library(ggcorrplot)
```

## *Question 1: Describe the dataset using appropriate plots/curves/charts, …*

### *About Dataset*

***Dataset Description****:* The Vehicle dataset is a collection of columns providing the sales transactional data. Dataset is appropriately chose so that its variables can be utilized in the analysis fulfilling the requirement of the assignment, like the presence of continues and categorical columns are checked so that the descriptive analytics can be done, probability models can be applied, and hypothesis testing can be done effectively to predict the sale prices and the affect of different variables on sales prices. The dataset has vast variety of different vehicles with different year, make and models registered with different states.

***Dataset Features****:* The dataset has vast variety of different vehicles with different year, make and models registered with different states. MMR column provides the market estimation of the vehicles. Odometer reading can also be helpful in analysing its influence on the selling price. This dataset can be utilized for research study to develop predictive models for estimating vehicle price and trends in automotive industry. Dataset is downloaded from the Kaggle website.

## *Loading and previewing the dataset*

The dataset is first imported into the R environment using appropriate functions such as read.csv() for CSV files

```
#Load the dataset

dsVehAll <- read.csv("C:/AssignmentFiles/VehicleDataV3.csv")
```

The head() function is utilized to print the first six rows of the dataset, giving a clear view of the data in its raw form.

```
> dsVeh <- VehicleDataV3[0:20000,]
> head(dsVeh)
  year   make              model       trim  body transmission              vin state condition odometer color interior
1 2015    Kia             Sorento         LX   SUV    automatic 5xyktca69fg566472    ca         5    16639 white    black
2 2015    Kia             Sorento         LX   SUV    automatic 5xyktca69fg561319    ca         5     9393 white    beige
3 2014    BMW            3 Series 328i SULEV Sedan    automatic wba3c1c51ek116351    ca        45     1331  gray    black
4 2015  Volvo                 S60         T5 Sedan    automatic yv1612tb4f1310987    ca        41    14282 white    black
5 2014    BMW 6 Series Gran Coupe       650i Sedan    automatic wba6b2c57ed129731    ca        43     2641  gray    black
6 2015 Nissan              Altima      2.5 S Sedan    automatic 1n4al3ap1fn326013    ca         1     5554  gray    black
                                  seller   mmr sellingprice                           saledate
1                          kia motors america  inc 20500        21500 Tue Dec 16 2014 12:30:00 GMT-0800 (PST)
2                          kia motors america  inc 20800        21500 Tue Dec 16 2014 12:30:00 GMT-0800 (PST)
3            financial services remarketing (lease) 31900        30000 Thu Jan 15 2015 04:30:00 GMT-0800 (PST)
4                          volvo na rep/world omni 27500        27750 Thu Jan 29 2015 04:30:00 GMT-0800 (PST)
5            financial services remarketing (lease) 66000        67000 Thu Dec 18 2014 12:30:00 GMT-0800 (PST)
6 enterprise vehicle exchange / tra / rental / tulsa 15350        10900 Tue Dec 30 2014 12:00:00 GMT-0800 (PST)
>
```

The summary() function is used to generate descriptive statistics for each variable in the dataset. Total number of variables\columns in the dataset are 16.

```
#Taking the vehicle data set (downloaded from kaggle).
dsVeh <- VehicleDataV3[0:20000,]
head(dsVeh)

#Summarizing each column of the dataset
summary(dsVeh)
```

```
21:15  (Top Level)                                                    R Script
R 4.4.1
>
> #Summarizing each column of the dataset
> summary(dsVeh)
      year          make              model              trim              body          transmission
 Min.   :1986   Length:20000       Length:20000       Length:20000       Length:20000       Length:20000
 1st Qu.:2007   Class :character   Class :character   Class :character   Class :character   Class :character
 Median :2010   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character
 Mean   :2009
 3rd Qu.:2012
 Max.   :2015

     vin               state              condition        odometer         color              interior
 Length:20000       Length:20000       Min.   : 1.00   Min.   :     1   Length:20000       Length:20000
 Class :character   Class :character   1st Qu.:23.00   1st Qu.: 36065   Class :character   Class :character
 Mode  :character   Mode  :character   Median :33.00   Median : 71224   Mode  :character   Mode  :character
                                       Mean   :29.54   Mean   : 77202
                                       3rd Qu.:39.00   3rd Qu.:108548
                                       Max.   :49.00   Max.   :999999
                                       NA's   :4379
    seller               mmr          sellingprice      saledate
 Length:20000       Min.   :    25   Min.   :     1   Length:20000
 Class :character   1st Qu.:  6450   1st Qu.:  6000   Class :character
 Mode  :character   Median : 11200   Median : 10800   Mode  :character
                    Mean   : 13106   Mean   : 12713
                    3rd Qu.: 17400   3rd Qu.: 17100
                    Max.   :157000   Max.   :154000
```

str() function is employed to examine the structure of the dataset.

```
26
27 str(dsVeh)
28
29
27:1  (Top Level)

R 4.4.1
> str(dsVeh)
'data.frame':   20000 obs. of  16 variables:
 $ year        : int  2015 2015 2014 2015 2014 2015 2014 2014 2014 2014 ...
 $ make        : chr  "Kia" "Kia" "BMW" "Volvo" ...
 $ model       : chr  "Sorento" "Sorento" "3 Series" "S60" ...
 $ trim        : chr  "LX" "LX" "328i SULEV" "T5" ...
 $ body        : chr  "SUV" "SUV" "Sedan" "Sedan" ...
 $ transmission: chr  "automatic" "automatic" "automatic" "automatic" ...
 $ vin         : chr  "5xyktca69fg566472" "5xyktca69fg561319" "wba3c1c51ek116351" "yv1612tb4f1310987" ...
 $ state       : chr  "ca" "ca" "ca" "ca" ...
 $ condition   : int  5 5 45 41 43 1 34 2 42 3 ...
 $ odometer    : int  16639 9393 1331 14282 2641 5554 14943 28617 9557 4809 ...
 $ color       : chr  "white" "white" "gray" "white" ...
 $ interior    : chr  "black" "beige" "black" "black" ...
 $ seller      : chr  "kia motors america  inc" "kia motors america  inc" "financial services remarketing (lease)" "volvo na rep/world omni" ...
 $ mmr         : int  20500 20800 31900 27500 66000 15350 69000 11900 32100 26300 ...
 $ sellingprice: int  21500 21500 30000 27750 67000 10900 65000 9800 32250 17500 ...
 $ saledate    : chr  "Tue Dec 16 2014 12:30:00 GMT-0800 (PST)" "Tue Dec 16 2014 12:30:00 GMT-0800 (PST)" "Thu Jan 15 2015 04:30:00 GMT-0800 (PST)" "Thu Jan
29 2015 04:30:00 GMT-0800 (PST)" ...
```

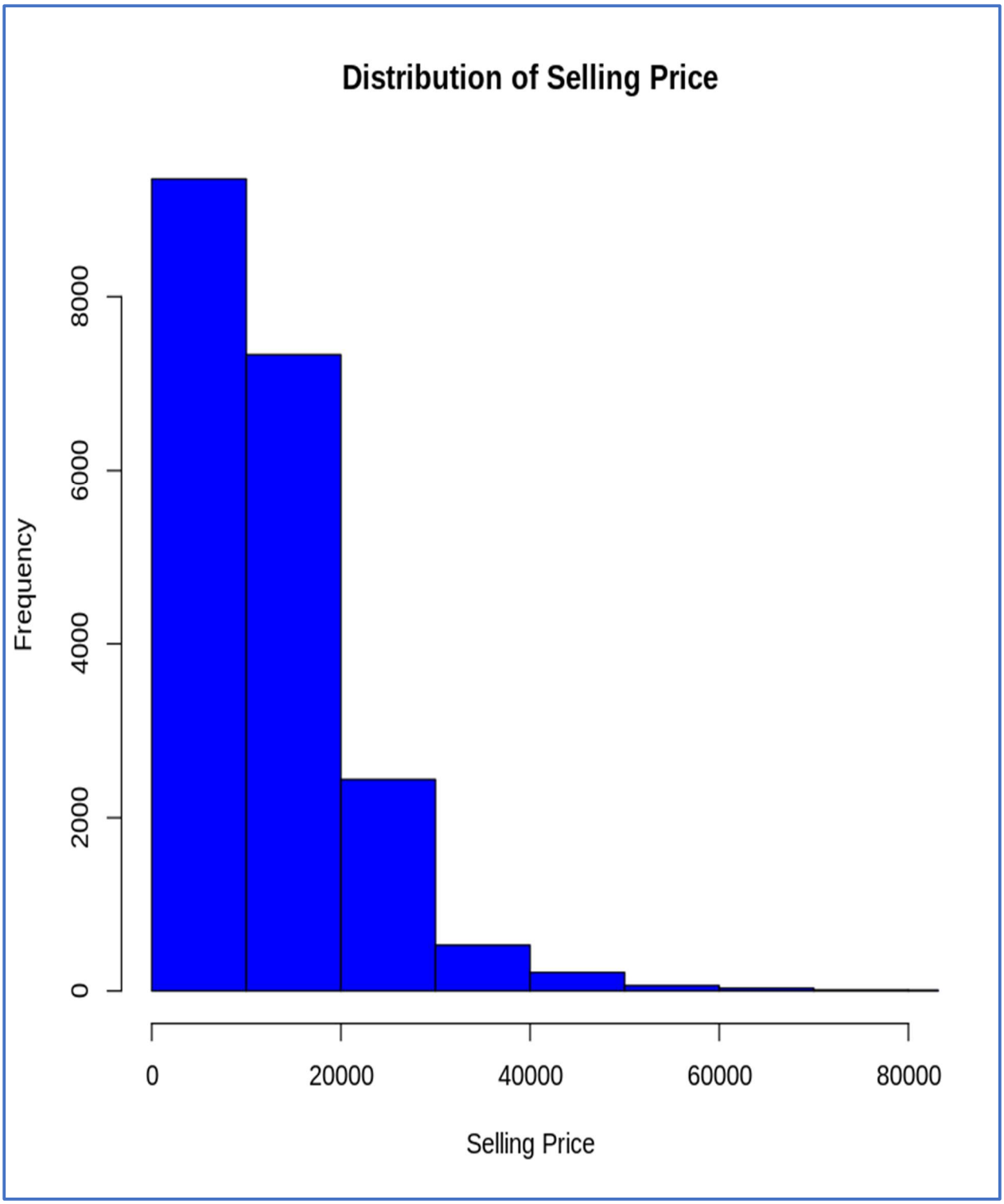
## *Question 1 a) Describe the dataset using appropriate plots/curves/charts*

### *1.a) 1. Histogram of the sellingprice column of the dataset.*

Histogram is plotted using hist(x,xlab,ylab,xlim,main,col) function in R which are graphical representations of the distribution of a dataset. The parameters provided in the hist() function includes the sellingprice variable in the x vector, xlab and ylab are x and y labels respectively, col for color of the histogram.

- It is evident from the histogram of the selling price that it is a right-skewed (or positively skewed) distribution.
- Frequency distribution of the vehicles with lower range price is much higher than the vehicles with higher price units that is frequencies decrease as values increase to the right gradually.
- The number of bins are around 6, which provides clear visualization of the skewness of the data.

```
35 #Question 1 - (a) Describe the dataset using appropriate plots/curves/charts,
36
37   #Histogram of the sellingprice column of the dataset.
38 hist(dsVeh$sellingprice, xlab = "Selling Price", ylab = "Frequency", xlim = c(0,80000),
39      main = "Distribution of Selling Price", col = "blue")
```


Distribution of Selling Price
Frequency
0
2000
4000
6000
8000
0
20000
40000
60000
80000
Selling Price

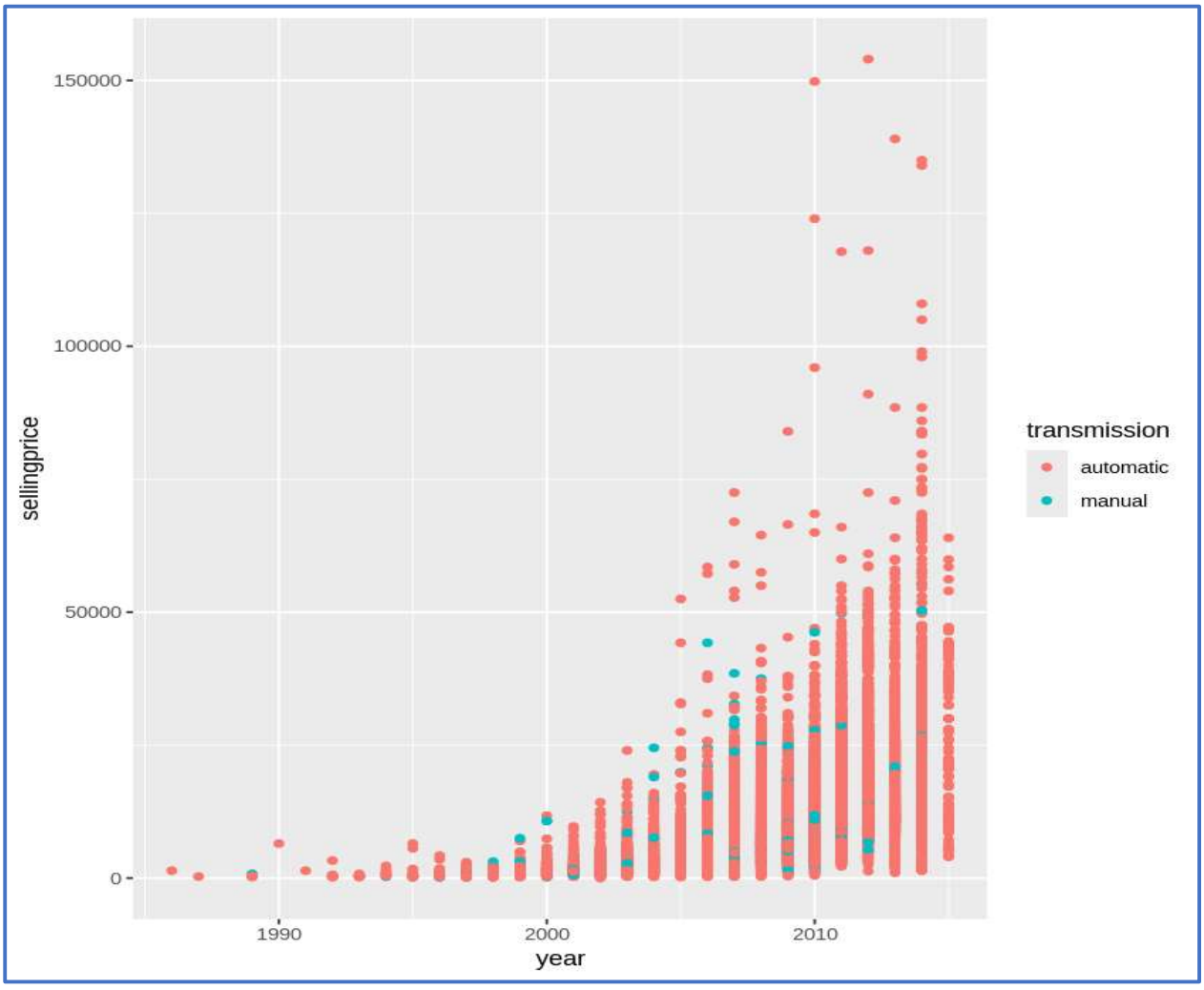
### *1 a) 2. Scatter Plotting of the selling price based on year.*

Scatter Plotting is done using the ggplot function. The parameters provided are the vehicle dataset in its data parameter, year in the x axis, selling price in the y axis, while color is dependent on the transmission variable, that means different transmissions can be recognized by distinct colors.

It highlights the relationship between car age, selling price, and transmission type

- It is evident from the scatter plot that there is an upward trend in selling prices as the vehicle manufacturing year increases, especially after the year 2000.
- Older cars have lower prices.
- Dataset is dominated by automatic cars while manual cars are less frequent

```
29  #Plotting of the selling price based on year. Different transmission can be recongnized by distinct colors.
30  ggplot(data = dsVeh, aes(x = year , y = sellingprice)) +
31    geom_point(mapping = aes(x=year, y=sellingprice, color=transmission))
```

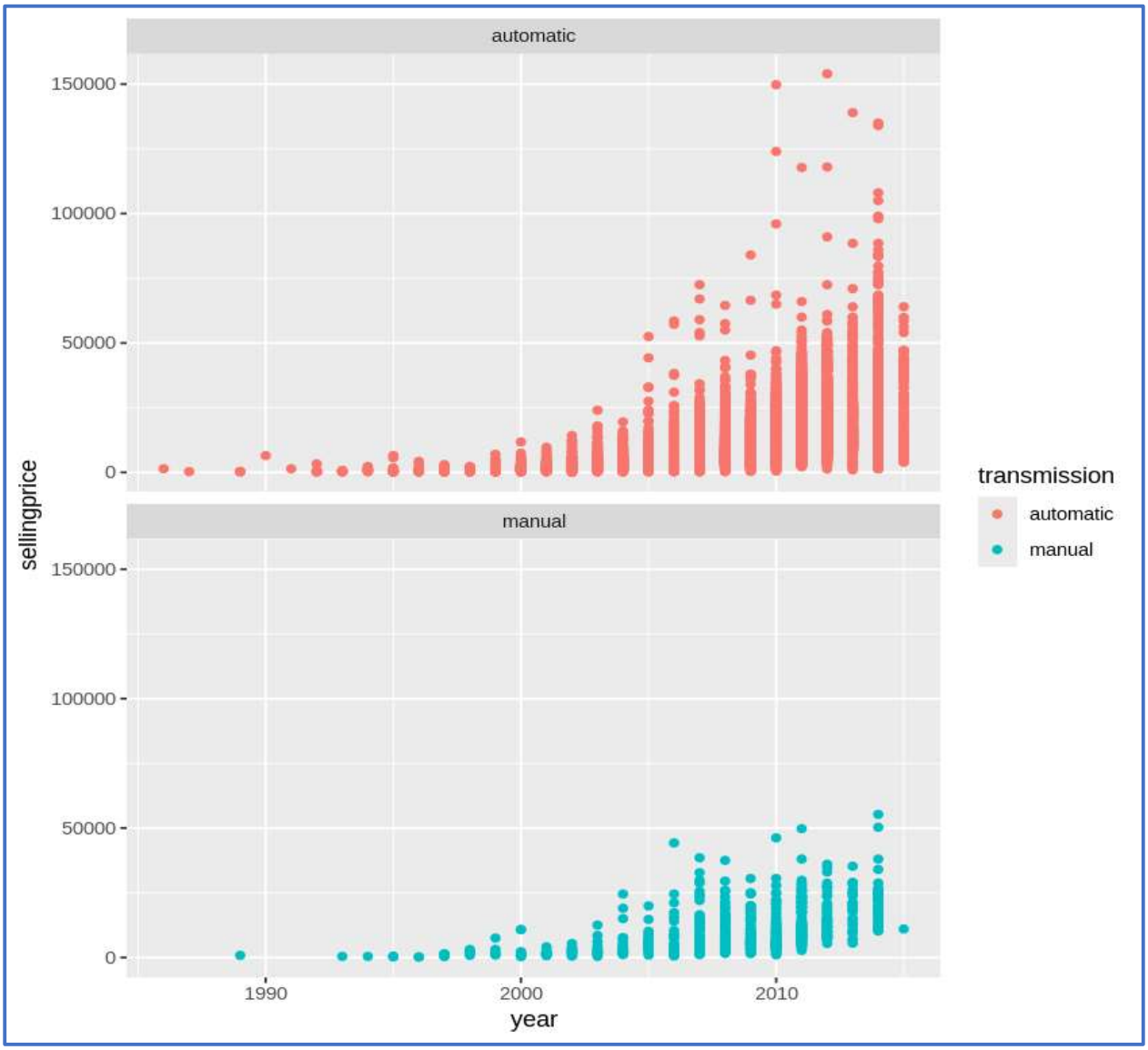


### *1 a) 3. Separate plots of the selling prices v/s year based on transmission.*

This is another representation of the relationship between the transmission and the selling price, however this time selling price distribution is shown on different scatter plots based on tranmission.

Selling price trend is similar for the transmission, that is it is increasing over the time period. The price for automatic transmission models is higher as compared to manuals.

```
33  #Seperate plots of the selling prices v/s year based on transmission
34  ggplot(data = dsVeh, aes(x = year , y = sellingprice)) +
35    geom_point(mapping = aes(x=year, y=sellingprice, color=transmission)) +
36    facet_wrap(~transmission, nrow=2)
```

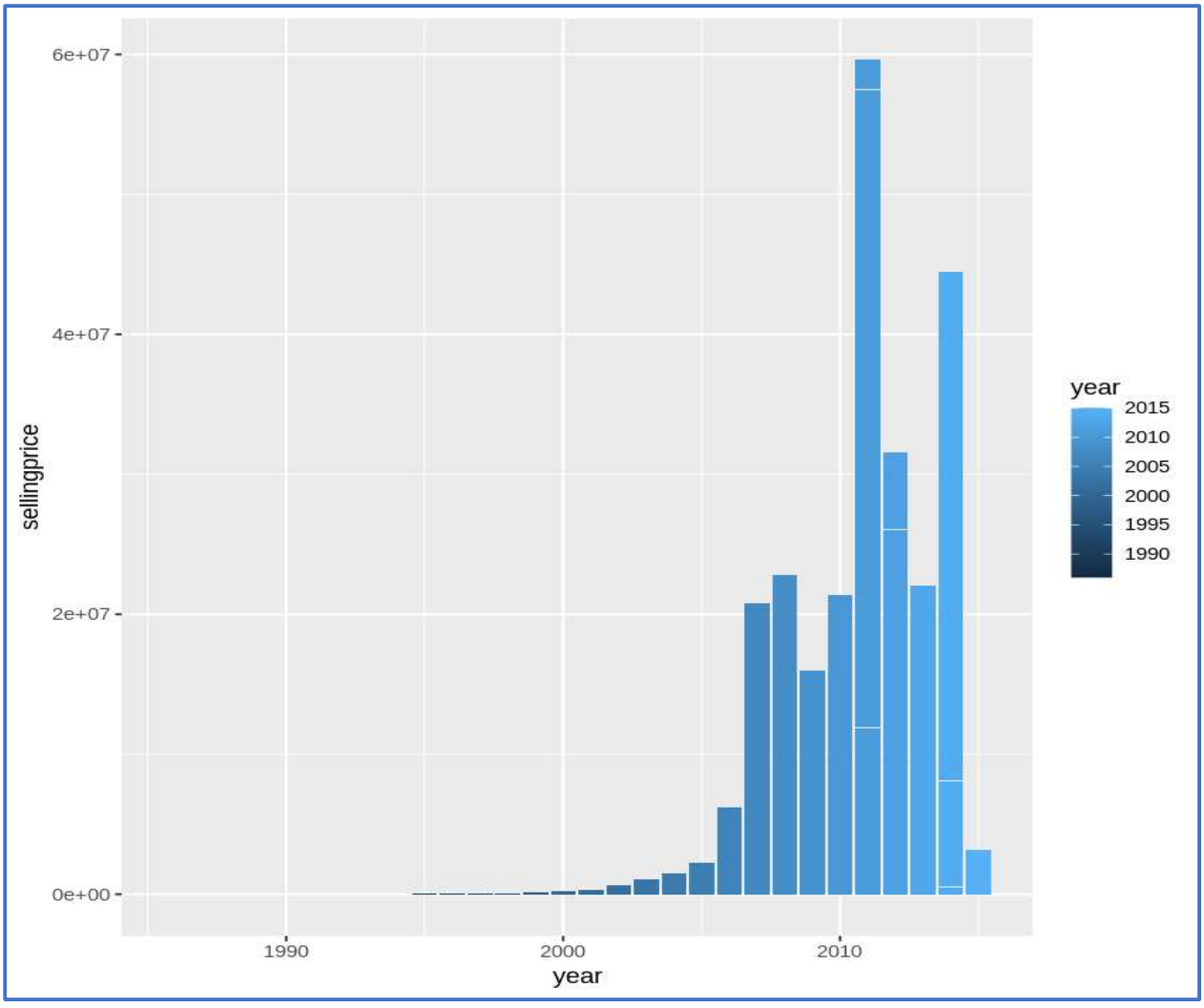


### *1 a) 4. Bar plotting of the selling price against year*

Bar plotting is done using ggplot function. On the X axis variable Year is provided while on the x axis we have the selling price, using color gradients to differentiate data points by year groupings.

- As the vehicle's production year increases, there is an increase in the selling price. This could be because of multiple factors like use of modern technology (as we have already seen in above plots how the automatic transmission influences the price of the vehicle)
- The color gradient helps in visualizing the data more effectively as the darker the blue color, the older the vehicle is.

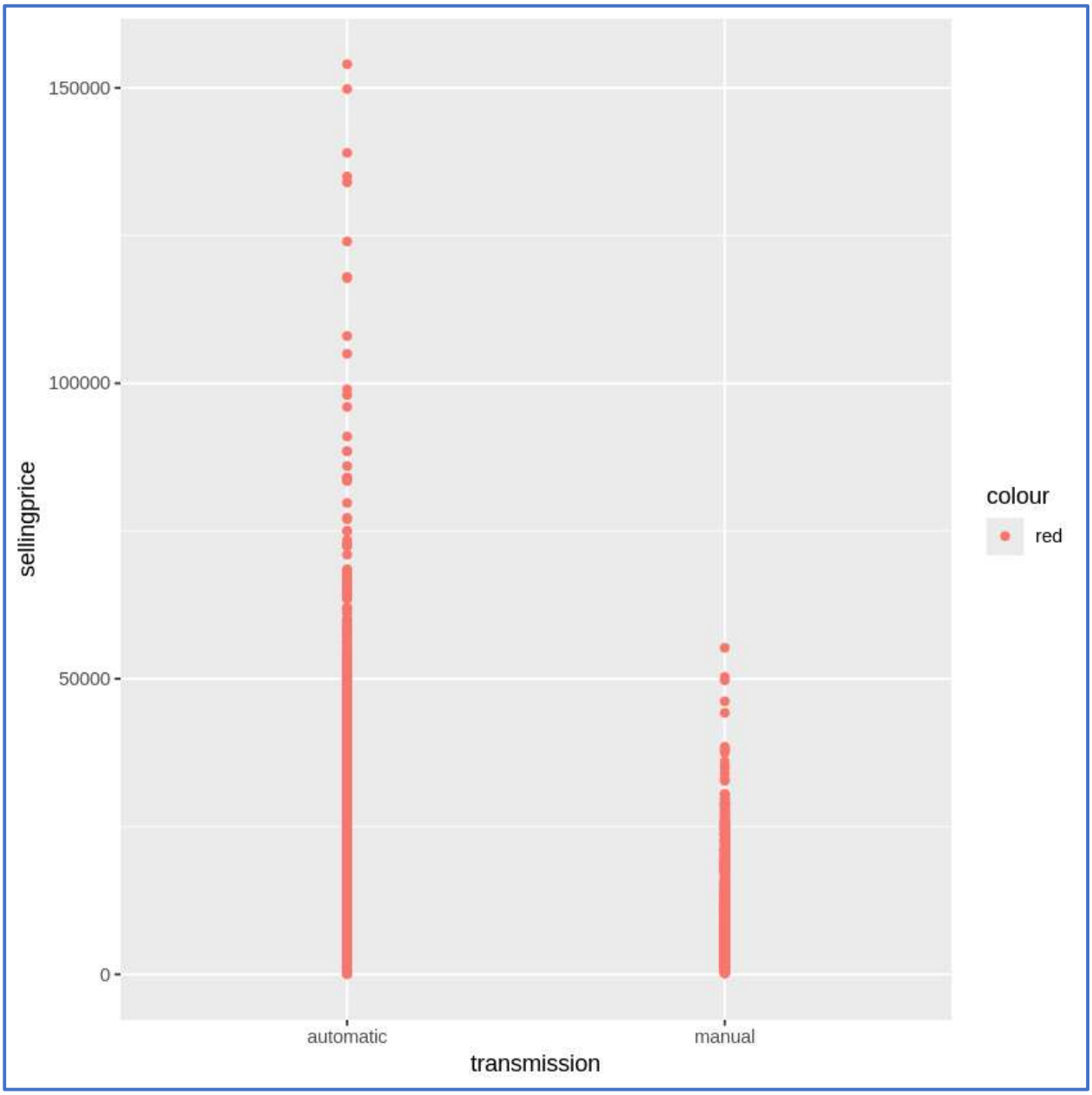
```
38 #Bar plotting of the selling price against year
39 ggplot(data = dsVeh, aes(x = year , y = sellingprice, fill=year)) +
40   geom_bar(stat="identity")
41
```



### *1 a) 5. Scatter plot of selling price based on two different transmissions.*

*Another way of scatter plotting is to keep the transmission on the x asix and price on the y axis. It is clear that the data is concentrated/scattered highly in the automatic transmission as compared to the manual transmission.*

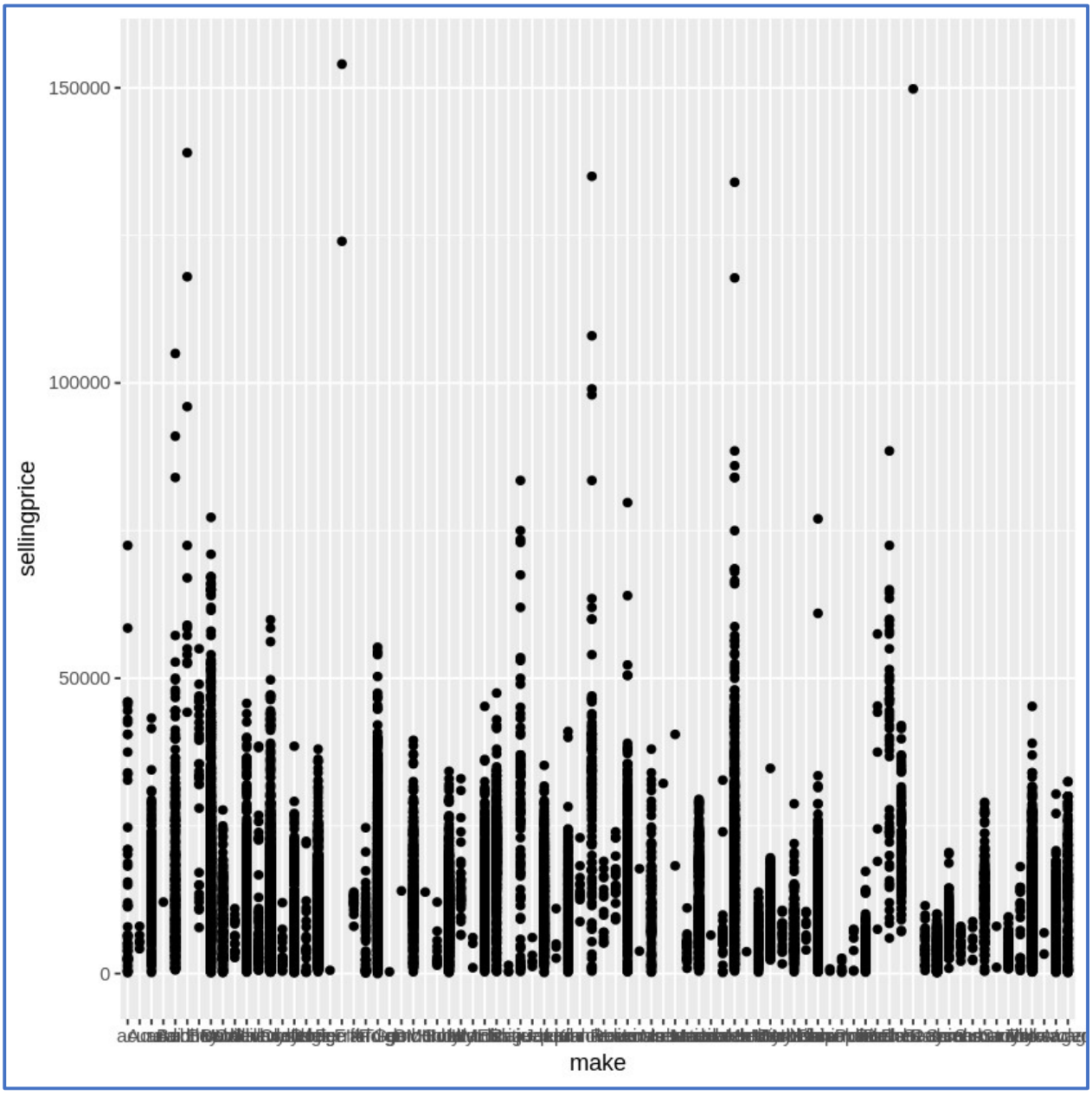
```
43
44  ggplot(data = dsVeh, aes(x = transmission,  y = sellingprice, color="red")) +
45    geom_point()
46
```



### *1 a) 6. Scatter plot of selling price based on different makes.*

The scatter plot represents the relationship between the selling price and makers of the vehicle. Some of the makers have higher selling prices compared to others.

```
47  #Scatter plot of selling price based on different makes.
48  ggplot(data = dsVeh, aes(x = make,  y = sellingprice)) +
49    geom_point()
```

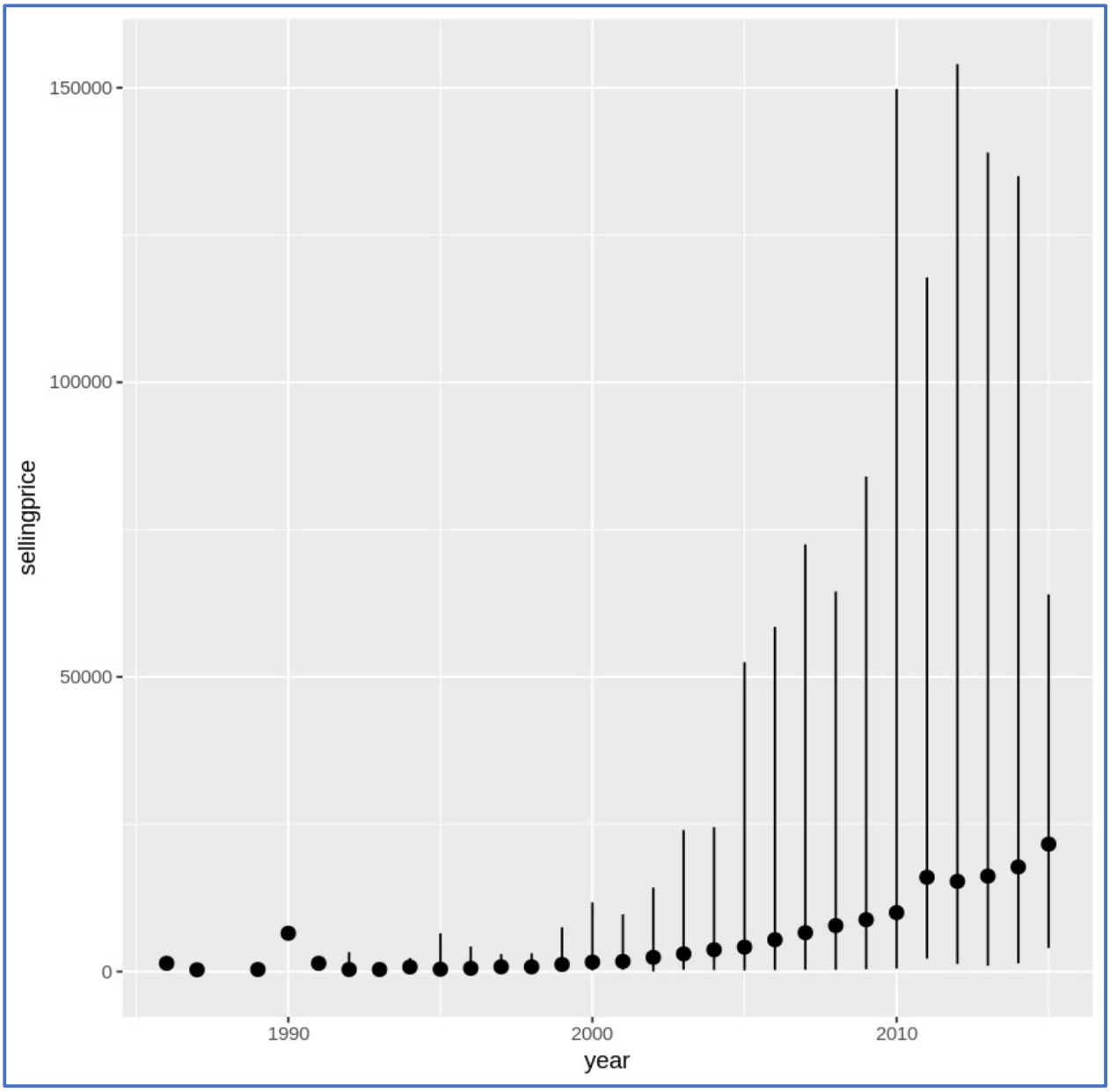


### *1 a) 7. Summary Plot of the selling price data against the year variable of the data set.*

stat_summary() function is used in the below plot to summarize the sellingprice (y value) at every unique year (x variable)

fun.ymin, fun.ymax are the min & max summary function respectively.

```
51  #Summary Plot of the selling price data against the year variable ofthe data set.
52  ggplot(data = dsVeh) +
53    stat_summary(
54      mapping = aes(x = year, y = sellingprice),
55      fun.ymin = min,
56      fun.ymax = max,
57      fun.y = median
58    )
```

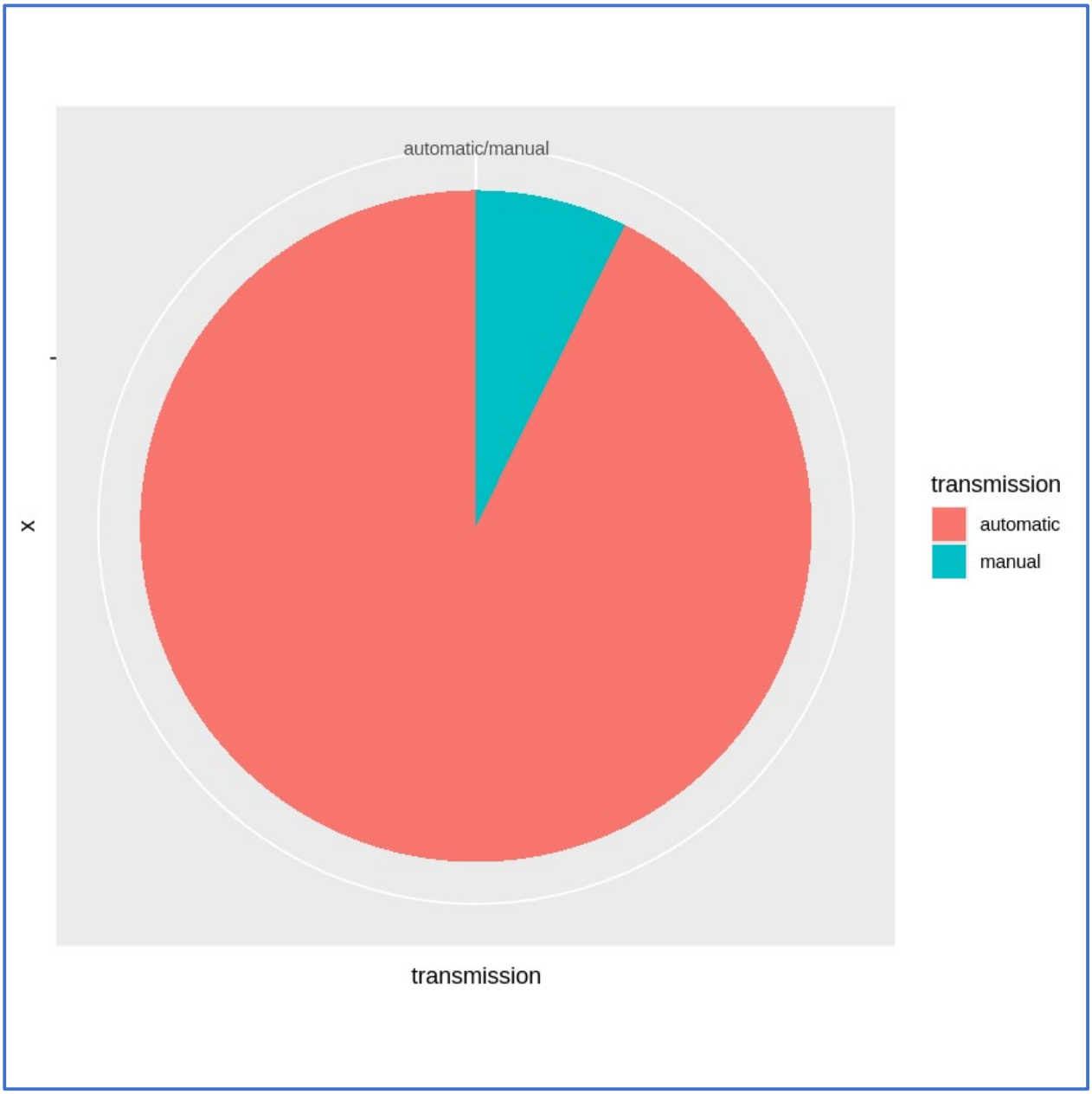


### *1 a) 8. Pie Chart distribution of the different transmissions in the dataset.*

Pie chart can be plotted using the ggplot function where x axis is kept as blank while y axis is assigned to the transmission vector.

Pie Chart distribution below shows relationship between the different transmissions in the dataset. Majority of the vehicles are automatic.

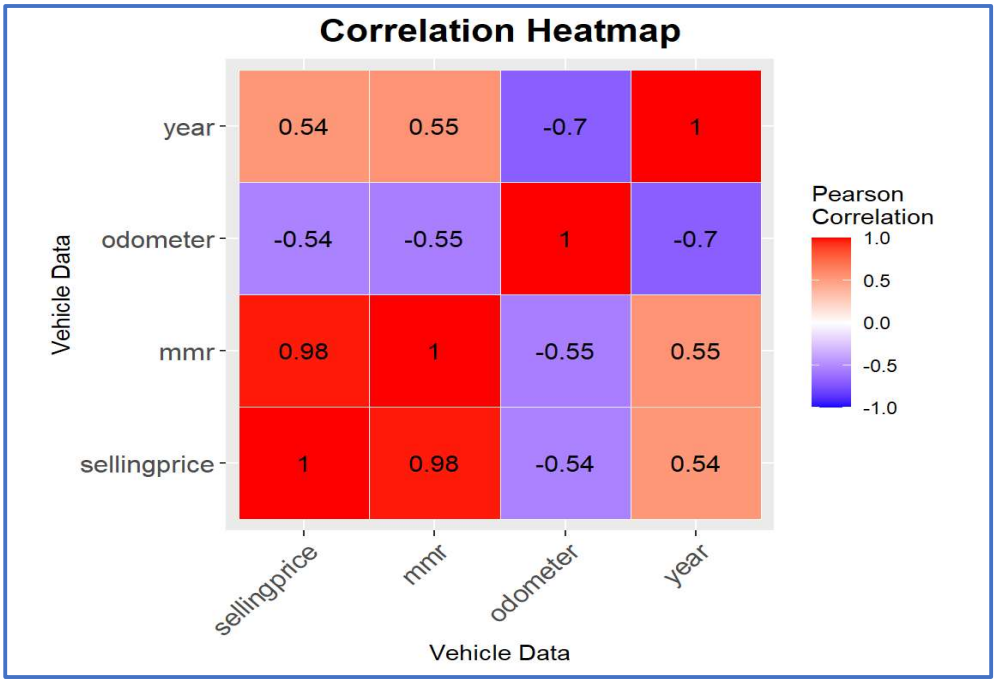
```
60  #Pie Chart distribution of the different transmission in the dataset.
61  ggplot(dsVeh, aes(x="", y=transmission, fill=transmission)) +
62    geom_bar(stat="identity", width=1) + coord_polar("y", start=0)
```



***1 a) 9. Correlation Heatmap - visualizing the correlation between the year, mmr, odometer and sellingprice. This helps us understand the relation between them.***

For correlation plotting, ggplot function is used. Melted_cormat is the dataset containing the correlation matrix, created using melt which changes the format to "long". Year, Odometer, MMR and Selling price columns are selected for correlation matrix. Negative correlations are colored blue, positive correlations are red, and correlations approximating to zero are white in color.

```
# This helps understand the relation between them.

cor_data <- dsVeh[c('sellingprice', 'mmr', 'odometer', 'year' )]

correlation_matrix <- cor(cor_data)

melted_cormat <- melt(correlation_matrix)

ggplot(data = melted_cormat, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1, 1), space = "Lab",
                       name = "Correlation") +
  geom_text(aes(label = round(value, 2)), color = "black", size = 4) +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1, size = 12),
        axis.text.y = element_text(size = 12),
        plot.title = element_text(hjust = 0.5, face = "bold", size = 16)) +
  labs(title = "Correlation Map", x = "Vehicle Data", y = "Vehicle Data") +
  coord_fixed()

```



## *Question 1 b) Consider one of continuous attributes, and compute central and variational measures.*

### *Continues variable selected: Selling Price*

For calculations of central and variational measures separate functions for mean, mode median, standard deviation and variance, are created.

```
 88  find_mode <- function(colData)
 89  {
 90    tblColData = table(colData)
 91    mode = names(tblColData)[which(tblColData==max(tblColData))]
 92    mode
 93  }
 94
 95  find_central_tendency <- function(colData)
 96  {
 97    mode <- find_mode(colData)
 98    mean <- mean(colData)
 99    median <- median(colData)
100    return(c(mode, mean, median))
101  }
102
103  find_variance <- function(colData)
104  {
105    varData <- var(colData)
106    stndDev <- sd(colData)
107    range <- max(colData) - min(colData)
108    iqr <- IQR(colData)
109    return (c(varData,stndDev,range,iqr))
110  }
```

```
112  find_centralTendency_and_variance <- function(dataColumn)
113  {
114    print("Mode, Mean & Median of dataset are")
115    for (x in find_central_tendency(dataColumn))
116    {
117      print(x)
118    }
119    print("Variance, standard deviation, range and IQR of dataset are")
120    for (x in find_variance(dataColumn))
121    {
122      print(x)
123    }
124  }
125
126  find_centralTendency_and_variance(dsVeh$sellingprice)
127
```

Parent function when executed provided following results.

```
> find_centralTendency_and_variance(dsVeh$sellingprice)
[1] "Mode, Mean & Median of dataset are"
[1] "7000"
[1] "12712.64805"
[1] "10800"
[1] "Variance, standard deviation, range and IQR of dataset are"
[1] 93075186
[1] 9647.548
[1] 153999
[1] 11100
```

```
> modeSellingPrice = find_mode(dsVehSp)
> print(paste("The mode of the Selling Price Column = ", modeSellingPrice))
[1] "The mode of the Selling Price Column =  7000"
>
> dsVehMean = mean(dsVehSp)
> print(paste("The mean of the Selling Price Column = ", dsVehMean))
[1] "The mean of the Selling Price Column =  12712.64805"
>
> dsVehMedian = median(dsVehSp)
> print(paste("The median of the Selling Price Column = ", dsVehMedian))
[1] "The median of the Selling Price Column =  10800"
>
> dsVehRange = max(dsVehSp) - min(dsVehSp)
> print(paste("The max of Selling Price Column = ", max(dsVehSp)))
[1] "The max of Selling Price Column =  154000"
> print(paste("The min of the Selling Price Column = ", min(dsVehSp)))
[1] "The min of the Selling Price Column =  1"
> print(paste("The range of the Selling Price Column = ", dsVehRange))
[1] "The range of the Selling Price Column =  153999"
>
> QSellingPrice = quantile(dsVeh$sellingprice)
> print("Quantile of the Selling Price column is ")
[1] "Quantile of the Selling Price column is "
> QSellingPrice
    0%    25%    50%    75%   100%
     1   6000  10800  17100 154000
```

```
> QSellingPrice1 = QSellingPrice[2]
> print("First Quantile of the Selling Price column is ")
[1] "First Quantile of the Selling Price column is "
> QSellingPrice1
 25%
6000
>
> QSellingPrice3 = QSellingPrice[4]
> print("Third Quantile of the Selling Price column is ")
[1] "Third Quantile of the Selling Price column is "
> QSellingPrice3
  75%
17100
>
> IQRSellingPrice = QSellingPrice3 - QSellingPrice1
> print("IQR of the Selling Price column is ")
[1] "IQR of the Selling Price column is "
> IQRSellingPrice
  75%
11100
>
> dsVehStndDev = sd(dsVehSp)
> print("Standard Deviation of the Selling Price column is ")
[1] "Standard Deviation of the Selling Price column is "
> dsVehStndDev
[1] 9647.548
>
> dsVehVar = var(dsVehSp)
> print("Variance of the Selling Price column is ")
[1] "Variance of the Selling Price column is "
> dsVehVar
[1] 93075186
```

***Summary***

- The most frequent value of the selling prices is 7000 which is the mode.
- The average selling price is 12712.65 which is the mean.
- The median of the selling price is 10800 (when sorted).
- The Difference between the maximum and minimum values is 153999 which is the range.
- 25th percentile or the first quantile is 6000.
- 50th percentile or the second quantile is 10800.

- 75th percentile or the third quantile is 17100.
- Difference between the third and first quartiles or the interquartile range is 11100.
- The standard deviation is 9647.55. It provides spread information of the data.
- The variance is 93075186. It provides spread information of the data.

## *Question 1 (c) For a particular variable of the dataset, use Chebyshev's rule, and propose one-sigma interval.*

According to Chebyshev's theorem for any distribution (bell shaped or non-bell shaped) at least (1 - 1\k**2) proportion of values lies within k standard deviation for k > 1.

To find outliers

1. Find the lower bound = mean - k * standard dev
2. Find the upper bound = mean + k * standard dev
3. outliers will be found below the lower bound value and above the upper bound

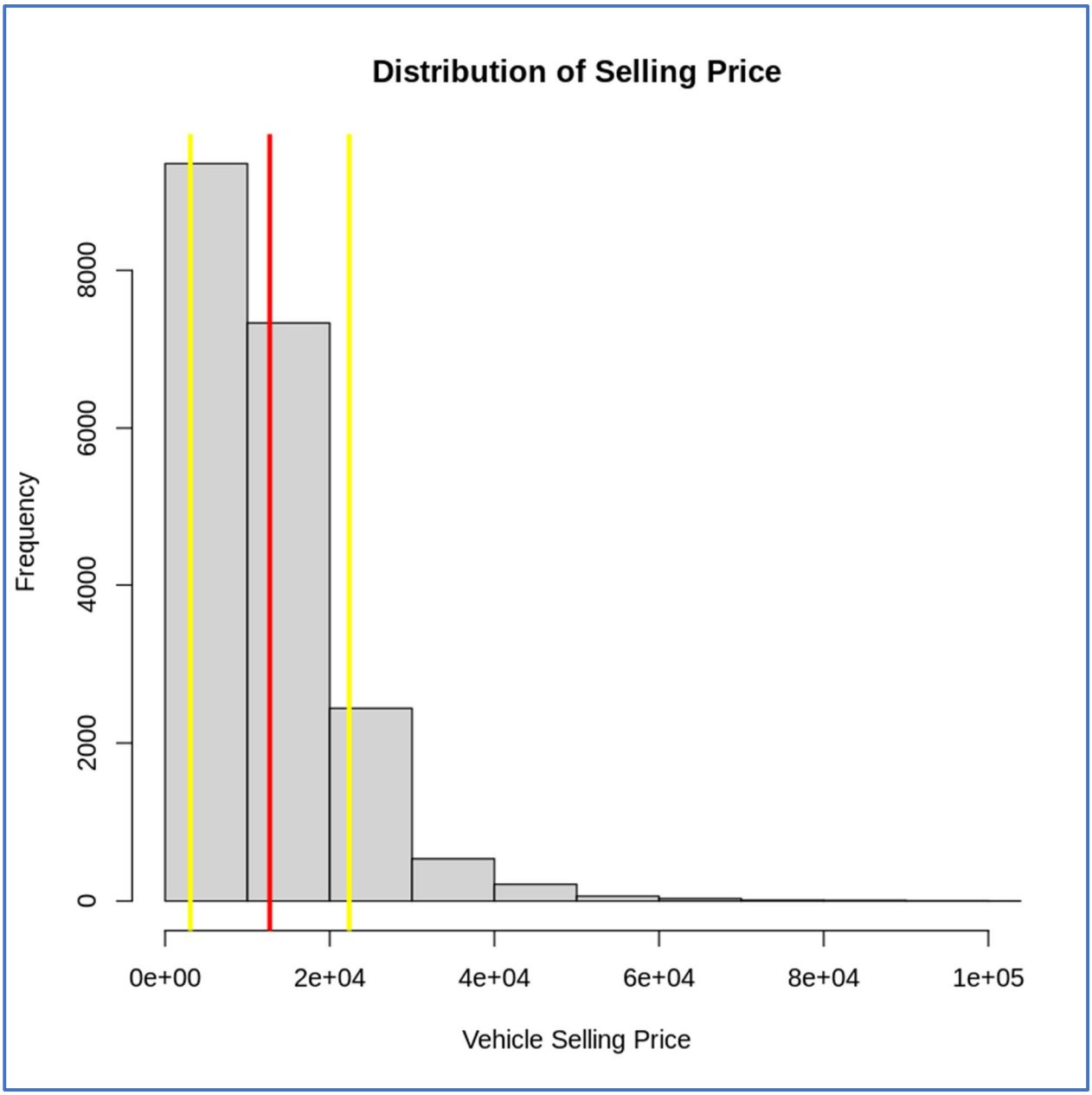
outliers within k = 1 ( 1 * standard dev ) will be

```
k = 1
lowerRange = dsVehMean - k * dsVehStndDev
upperRange = dsVehMean + k * dsVehStndDev
print(paste("Taking k = 1, the lower range of sellingprice = ",round(lowerRange, 2)))
print(paste("Taking k = 1, the upper range of sellingprice = ",round(upperRange, 2)))

outliers = dsVehSp < lowerRange | dsVehSp > upperRange
totalOutliers =  sum(outliers)
print(paste("Total Outliers = ", totalOutliers))

dsVehSellingPriceLowerRange =  filter(dsVeh, sellingprice < lowerRange)
dsVehSellingPriceUpperRange =  filter(dsVeh, sellingprice > upperRange)

print(paste("Lower Range Outliers Count = ", count(dsVehSellingPriceLowerRange)))
print(paste("Upper Range Outliers Count ", count(dsVehSellingPriceUpperRange)))
print(paste("Total Outliers = ", count(dsVehSellingPriceLowerRange) + count(dsVehSellingPriceUpperRange)))

#For k = 1, Mean is shown below in red line, the standard deviation is shown as yellow line.

hist(dsVehSp, xlab = "Vehicle Selling Price", ylab = "Frequency", main="Distribution of Selling Price" , xlim = c(0, 100000))
abline(v = mean(dsVehSp), col="red", lwd = 3)
abline(v = lowerRange, col="yellow", lwd = 3)
abline(v = upperRange, col="yellow", lwd = 3)
```

```
> k = 1
> lowerRange = dsVehMean - k * dsVehStndDev
> upperRange = dsVehMean + k * dsVehStndDev
> print(paste("Taking k = 1, the lower range of sellingprice = ",round(lowerRange, 2)))
[1] "Taking k = 1, the lower range of sellingprice =  3065.1"
> print(paste("Taking k = 1, the upper range of sellingprice = ",round(upperRange, 2)))
[1] "Taking k = 1, the upper range of sellingprice =  22360.2"
>
>
> outliers = dsVehSp < lowerRange | dsVehSp > upperRange
> totalOutliers =  sum(outliers)
> print(paste("Total Outliers = ", totalOutliers))
[1] "Total Outliers =  4421"
>
> dsVehSellingPriceLowerRange =  filter(dsVeh, sellingprice < lowerRange)
> dsVehSellingPriceUpperRange =  filter(dsVeh, sellingprice > upperRange)
>
> print(paste("Lower Range Outliers Count = ", count(dsVehSellingPriceLowerRange)))
[1] "Lower Range Outliers Count =  1949"
> print(paste("Upper Range Outliers Count ", count(dsVehSellingPriceUpperRange)))
[1] "Upper Range Outliers Count  2472"
> print(paste("Total Outliers = ", count(dsVehSellingPriceLowerRange) + count(dsVehSellingPriceUpperRange)))
[1] "Total Outliers =  4421"
>
```

```r
#For k = 1, Mean is shown below in red line, the standard deviation is shown as yellow line.

hist(dsVehSp, xlab = "Vehicle Selling Price", ylab = "Frequency", main="Distribution of Selling Price" ,
     xlim = c(0, 100000))
abline(v = mean(dsVehSp), col='red', lwd = 3)
abline(v = lowerRange, col='yellow', lwd = 3)
abline(v = upperRange, col='yellow', lwd = 3)
```

The Yellow lines are the k=1 standard deviations from the red line which is mean. Outliers are the points that lies on the left of the left yellow line and on the right of the right yellow line.

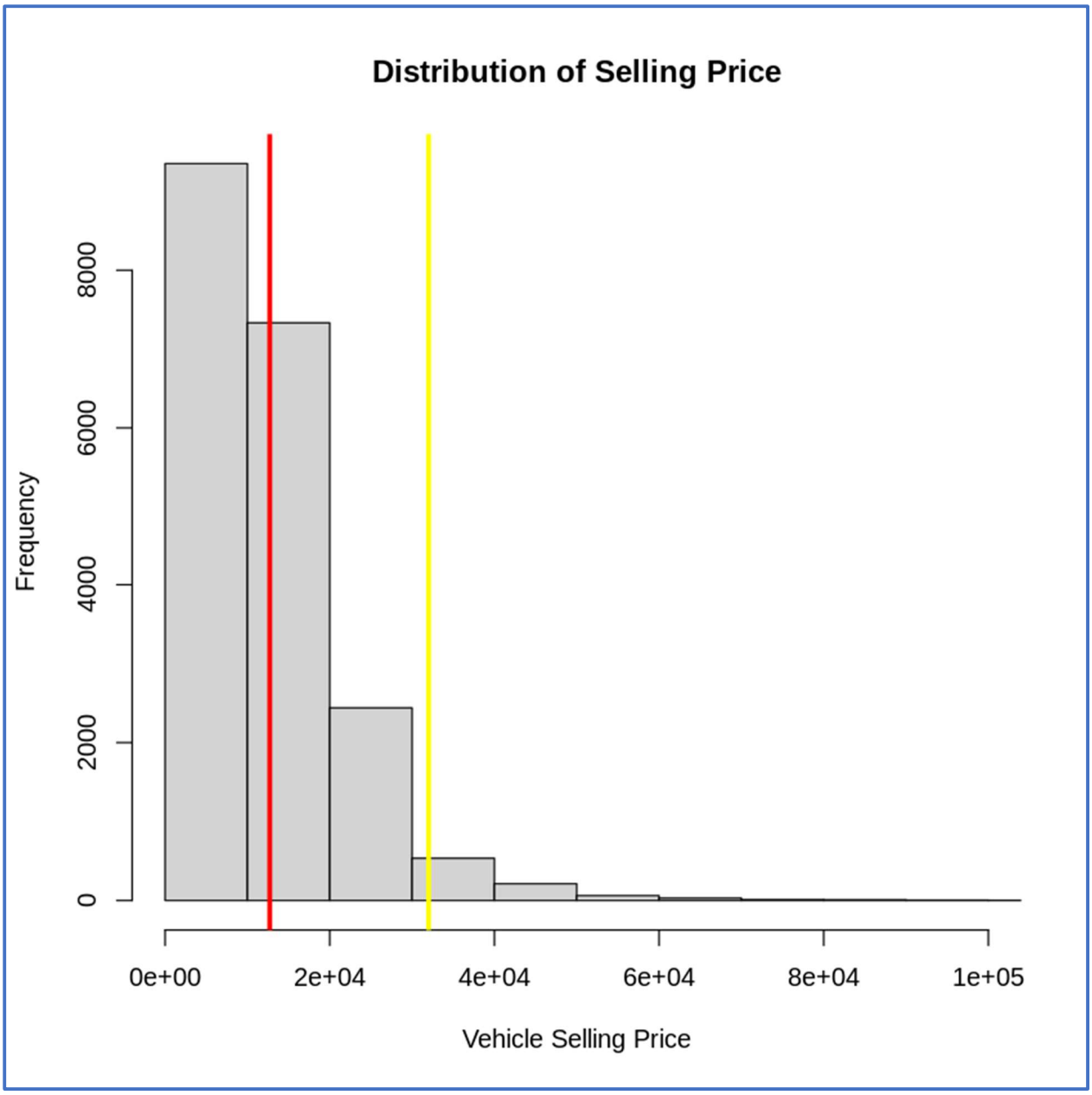


```
209 k = 2
210 lowerRangeK2 = dsVehMean - k * dsVehStndDev
211 upperRangeK2 = dsVehMean + k * dsVehStndDev
212 print(paste("Taking k = 2, the lower range of sellingprice = ",round(lowerRangeK2, 2)))
213 print(paste("Taking k = 2, the upper range of sellingprice = ",round(upperRangeK2, 2)))
214
215 outliersK2 = dsVehSp < lowerRangeK2 | dsVehSp > upperRangeK2
216 totalOutliersK2 =  sum(outliersK2)
217 print(paste("Total Outliers for k=2 = ", totalOutliersK2))
218
219 dsVehSellingPriceLowerRangeK2 =  filter(dsVeh, sellingprice < lowerRangeK2)
220 dsVehSellingPriceUpperRangeK2 =  filter(dsVeh, sellingprice > upperRangeK2)
221
222 print(paste("Lower Range Outliers Count for k = 2= ", count(dsVehSellingPriceLowerRangeK2)))
223 print(paste("Upper Range Outliers Count for k = 2", count(dsVehSellingPriceUpperRangeK2)))
224 print(paste("Total Outliers for k = 2  ", count(dsVehSellingPriceLowerRangeK2) +
225               count(dsVehSellingPriceUpperRangeK2)))
```

```
[1] "Taking k = 2, the lower range of sellingprice =  -6582.45"
> print(paste("Taking k = 2, the upper range of sellingprice = ",round(upperRangeK2, 2)))
[1] "Taking k = 2, the upper range of sellingprice =  32007.74"
>
> outliersK2 = dsVehSp < lowerRangeK2 | dsVehSp > upperRangeK2
> totalOutliersK2 =  sum(outliersK2)
> print(paste("Total Outliers for k=2 = ", totalOutliersK2))
[1] "Total Outliers for k=2 =  716"
>
> dsVehSellingPriceLowerRangeK2 =  filter(dsVeh, sellingprice < lowerRangeK2)
> dsVehSellingPriceUpperRangeK2 =  filter(dsVeh, sellingprice > upperRangeK2)
>
> print(paste("Lower Range Outliers Count for k = 2= ", count(dsVehSellingPriceLowerRangeK2)))
[1] "Lower Range Outliers Count for k = 2=  0"
> print(paste("Upper Range Outliers Count for k = 2", count(dsVehSellingPriceUpperRangeK2)))
[1] "Upper Range Outliers Count for k = 2 716"
> print(paste("Total Outliers for k = 2  ", count(dsVehSellingPriceLowerRangeK2) + count(dsVehSellingPriceUpperR
geK2)))
[1] "Total Outliers for k = 2   716"
```

For k = 2, Mean is shown below in red line, the standard deviation is shown as yellow line. As per the Chebyshev's rule (1 - 1\k*2) which is 3\4. Therefore at least 75% of the total selling price distribution will be withing range of (mean -2 standard dev) and (mean + 2* standard deviation). This comes true in our case as 75% of 20000 is 15000 and 716 lies outside the k=2 standard deviation range.

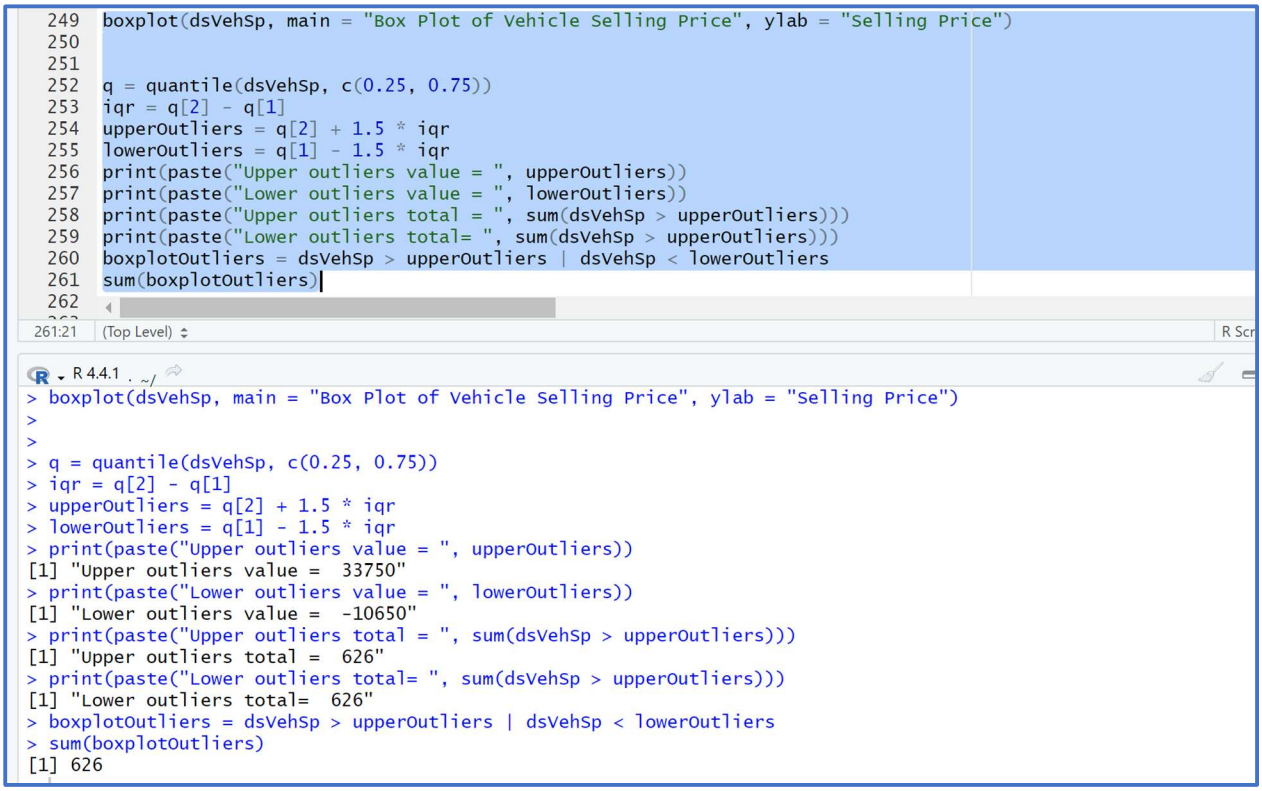
```
209 k = 2
210 lowerRangeK2 = dsVehMean - k * dsVehStndDev
211 upperRangeK2 = dsVehMean + k * dsVehStndDev
212 print(paste("Taking k = 2, the lower range of sellingprice = ",round(lowerRangeK2, 2)))
213 print(paste("Taking k = 2, the upper range of sellingprice = ",round(upperRangeK2, 2)))
214
215 outliersK2 = dsVehSp < lowerRangeK2 | dsVehSp > upperRangeK2
216 totalOutliersK2 =  sum(outliersK2)
217 print(paste("Total Outliers for k=2 = ", totalOutliersK2))
218
219 dsVehSellingPriceLowerRangeK2 =  filter(dsVeh, sellingprice < lowerRangeK2)
220 dsVehSellingPriceUpperRangeK2 =  filter(dsVeh, sellingprice > upperRangeK2)
221
222 print(paste("Lower Range Outliers Count for k = 2= ", count(dsVehSellingPriceLowerRangeK2)))
223 print(paste("Upper Range Outliers Count for k = 2", count(dsVehSellingPriceUpperRangeK2)))
224 print(paste("Total Outliers for k = 2  ", count(dsVehSellingPriceLowerRangeK2) +
225               count(dsVehSellingPriceUpperRangeK2)))
```


Distribution of Selling Price
Frequency
0
2000
4000
6000
8000
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
Vehicle Selling Price

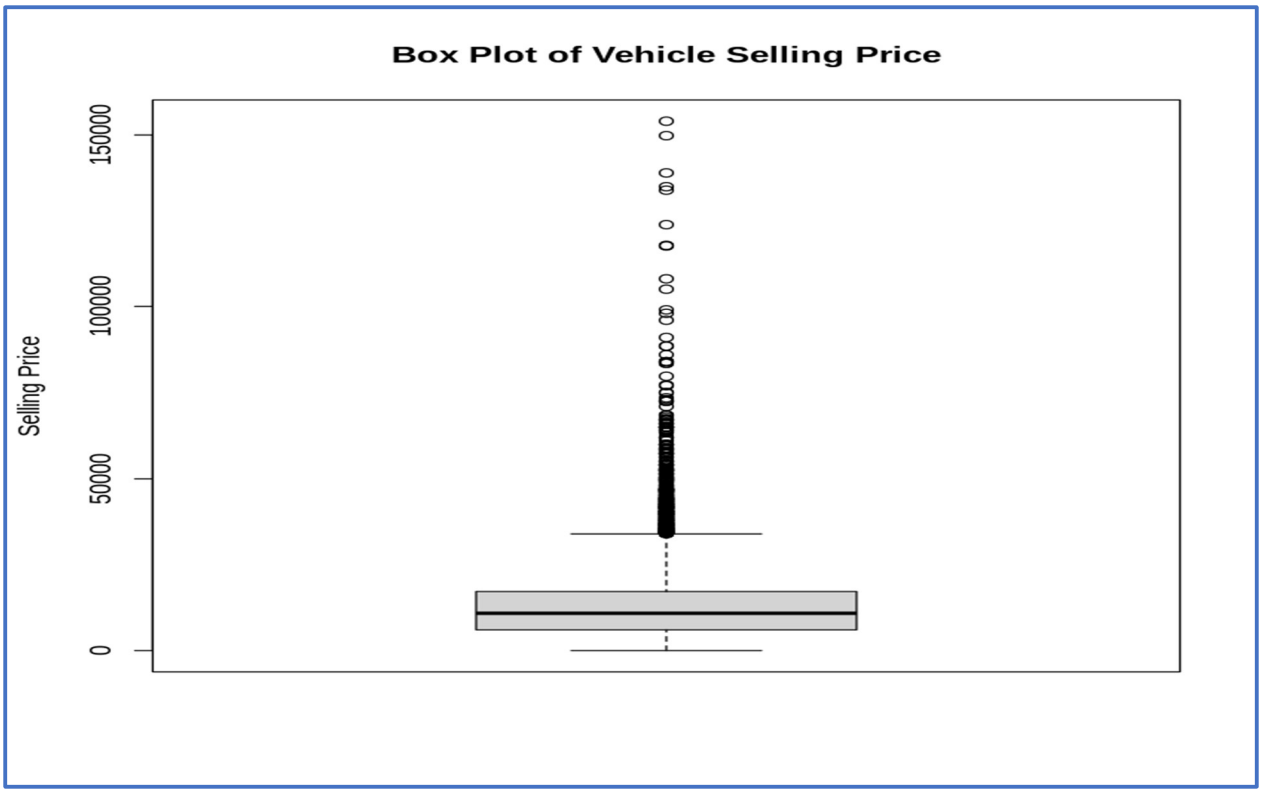
## *Question 1 (d) Explain how the box-plot technique can be used to detect outliers. Apply this technique for one attribute of the dataset.*

Boxplot is constructed using the min max, median, first and third quantile. A box is drawn between the lower and upper quartiles. A line across the box is the median. The straight lines are connected to min and max which are also known as whiskers. We can easily find out

1. The lower and upper quartiles, Q1 and Q3
2. The interquartile range (IQR), the distance between the lower and upper quartiles
3. The most extreme (lowest and highest) values
4. The symmetry or asymmetry of the distribution of scores (R.Lyman, 106)



```
249 boxplot(dsVehSp, main = "Box Plot of Vehicle Selling Price", ylab = "Selling Price")
250
251
252 q = quantile(dsVehSp, c(0.25, 0.75))
253 iqr = q[2] - q[1]
254 upperOutliers = q[2] + 1.5 * iqr
255 lowerOutliers = q[1] - 1.5 * iqr
256 print(paste("Upper outliers value = ", upperOutliers))
257 print(paste("Lower outliers value = ", lowerOutliers))
258 print(paste("Upper outliers total = ", sum(dsVehSp > upperOutliers)))
259 print(paste("Lower outliers total= ", sum(dsVehSp > upperOutliers)))
260 boxplotOutliers = dsVehSp > upperOutliers | dsVehSp < lowerOutliers
261 sum(boxplotOutliers)
262
261:21 (Top Level)

R 4.4.1 . ~/
> boxplot(dsVehSp, main = "Box Plot of Vehicle Selling Price", ylab = "Selling Price")
>
>
> q = quantile(dsVehSp, c(0.25, 0.75))
> iqr = q[2] - q[1]
> upperOutliers = q[2] + 1.5 * iqr
> lowerOutliers = q[1] - 1.5 * iqr
> print(paste("Upper outliers value = ", upperOutliers))
[1] "Upper outliers value =  33750"
> print(paste("Lower outliers value = ", lowerOutliers))
[1] "Lower outliers value =  -10650"
> print(paste("Upper outliers total = ", sum(dsVehSp > upperOutliers)))
[1] "Upper outliers total =  626"
> print(paste("Lower outliers total= ", sum(dsVehSp > upperOutliers)))
[1] "Lower outliers total=  626"
> boxplotOutliers = dsVehSp > upperOutliers | dsVehSp < lowerOutliers
> sum(boxplotOutliers)
[1] 626
```




First 626 outlier data entries can be seen below.

```
outliersData = dsVehSp[dsVehSp > upperOutliers | dsVehSp < lowerOutliers]
print(outliersData)
```

```
[273] 40500 40500 43250 37500 57250 37500 44250 73000 55250 45250 41500 73500 75000 83500 34250 67500 41750
[290] 38000 43000 34250 72500 34750 83500 38000 62000 60000 99000 135000 98000 108000 63500 57500 35000 52000
[307] 88500 59000 41750 47000 55500 44000 86000 37250 38500 134000 84000 68500 35500 68000 84000 63500 75000
[324] 41500 48500 40250 139000 49500 71000 43500 41500 34250 53000 37250 42500 52500 40000 42750 42750 43500
[341] 41500 42000 34750 44500 52500 35250 35000 44000 39000 38000 39750 34300 40000 34700 41500 42000 44500
[358] 41500 105000 53000 43250 35750 46750 36500 36750 41250 39500 44000 41250 118000 44000 36250 46250 35000
[375] 39000 36000 34250 40000 47000 35250 43500 42750 34000 40750 41500 39000 42750 50250 36750 41750 36750
[392] 40500 36500 117800 39250 124000 34250 34000 45300 52750 67000 54000 64500 34250 58500 44250 52500 56500
[409] 38000 79750 62000 50500 45000 37000 47500 42250 44500 57250 39750 52750 34500 37000 48000 77000 44500
[426] 35000 51000 42000 34000 34000 35500 42300 44000 36300 37500 44000 43000 47200 47000 41900 46600 44500
[443] 46400 43500 64000 39000 42000 47000 56200 35500 41900 38500 37800 37200 42750 38700 41500 35400 42000
[460] 41500 36750 39750 35000 34000 37950 34750 38000 46200 55000 37000 34000 36000 36000 35000 36500 34200
[477] 42600 34000 40000 39800 49750 34400 34100 38000 34500 34300 36300 34500 37000 39200 35300 36200 36000
[494] 35100 39500 38600 35200 34400 37200 41700 35000 40400 34400 34100 45100 34000 34000 35900 35900 34400
[511] 40750 41750 41250 35100 34200 40600 34250 38000 33800 36500 54000 38600 35500 35400 33900 52250 50500
[528] 37500 50000 44600 52500 43500 37000 45800 43500 60000 48250 43000 47500 37000 38200 42500 37200 34000
[545] 35500 38000 39800 44000 44933 34600 68500 34500 34750 36750 46500 46200 65000 40000 36750 149800 84000
[562] 34000 36000 37250 38000 36400 37750 37800 55000 34000 42100 36900 40000 35300 35800 35200 38500 34100
[579] 36000 34700 39300 35000 34000 45400 36800 39100 37000 46900 34200 46600 34200 34800 35500 38500 38000
[596] 37250 39500 39800 39000 39500 35400 54500 40800 36500 47500 34700 35300 40800 57500 59000 72500 38500
[613] 54000 43200 39000 42000 34000 36500 44000 37200 43000 36250 58500 35000 50300 38250
```

## *Question 2 (a) - Select four variables of the dataset and propose an appropriate probability model to quantify uncertainty of each variable.*

1. ***MMR*** - Provides the most accurate wholesale valuations based on vehicle year, make, model and style, among other criteria.
   - It is a continues variable
   - Probability Model for this variable is Normal Distribution.
   - The valuation might vary but most of them will be around the mean.

2. ***State*** - State of the vehicle
   - It is a categorical variable (more than 2 categories)
   - Probability Model for this variable is Multinational Distribution.
   - There can be multiple probability outcomes as the state variable has multiple categories.

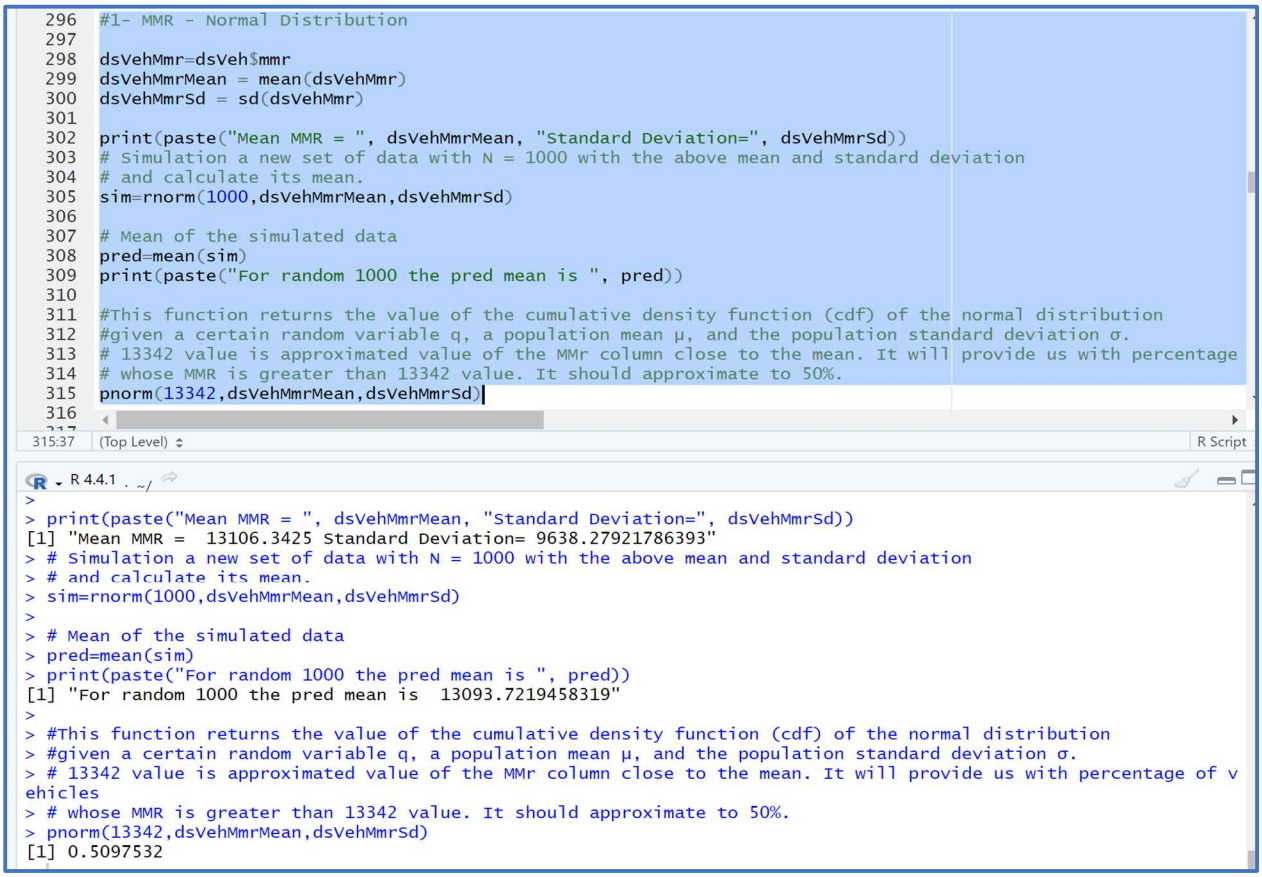
3. ***Transmission*** - Provides the car transmission and the current data is categorized into Manual and Automatic
   - It is a Binary Variable.
   - Probability Model for this variable is Bernoulli Distribution.
   - Transmission in the car can be encodes as a binary outcome (yes/no, 1,0 or in other words Automatic\Manual).

4. ***Odometer*** - Provides Odometer reading of the car.
   - It is a continuous variables that possess positive and skewed distributions
   - Probability Model for this variable is Gamma Distribution.

```
286
287  # Selecting variables for probability models
288  Selected_Data <- dsVeh[, c('mmr', 'state', 'transmission', 'odometer')]
289  # Previewing the selected data
290  head(Selected_Data)
291
292
292:1  (Top Level)                                                    R Scrip

R 4.4.1 . ~/
> # Selecting variables for probability models
> Selected_Data <- dsVeh[, c('mmr', 'state', 'transmission', 'odometer')]
> # Previewing the selected data
> head(Selected_Data)
    mmr state transmission odometer
1 20500    ca    automatic    16639
2 20800    ca    automatic     9393
3 31900    ca    automatic     1331
4 27500    ca    automatic    14282
5 66000    ca    automatic     2641
6 15350    ca    automatic     5554
```

## *Question 2 (b) For each model in part (a), estimate the parameters of the model.*

1. ***MMR*** – Normal Distribution.



```
#1- MMR - Normal Distribution

dsVehMmr=dsVeh$mmr
dsVehMmrMean = mean(dsVehMmr)
dsVehMmrSd = sd(dsVehMmr)

print(paste("Mean MMR = ", dsVehMmrMean, "Standard Deviation=", dsVehMmrSd))
# Simulation a new set of data with N = 1000 with the above mean and standard deviation
# and calculate its mean.
sim=rnorm(1000,dsVehMmrMean,dsVehMmrSd)

# Mean of the simulated data
pred=mean(sim)
print(paste("For random 1000 the pred mean is ", pred))

#This function returns the value of the cumulative density function (cdf) of the normal distribution
#given a certain random variable q, a population mean μ, and the population standard deviation σ.
# 13342 value is approximated value of the MMr column close to the mean. It will provide us with percentage
# whose MMR is greater than 13342 value. It should approximate to 50%.
pnorm(13342,dsVehMmrMean,dsVehMmrSd)
```

```
> print(paste("Mean MMR = ", dsVehMmrMean, "Standard Deviation=", dsVehMmrSd))
[1] "Mean MMR =  13106.3425 Standard Deviation= 9638.27921786393"
> # Simulation a new set of data with N = 1000 with the above mean and standard deviation
> # and calculate its mean.
> sim=rnorm(1000,dsVehMmrMean,dsVehMmrSd)
>
> # Mean of the simulated data
> pred=mean(sim)
> print(paste("For random 1000 the pred mean is ", pred))
[1] "For random 1000 the pred mean is  13093.7219458319"
>
> #This function returns the value of the cumulative density function (cdf) of the normal distribution
> #given a certain random variable q, a population mean μ, and the population standard deviation σ.
> # 13342 value is approximated value of the MMr column close to the mean. It will provide us with percentage of v
ehicles
> # whose MMR is greater than 13342 value. It should approximate to 50%.
> pnorm(13342,dsVehMmrMean,dsVehMmrSd)
[1] 0.5097532
```

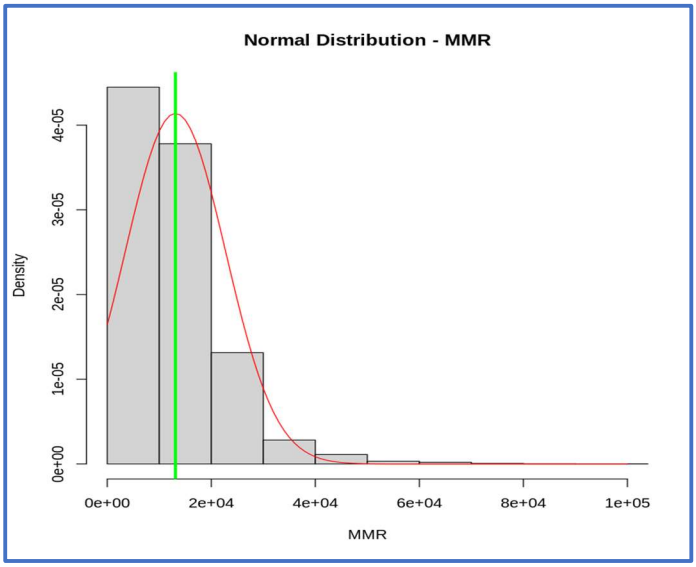
For the distribution with mean = 13106.3425 and standard deviation of 9638.278, The probability that the randomly selected selling price will be less than 13342 is 0.509. This can also be checked by calculating the z-value which comes out to be 0.02445 (below code). This means that the 13342 price is just 0.02445 standard deviation away from the mean. Again the pnorm() can be used to find out the area under the curve or the probability of the event.

```
356  #This implies that for the distribution with mean = 13106.3425 and standard deviation of 9638.278,
357  # The probability that the randomly selected selling price will be less than 13342 is 0.509.
358  # This can also be checked by calculating the
359
360  zvalue = (13342 - dsVehMmrMean)/dsVehMmrSd
361
362  print(paste("Z-Value = ",zvalue))
363
364  # Using the pnomr function we can calculate the area under the zvalue.
365
366  print(paste("Area under Z value", zvalue, "is ",pnorm(zvalue) * 100))
367
368
369  hist(dsVeh$mmr, main = "Normal Distribution - MMR", xlab = "MMR", freq = FALSE, xlim=c(0,100000))
370  curve(dnorm(x, mean = dsVehMmrMean, sd = dsVehMmrSd), add = TRUE, col = "red")
371
367:3  (Top Level)
```

```
R 4.4.1 · ~/
> zvalue = (13342 - dsVehMmrMean)/dsVehMmrSd
>
> print(paste("Z-Value = ",zvalue))
[1] "Z-Value =  0.0244501632161915"
>
> # Using the pnomr function we can calculate the area under the zvalue.
>
> print(paste("Area under Z value", zvalue, "is ",pnorm(zvalue) * 100))
[1] "Area under Z value 0.0244501632161915 is  50.9753232095912"
```

The histogram shows the normal distribution plot below. Green line shows the mean. Red line is plotted using curve function over (pdf) of the normal distribution.

```
316
317  hist(dsVeh$mmr, main = "Normal Distribution - MMR", xlab = "MMR", freq = FALSE, xlim=c(0,100000))
318  curve(dnorm(x, mean = dsVehMmrMean, sd = dsVehMmrSd), add = TRUE, col = "red")
319  abline(v = dsVehMmrMean, col='green', lwd = 3)
320
```



2. ***State*** – Multinomial Distribution.
The multinomial distribution is a distribution function for discrete processes in which fixed probabilities prevail for each independently generated value.

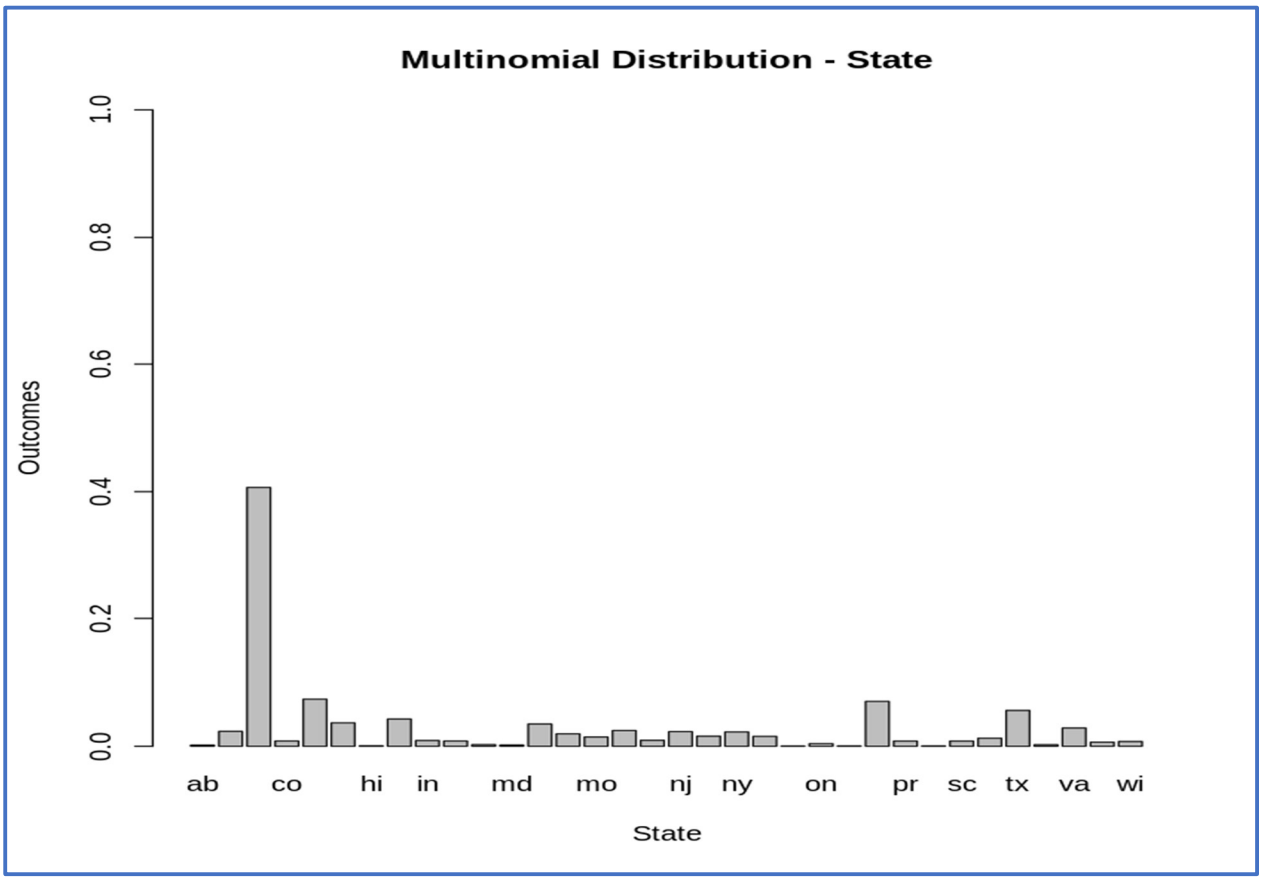
The different states have different number of registered vehicles. If each state registered vehicles are grouped and divided by the total number, we get the probability table per state.

```
#2- State - Multinational Distribution

dsVehState=dsVeh$state
tbldsVehState=table(dsVehState);

print("Number of vehicles per state")
tbldsVehState

# Total number of vehicles in all the states.
sum(tbldsVehState)

dsVehStateMode=names(tbldsVehState)[which(tbldsVehState==max(tbldsVehState))];
print(paste("The Most vehicles belongs to the state", dsVehStateMode))

# Probability of number of vehicles per state. That means number of vehicles per state
# divided by total number of vehicles.
probPerState=tbldsVehState/sum(tbldsVehState);
probPerState
```

```
> #2- State - Multinational Distribution
>
> dsVehState=dsVeh$state
> tbldsVehState=table(dsVehState);
>
> print("Number of vehicles per state")
[1] "Number of vehicles per state"
> tbldsVehState
dsVehState
  ab   az   ca   co   fl   ga   hi   il   in   la   ma   md   mi   mn   mo   nc   ne   nj   nv   ny   oh   ok
  38  466 8133  164 1473  732   14  852  180  164   51   40  697  389  289  492  187  457  316  448  310    7
  on   or   pa   pr   qc   sc   tn   tx   ut   va   wa   wi
  82   10 1404  163   10  163  252 1121   48  571  127  150
>
> # Total number of vehicles in all the states.
> sum(tbldsVehState)
[1] 20000
>
> dsVehStateMode=names(tbldsVehState)[which(tbldsVehState==max(tbldsVehState))];
> print(paste("The Most vehicles belongs to the state", dsVehStateMode))
[1] "The Most vehicles belongs to the state ca"
>
> # Probability of number of vehicles per state. That means number of vehicles per state
> # divided by total number of vehicles.
> probPerState=tbldsVehState/sum(tbldsVehState);
> probPerState
dsVehState
     ab      az      ca      co      fl      ga      hi      il      in      la      ma      md      mi
0.00190 0.02330 0.40665 0.00820 0.07365 0.03660 0.00070 0.04260 0.00900 0.00820 0.00255 0.00200 0.03485
     mn      mo      nc      ne      nj      nv      ny      oh      ok      on      or      pa      pr
0.01945 0.01445 0.02460 0.00935 0.02285 0.01580 0.02240 0.01550 0.00035 0.00410 0.00050 0.07020 0.00815
     qc      sc      tn      tx      ut      va      wa      wi
0.00050 0.00815 0.01260 0.05605 0.00240 0.02855 0.00635 0.00750
```

```
339
340 barplot(probPerState, main = "Multinomial Distribution - State",
341         xlab = "State", ylab = "Outcomes", ylim = c(0, 1))
342
```

The distribution below shows the probability of the outcome of each state.



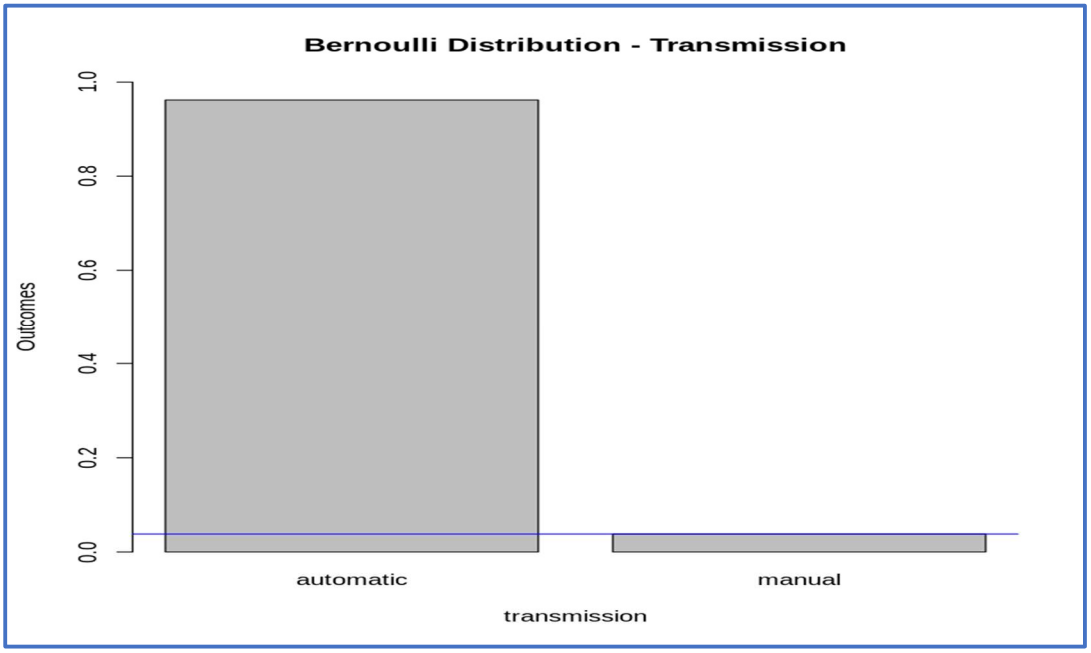
## 3. *Transmission - Bernoulli Distribution.*

The Bernoulli distribution is a discrete distribution having two possible outcome 0,1 in which 1 (success) occurs with probability p and 0 (failure) occurs with probability 1-p

```
print("The distribution of the vehicles based on their transmission is ")
transmission = dsVeh$transmission
table(transmission)

# Dividing number of vehicles of a particular transmission by total number of the vehicles
# gives the individual probability distribution.
transmission_prob = table(transmission) / nrow(dsVeh)
transmission_prob

print(paste("Result shows that about ", transmission_prob[1] * 100 , "% are
            Automatic, while rest ", transmission_prob[2] * 100, "% are Manual"))

transmission_manual = transmission_prob[2]
```

```
> #3 - Transmission - Bernoulli Distribution.
>
> print("The distribution of the vehicles based on their transmission is ")
[1] "The distribution of the vehicles based on their transmission is "
> transmission = dsVeh$transmission
> table(transmission)
transmission
automatic    manual
    19239       761
>
> # Dividing number of vehicles of a particular transmission by total number of the vehicles
> # gives the individual probability distribution.
> transmission_prob = table(transmission) / nrow(dsVeh)
> transmission_prob
transmission
automatic    manual
  0.96195   0.03805
>
> print(paste("Result shows that about ", transmission_prob[1] * 100 , "% are
+             Automatic, while rest ", transmission_prob[2] * 100, "% are Manual"))
[1] "Result shows that about  96.195 % are \n            Automatic, while rest  3.805 % are Manual"
>
> transmission_manual = transmission_prob[2]
> barplot(transmission_prob, main = "Bernoulli Distribution - Transmission", xlab = "transmission", ylab = "Outc
es", ylim = c(0, 1))
> abline(h = transmission_manual, col = "blue")
```

```
359 barplot(transmission_prob, main = "Bernoulli Distribution - Transmission",
360         xlab = "transmission", ylab = "Outcomes", ylim = c(0, 1))
361 abline(h = transmission_manual, col = "blue")
```



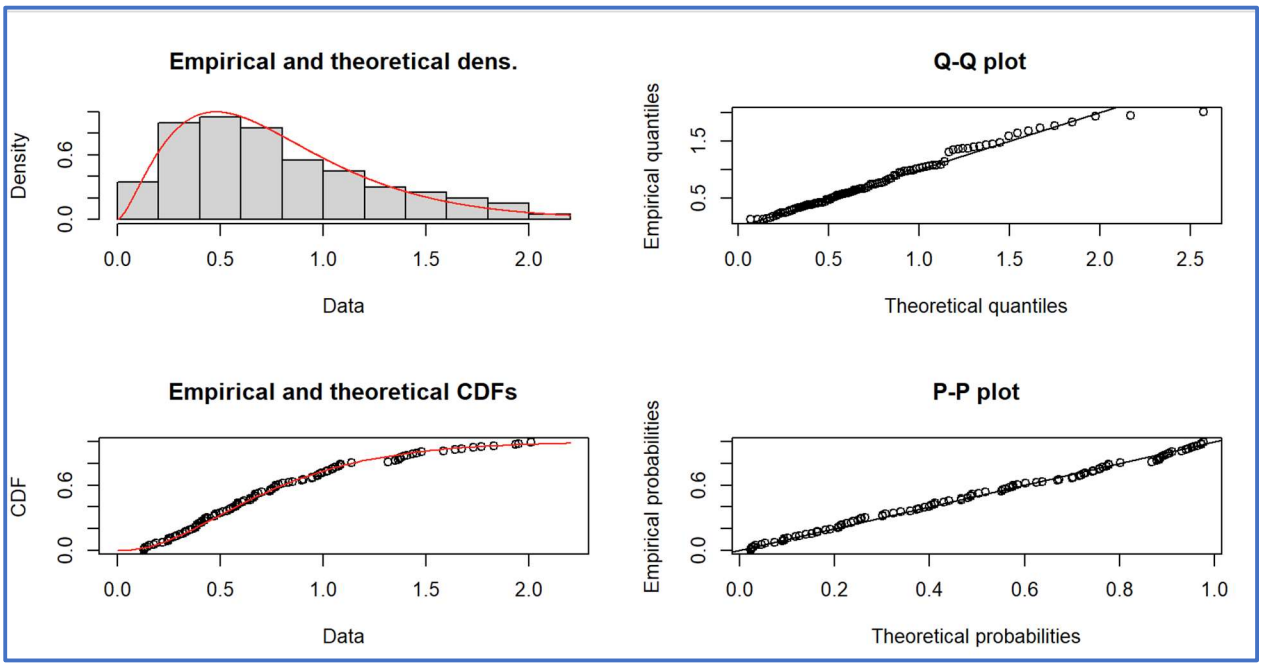
4. ***Odometer*** - Gamma Distribution.

The Gamma distribution is a particular case of the normal distribution, which have only positive results (to measure continuous variables that possess positive and skewed distributions).

```
#4 - Odometer - Gamma Distribution.

# Taking the numeric data from the data set for odometer readings.
dsVehOdometerNumeric = filter(dsVeh,odometer != '')
dsVehOdometer = dsVehOdometerNumeric$odometer

dsVehOdometerMean=mean(dsVehOdometer);
dsVehOdometerVar=var(dsVehOdometer)

print(paste("Mean = ",dsVehOdometerMean, "Variance = ",dsVehOdometerVar))

# Lambda is mean/variance
lambdaOdometer =dsVehOdometerMean/dsVehOdometerVar
# Alpha is lambda * mean
alphaOdometer=lambdaOdometer*dsVehOdometerMean

print(paste("Lambda = ",lambdaOdometer, "Alpha = ",alphaOdometer))

#Probability of getting odometer reading more than 65000 is P(>65000) = 1-P(X<65000)
print(paste("Probability of getting odometer reading more than 65000 is",
            1-pgamma(65000,alphaOdometer,lambdaOdometer)))
```

```
> #4 - Odometer - Gamma Distribution.
>
> # Taking the numeric data from the data set for odometer readings.
> dsVehOdometerNumeric = filter(dsVeh,odometer != '')
> dsVehOdometer = dsVehOdometerNumeric$odometer
>
> dsVehOdometerMean=mean(dsVehOdometer);
> dsVehOdometerVar=var(dsVehOdometer)
>
> print(paste("Mean = ",dsVehOdometerMean, "Variance = ",dsVehOdometerVar))
[1] "Mean =  77201.5045 Variance =  2734382022.10138"
>
> # Lambda is mean/variance
> lambdaOdometer =dsVehOdometerMean/dsVehOdometerVar
> # Alpha is lambda * mean
> alphaOdometer=lambdaOdometer*dsVehOdometerMean
>
> print(paste("Lambda = ",lambdaOdometer, "Alpha = ",alphaOdometer))
[1] "Lambda =  2.82336205680106e-05 Alpha =  2.17967798533256"
>
> #Probability of getting odometer reading more than 65000 is P(>65000) = 1-P(X<65000)
> print(paste("Probability of getting odometer reading more than 65000 is",
+             1-pgamma(65000,alphaOdometer,lambdaOdometer)))
[1] "Probability of getting odometer reading more than 65000 is 0.506491714175235"
```

```r
# The fitdistrplus package provides us fitdist function to fit a distribution
#fitdist(dataset, distr = "choice", method = "method")
#rgamma(s, α, λ) = s-sample, the parameter counter - rate, gamma- rate
#lambdaOdometer = 2.82, alpha = 2.17 sample size = 10

sampleGamma <- rgamma(100, 2.17, 2.82)

ans <- fitdist(sampleGamma, distr = "gamma", method = "mle")

summary(ans)
plot(ans)
```



## *Question 2 (c) - Express the way in which each model can be used for the predictive analytics, then find the prediction for each attribute.*

1. ***Normal Distribution*** - Prediction of MMR
   Using pnorm to get the vector of MMR variable pnorm value (area under the curve for the corresponding value) and maximum is chosen which gives us the MMR value with highest area under it.

```
401 #Normal Distribution - Prediction of MMR
402 dsVehMmrMean
403 dsVehMmrSd
404
405 dsVehMmrProbability = pnorm(dsVehMmr, mean = dsVehMmrMean, sd = dsVehMmrSd)
406 dsVehMmrProbabilityPrediction = dsVehMmr[which.max(dsVehMmrProbability)]
407
408 ## prediction
409 c("The predicted of mmr is", dsVehMmrProbabilityPrediction)
410
411
409:60  (Top Level)

R 4.4.1 · ~/
> #Normal Distribution - Prediction of MMR
> dsVehMmrMean
[1] 13106.34
> dsVehMmrSd
[1] 9638.279
>
> dsVehMmrProbability = pnorm(dsVehMmr, mean = dsVehMmrMean, sd = dsVehMmrSd)
> dsVehMmrProbabilityPrediction = dsVehMmr[which.max(dsVehMmrProbability)]
>
> ## prediction
> c("The predicted of mmr is", dsVehMmrProbabilityPrediction)
[1] "The predicted of mmr is" "149000"
```

2. ***Multinomial Distribution***

   Total number of states = 34 therefore N = 34. The states probability looks like

```
411 #Multinomial Distribution
412
413 # Total number of states  = 34 therefore n = 34
414 # The states probability looks like
415 probPerState
```

```
dsVehState
     ab      az      ca      co      fl      ga      hi      il      in      la      ma      md      mi      mn      mo      nc      ne
0.00190 0.02330 0.40665 0.00820 0.07365 0.03660 0.00070 0.04260 0.00900 0.00820 0.00255 0.00200 0.03485 0.01945 0.01445 0.02460 0.00935
     nj      nv      ny      oh      ok      on      or      pa      pr      qc      sc      tn      tx      ut      va      wa      wi
0.02285 0.01580 0.02240 0.01550 0.00035 0.00410 0.00050 0.07020 0.00815 0.00050 0.00815 0.01260 0.05605 0.00240 0.02855 0.00635 0.00750
```

   total sum of the values in the vector should equals N=34. let X is a vector displaying the frequency of each result. Assuming that we want to calculate probability of getting combination where 29 vehicles belongs to ca, 1 to fl,hi, pa, tx, va then x will be represented like below vector.

```
421 x=c(0,0,29,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,1,0,0)
422
423 n =34
424
425 p = dmultinom(x,n,probPerState)
426
427 print(paste("The predicted probality of above combination is :",p))
428
429 print(paste("The predicted State is:",dsVehStateMode))
```

```
> x=c(0,0,29,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,1,0,0)
>
> n =34
>
> p = dmultinom(x,n,probPerState)
>
> print(paste("The predicted probality of above combination is :",p))
[1] "The predicted probality of above combination is : 8.99120581488319e-13"
>
> print(paste("The predicted State is:",dsVehStateMode))
[1] "The predicted State is: ca"
```

## 3. *Bernoulli Distribution*

Probability distribution for transmission is

```
#Bernoulli Distribution

# probability distribution for transmission is
transmission_prob
```

```
> # probability distribution for transmission is
> transmission_prob
transmission
automatic    manual
  0.96195   0.03805
```

Out of two the one with the higher probability is picked up

```
# out of two the one with the higher probablity is picked up
trans = transmission_prob
transmission_pred = names(transmission_prob)[which.max(transmission_prob)]

c("The predicted transmission type  is", transmission_pred)
```

```
> # out of two the one with the higher probablity is picked up
> trans = transmission_prob
> transmission_pred = names(transmission_prob)[which.max(transmission_prob)]
>
> c("The predicted transmission type  is", transmission_pred)
[1] "The predicted transmission type  is" "automatic"
```

## 4. *Gamma Distribution*

10000 sample is generated through rgama for the alpha and lambda of the current data set Odometer values calculated in 2.b. The mean of the sample is the prediction value.

```
#Gamma Distribution

# 10000 sample is generated through rgama for the alpha and lambda of the current data set Odometer values
# calculated in 2.b. The mean of the sample is the prediction value.
rgammadsVehOdometerdata=rgamma(10000,alphaOdometer,lambdaOdometer)
predDsVehOdometerGamma=mean(rgammadsVehOdometerdata)
print(paste("The predicted odometer reading is", predDsVehOdometerGamma))
```

```
> #Gamma Distribution
>
> # 10000 sample is generated through rgama for the alpha and lambda of the current data set Odometer values
> # calculated in 2.b. The mean of the sample is the prediction value.
> rgammadsVehOdometerdata=rgamma(10000,alphaOdometer,lambdaOdometer)
> predDsVehOdometerGamma=mean(rgammadsVehOdometerdata)
> print(paste("The predicted odometer reading is", predDsVehOdometerGamma))
[1] "The predicted odometer reading is 76863.643164652"
```

## *Question 3 (a) -Consider two categorical variables of the dataset, develop a binary decision making strategy to check whether two variables are independent at the significant level alpha=0.01. To do so,*

### *i. State the hypotheses*
### *ii. Find the statistic and critical values.*
### *iii. Explain your decision and Interpret results.*

Since the two categorical variables are required to be checked for dependence, two-way or contingency tables method is employed. These tables will provide statistical inference based on data observed.

Furthermore, the chi-square test will be used as it provides the best method for testing the association between the row and column variables in a two-way table. The chi-square test requires calculation of the expected values based on the data. The expected value for each cell in a two-way table is equal to (row total*column total)/n, where n is the total number of observations included in the table.

A new categorical column (cond_rating) is added in the dataset based on the condition column of the dataset. Condition of the vehicle (condition column) is possibly rated on a scale. If you are reading a car review or attending an auction that uses the six-category system, it is easily translatable from the 100 point system (nationalmusclecars->Ratings Guide).

- Category 1 would be a 90 + point car
- Category 2 would be a 80 - 89 point car
- Category 3 would be a 70 - 79 point car
- Category 4 would be a 60 - 69 point car
- Category 5 would be a 40 - 59 point car
- Category 6 is any car under a 40 point car

#### *# Step 1: State the hypotheses*

The null hypothesis H0 assumes that there is no association between the variables (in other words, one variable does not vary according to the other variable)

H0: X1 (Category Ratings) and X2 (Transmission) are independent

H1: X1 (Category Ratings) and X2 (Transmission) are dependent

A function is created to provide category ratings based on the conditional ratings of the dataset.

```
# function to create Category ratings based on the conditional ratings of the dataset
condition_new_mult <- function(cond) {
  ifelse(cond >= 90, "Category 1",
         ifelse(cond >= 80 & cond <= 89, "Category 2",
                ifelse(cond >= 70 & cond <= 79, "Category 3",
                       ifelse(cond >= 60 & cond <= 69, "Category 4",
                              ifelse(cond >= 50 & cond <= 59, "Category 5",
                                     ifelse(cond >= 40 & cond <= 49, "Category 6",
                                            ifelse(cond >= 30 & cond <= 39, "Category 7",
                                                   ifelse(cond <= 20, "Category 7", "Category 7")))))))))
}
```

Filtering out the rows which has blank conditional ratings

```
# Filtering out the rows which has blank conditional ratings
dsVehRatings <- filter(dsVeh, condition != '')
```

New vector for categorical ratings is created by calling the function and New column is added in the dsVehRatings dataset. Dataset looks like.

```
dsVehRatings <- filter(dsVeh, condition != '')
# New vector for categorical ratings is created by calling the function
dsCol <-condition_new_mult(dsVehRatings$condition)
# New column is added in the dsVehRatings dataset
dsVehRatings$cond_rating <- dsCol

print("The data set with new column cond_rating")
head(dsVehRatings)
```

```
> # Filtering out the rows which has blank conditional ratings
> dsVehRatings <- filter(dsVeh, condition != '')
> dsCol <-condition_new_mult(dsVehRatings$condition)
> # New column is added in the dsVehRatings dataset
> dsVehRatings$cond_rating <- dsCol
>
> print("The data set with new column cond_rating")
[1] "The data set with new column cond_rating"
> head(dsVehRatings)
  year    make               model       trim  body transmission              vin state condition odometer color interior
1 2015     Kia             Sorento         LX   SUV    automatic 5xyktca69fg566472    ca         5    16639 white    black
2 2015     Kia             Sorento         LX   SUV    automatic 5xyktca69fg561319    ca         5     9393 white    beige
3 2014     BMW            3 Series 328i SULEV Sedan    automatic wba3c1c51ek116351    ca        45     1331  gray    black
4 2015   Volvo                 S60         T5 Sedan    automatic yv1612tb4f1310987    ca        41    14282 white    black
5 2014     BMW 6 Series Gran Coupe       650i Sedan    automatic wba6b2c57ed129731    ca        43     2641  gray    black
6 2015  Nissan              Altima      2.5 S Sedan    automatic 1n4al3ap1fn326013    ca         1     5554  gray    black
                                      seller   mmr sellingprice                      saledate cond_rating
1                         kia motors america  inc 20500        21500 Tue Dec 16 2014 12:30:00 GMT-0800 (PST)  Category 7
2                         kia motors america  inc 20800        21500 Tue Dec 16 2014 12:30:00 GMT-0800 (PST)  Category 7
3            financial services remarketing (lease) 31900        30000 Thu Jan 15 2015 04:30:00 GMT-0800 (PST)  Category 6
4                           volvo na rep/world omni 27500        27750 Thu Jan 29 2015 04:30:00 GMT-0800 (PST)  Category 6
5            financial services remarketing (lease) 66000        67000 Thu Dec 18 2014 12:30:00 GMT-0800 (PST)  Category 6
6 enterprise vehicle exchange / tra / rental / tulsa 15350        10900 Tue Dec 30 2014 12:00:00 GMT-0800 (PST)  Category 7
>
```

category Table is created using the cond_rating and transmission columns

```
#category Table is created using the cond_rating and transmission columns

X1=dsVehRatings$cond_rating
X2=dsVehRatings$transmission
C_table=table(X1,X2)
```

```
> X1=dsVehRatings$cond_rating
> X2=dsVehRatings$transmission
> C_table=table(X1,X2)
```

```
> C_table
            X2
X1           automatic manual
  Category 6      3679    132
  Category 7     11344    466
>
```

***Step 2: Setting significance level to 0.01.*** A matrix (2X2) is created

```
516  #Step 2: Setting significance level to 0.01. A matrix (2X2) is created
517  alpha = 0.01
518  E=matrix(NA,2,2)
519  N = nrow(dsVehRatings)
520  E
521
516:1  (Top Level)

R 4.4.1 · ~/
[2,]   NA   NA
> #Step 2: Setting significance level to 0.01. A matrix (2X2) is created
> alpha = 0.01
> E=matrix(NA,2,2)
> N = nrow(dsVehRatings)
> E
     [,1] [,2]
[1,]   NA   NA
[2,]   NA   NA
```

Calculation of the expected values based on the data. The expected value for each cell in a two-way table is equal to (row total*column total)/n, where n is the total number of observations included in the table. Calculation of the expected values based on the data. The expected value for each cell

```
522  #Calculation of the expected values based on the data. The expected value for each cell
523  # in a two-way table is equal to (row total*column total)/n, where n is the total number
524  # of observations included in the table.
525  X1=2;X2=2
526  for(i in 1:X1){
527    for(j in 1:X2){
528      Ci=sum(C_table[i,]);
529      Cj=sum(C_table[,j])
530      E[i,j]=(Ci*Cj)/N
531    }
532  }
533
534  print("The original relation table for ratings and transmission is")
535  C_table
536  print("The Expected relation table for ratings and transmission is")
537  E
```

```
> #Calculation of the expected values based on the data. The expected value for each cell
> # in a two-way table is equal to (row total*column total)/n, where n is the total number
> # of observations included in the table.
> X1=2;X2=2
> for(i in 1:X1){
+   for(j in 1:X2){
+     Ci=sum(C_table[i,]);
+     Cj=sum(C_table[,j])
+     E[i,j]=(Ci*Cj)/N
+   }
+ }
>
> print("The original relation table for ratings and transmission is")
[1] "The original relation table for ratings and transmission is"
> C_table
            X2
X1           automatic manual
  Category 6      3679    132
  Category 7     11344    466
> print("The Expected relation table for ratings and transmission is")
[1] "The Expected relation table for ratings and transmission is"
> E
          [,1]     [,2]
[1,]  3665.108 145.8919
[2,] 11357.892 452.1081
```

***Step 3: Test Value:*** *Once the expected values have been computed the chi-square test statistics*

is computed as

X**2 = SUM(observed - Expected)POWER2/Expected

where the square of the differences between the observed and expected values in each cell,

divided by the expected value, are added across all of the cells in the table.

The distribution of statistic X2 is chi-square with (row-1)(col-1) degrees of freedom,

where row represents the number of rows in the two-way table and col represents the number of columns.

The distribution is denoted (df), where df is the number of degrees of freedom.

```
539 # Step 3: Test Value : Once the expected values have been computed the chi-square test statistic
540 #is computed as
541 #
542 # X**2 = SUM(observed - Expected)POWER2/Expected
543 #
544 # where the square of the differences between the observed and expected values in each cell,
545 # divided by the expected value, are added across all of the cells in the table.
546 # The distribution of the statistic X2 is chi-square with (r-1)(c-1) degrees of freedom,
547 # where r represents the number of rows in the two-way table and c represents the number of columns.
548 # The distribution is denoted (df), where df is the number of degrees of freedom.
549
550 test.value=sum((C_table-E)^2/E)
551 test.value
```

```
> # Step 3: Test Value : Once the expected values have been computed the chi-square test statistic
> #is computed as
> #
> # X**2 = SUM(observed - Expected)POWER2/Expected
> #
> # where the square of the differences between the observed and expected values in each cell,
> # divided by the expected value, are added across all of the cells in the table.
> # The distribution of the statistic X2 is chi-square with (r-1)(c-1) degrees of freedom,
> # where r represents the number of rows in the two-way table and c represents the number of columns.
> # The distribution is denoted (df), where df is the number of degrees of freedom.
>
> test.value=sum((C_table-E)^2/E)
> test.value
[1] 1.819305
```

## *Step 4: C Value*:

```
553  #
554  # Step 4: C Value: The chi-square test is based on a test statistic that measures the divergence of
555  # the observed data from the values that would be expected under the null hypothesis of no association.
556  # This requires calculation of the expected values based on the data. The expected value for each cell
557  # -way table is equal to (row total*column total)/n, where n is the total number of observations
558  # included in the table.
559
560  df = 1
561  c.value = qchisq(1-alpha,df)
562  c.value
563
564  #Step 5: Decision Rule
562:8  (Top Level)

R 4.4.1 . ~/
> # Step 4: C Value: The chi-square test is based on a test statistic that measures the divergence of
> # the observed data from the values that would be expected under the null hypothesis of no association.
> # This requires calculation of the expected values based on the data. The expected value for each cell
> # -way table is equal to (row total*column total)/n, where n is the total number of observations
> # included in the table.
>
> df = 1
> c.value = qchisq(1-alpha,df)
> c.value
[1] 6.634897
```

## *Step 5: Decision Rule*

```
564  #Step 5: Decision Rule
565
566  ifelse(test.value < c.value, "H0 is accepted: ie, X1 and X2 are independent", "H0 is rejected: ie, X1 and X2 are dependent")
567
568
566:125  (Top Level)

R 4.4.1 . ~/
> #Step 5: Decision Rule
>
> ifelse(test.value < c.value, "H0 is accepted: ie, X1 and X2 are independent", "H0 is rejected: ie, X1 and X2 are dependent")
[1] "H0 is accepted: ie, X1 and X2 are independent"
```

## *Step 6: Make a decision and conclusion*

```
568
569  #Step 6: Make a decision and conclusion
570
571  #H0 is accepted: ie, Category Ratings and Transmission are independent. Here the null hypothesis (H0) is satisfied,
572  #therefore two categorical variables Conditional Rating and Transmission are independent at the significance level α = 0.01
573
574
573:1  (Top Level)

R 4.4.1 . ~/
> #Step 6: Make a decision and conclusion
>
> #H0 is accepted: ie, Category Ratings and Transmission are independent. Here the null hypothesis (H0) is satisfied,
> #therefore two categorical variables Conditional Rating and Transmission are independent at the significance level α = 0.01
```

***Summary***

- Considered the two independent variables (conditional ratings and transmission) and conducted chi-square test to for independence between the two with significance level of 0.01.
- *test value came out to be 1.819 and the c value came out to be 6.634 using chi-square.*
- *Null Hypothesis (i.e the variables are independent) was accepted as test value is less than c value.*
- *From the above test it is concluded that* H0 is accepted: i.e. the two categorical variables Conditional Rating and Transmission are independent at the significance level $\alpha = 0.01$

## *3 b) Consider one categorical variable, apply goodness of fit test to evaluate whether a candidate set of probabilities can be appropriate to quantify the uncertainty of class frequency at the significant level alpha=0.05.*

Considering Transmission as the categorical vehicle for this test

### *Step 1: State the hypotheses*

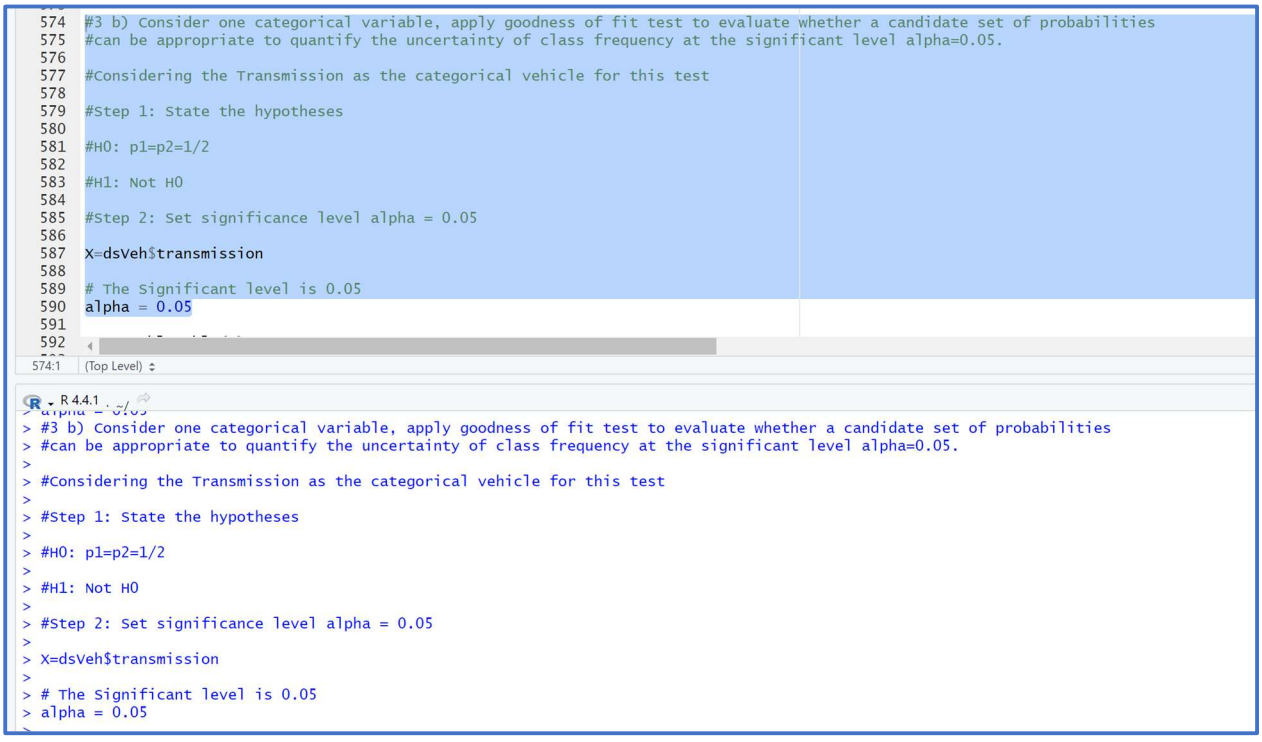
H0: The observed set of frequencies of Transmission align with the candidate set of frequencies.

H1: The observed set of frequencies of Transmission do not align with the candidate set of frequencies.

Candidate probabilities: Let's assume equal probability distribution(p1=p2=1/2) of transmission.

Since N=20000, the expected probability distribution table for two transmission should be [10000,10000]

### *Step 2: Set significance level alpha* = 0.05



```
574  #3 b) Consider one categorical variable, apply goodness of fit test to evaluate whether a candidate set of probabilities
575  #can be appropriate to quantify the uncertainty of class frequency at the significant level alpha=0.05.
576
577  #Considering the Transmission as the categorical vehicle for this test
578
579  #Step 1: State the hypotheses
580
581  #H0: p1=p2=1/2
582
583  #H1: Not H0
584
585  #Step 2: Set significance level alpha = 0.05
586
587  X=dsVeh$transmission
588
589  # The Significant level is 0.05
590  alpha = 0.05
591
592
574:1  (Top Level)

R 4.4.1
> #3 b) Consider one categorical variable, apply goodness of fit test to evaluate whether a candidate set of probabilities
> #can be appropriate to quantify the uncertainty of class frequency at the significant level alpha=0.05.
>
> #Considering the Transmission as the categorical vehicle for this test
>
> #Step 1: State the hypotheses
>
> #H0: p1=p2=1/2
>
> #H1: Not H0
>
> #Step 2: Set significance level alpha = 0.05
>
> X=dsVeh$transmission
>
> # The Significant level is 0.05
> alpha = 0.05
```

```
print("Original Transmission Table")
TransTabl

ProbabilityTransTable =TransTabl/sum(TransTabl)

print("Probability of the original Transmission Table")
ProbabilityTransTable

P0=rep(1/2,2)

print("Probability table 50 percent")

P0

N=nrow(dsVeh)

print(paste("Expected Probability table 50 percent= Prob * Total Number = ", N))
```

```
[1] "Original Transmission Table"
> TransTabl
X
automatic    manual
    19239       761
>
> ProbabilityTransTable =TransTabl/sum(TransTabl)
>
> print("Probability of the original Transmission Table")
[1] "Probability of the original Transmission Table"
> ProbabilityTransTable
X
automatic    manual
  0.96195   0.03805
>
> P0=rep(1/2,2)
>
> print("Probability table 50 percent")
[1] "Probability table 50 percent"
```

The test value is calculated using chi-square.

```
> N=nrow(dsVeh)
>
> print(paste("Expected Probability table 50 percent= Prob * Total Number = ", N))
[1] "Expected Probability table 50 percent= Prob * Total Number =  20000"
```

## *Step 3: Find the test.value*

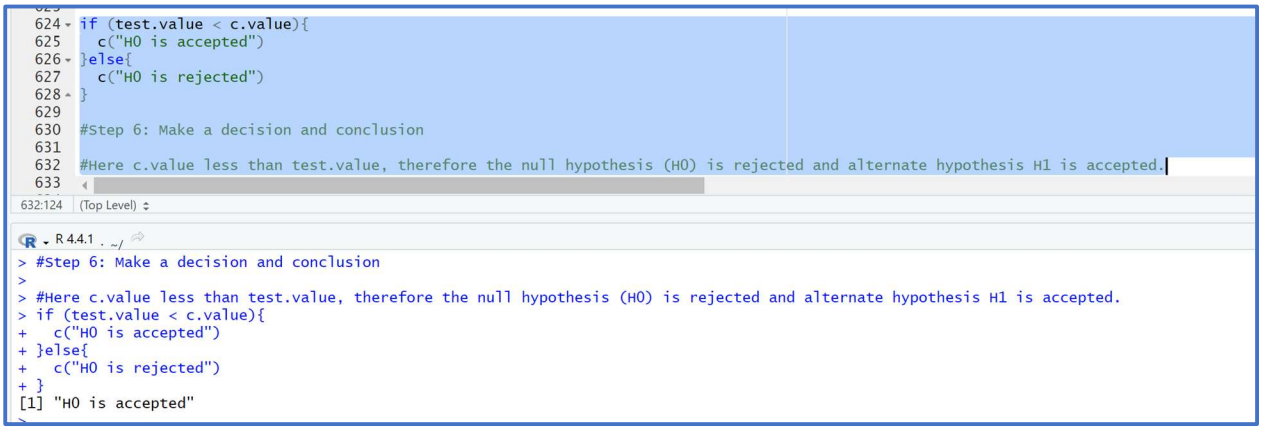
```
ExpectedProbTbl=N*P0

ExpectedProbTbl

test.value=sum((TransTabl-ExpectedProbTbl)^2/ExpectedProbTbl)
test.value
```

```
> ExpectedProbTbl=N*P0
>
> ExpectedProbTbl
[1] 10000 10000
>
> test.value=sum((TransTabl-ExpectedProbTbl)^2/ExpectedProbTbl)
> test.value
[1] 17071.82
```

### *Step 4: Find the c.value*

```
> #Step 4: Find the c.value
> df = N-1
> c.value = qchisq(1-alpha, df)
> c.value
[1] 20329.1
>
```

### *Step 5: Make a decision and conclusion*



```
624 if (test.value < c.value){
625   c("H0 is accepted")
626 }else{
627   c("H0 is rejected")
628 }
629
630 #Step 6: Make a decision and conclusion
631
632 #Here c.value less than test.value, therefore the null hypothesis (H0) is rejected and alternate hypothesis H1 is accepted.
633
632:124  (Top Level)

R 4.4.1 · ~/
> #Step 6: Make a decision and conclusion
>
> #Here c.value less than test.value, therefore the null hypothesis (H0) is rejected and alternate hypothesis H1 is accepted.
> if (test.value < c.value){
+   c("H0 is accepted")
+ }else{
+   c("H0 is rejected")
+ }
[1] "H0 is accepted"
>
```

### *Summary*

- Applying a goodness of fit test for one categorical variable (Tranmission) against a set of expected probabilities (50-50% for two values of transmission).
- The test value is calculated using chi-square is 17071.82
- C Value using chi-square is 20329.1 for significant level 0.05
- c.value less than test.value, therefore the null hypothesis (H0) is rejected and alternate hypothesis H1 is accepted
- It can be concluded that the null hypothesis that probability distribution of 50-50% was not true.

## *Question 3-(c) Consider one continuous variable in the data set, and apply test of mean for a proposed candidate of μ at the significant level alpha=0.05.*

### *Step 1: State the hypotheses*

Considering the selling price column as the continues column for this test.

Selling price of the vehicle in population will be < 11000.00

Let Mu0 be 11000

H0: Poulation Mean < 11000

H1: Poulation Mean >= 11000

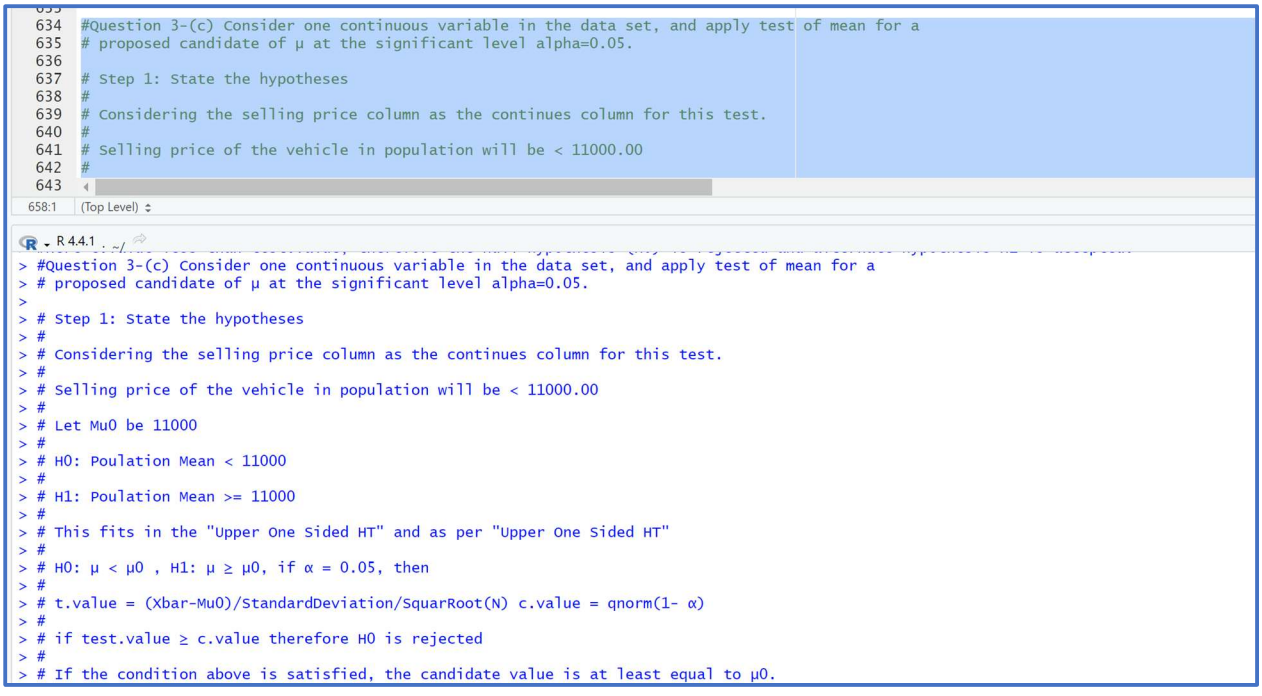
This fits in the "Upper One Sided HT" and as per "Upper One Sided HT"

H0: μ < μ0 , H1: μ ≥ μ0, if α = 0.05, then

t.value = (Xbar-Mu0)/StandardDeviation/SquarRoot(N) c.value = qnorm(1- α)

if test.value ≥ c.value therefore H0 is rejected

If the condition above is satisfied, the candidate value is at least equal to μ0.



### *Step 2: Setting Significance value*

```
660 #Step 2: Setting Significance value
661
662 alpha = 0.05
663 print(paste("Significance is set to : ", alpha))
664
665
```

```
[1] "Significance is set to :  0.05"
> #Step 2: Setting Significance value
>
> alpha = 0.05
```

```
665 Mu0 = 11000
666 X=dsVeh$sellingprice
667 X_bar=mean(X)
668 SD=sd(X)
669 N = length(X)
670
671 print(paste("Sample Mean: ", X_bar,"Standard Deviation: ", SD,"Sample Size: ", N))
```

```
> Mu0 = 11000
> X=dsVeh$sellingprice
> X_bar=mean(X)
> SD=sd(X)
> N = length(X)
>
> print(paste("Sample Mean: ", X_bar,"Standard Deviation: ", SD,"Sample Size: ", N))
[1] "Sample Mean:  12712.64805 Standard Deviation:  9647.54816769337 Sample Size:  20000"
```

### *Step 3: Compute the test.value*

```
673 #Step 3: Compute the test.value
674
675 test.value=(X_bar-Mu0)/(SD/sqrt(N))
676 print(paste("Test Value = : ",test.value))
677
```

### *Step 4: Find the c.value*

```
678 #Step 4: Find the c.value
679
680 c.value = qnorm(1-alpha)
681 print(paste("C Value = : ",c.value))
```

### *Step 5: Specify the decision rule*

If test.value ≥ c.value therefore H0 is rejected

```
683  # Step 5: Specify the decision rule
684  #
685  # If test.value ≥ c.value therefore H0 is rejected
686
687  evaluate_results <- function(tval, cval)
688  {
689    if (tval < cval)
690    {
691      print("H0 is accepted")
692    }
693    else
694    {
695      print("H0 is rejected")
696    }
697  }
698
699  evaluate_results(test.value,c.value)
700
701
699:37  (Top Level)
```

```
R 4.4.1 . ~/
> # Step 5: Specify the decision rule
> #
> # If test.value ≥ c.value therefore H0 is rejected
>
> evaluate_results <- function(tval, cval)
+ {
+   if (tval < cval)
+   {
+     print("H0 is accepted")
+   }
+   else
+   {
+     print("H0 is rejected")
+   }
+ }
>
> evaluate_results(test.value,c.value)
[1] "H0 is rejected"
```

## *Step 6: Make a decision and conclusion*

If the condition in step 5 is satisfied, the population mean is at least equal to Mu0. Here test.values is greater than c.value, therefore the null hypothesis (H0) is rejected and alternate hypothesis H1 is accepted. That means the population mean would be at least 11000.

```
701  # Step 6: Make a decision and conclusion
702  #
703  # If the condition in step 5 is satisfied, the population mean is atleast equal to Mu0.
704  #
705  # Here test.value greater than c.value, therefore the null hypothesis (H0) is rejected and alternate hypothesis H1 is accepted.
706  # That means the population mean would be at least 11000.
707
708  ### End of the Assignment
709
```

```
> ### End of the Assignment
> # Step 6: Make a decision and conclusion
> #
> # If the condition in step 5 is satisfied, the population mean is atleast equal to Mu0.
> #
> # Here test.value greater than c.value, therefore the null hypothesis (H0) is rejected and alternate hypothesis H1 is accepted.
> # That means the population mean would be at least 11000.
>
> ### End of the Assignment
```

***Summary***

- *Considered selling price column as the continues column for this test.*
- *Considered a proposed mean candidate value(11000).*
- *Null Hypothesis is population mean < candidate value.*
- *Alternate Hypothesis is population mean >= candidate value*
- *Data fitted in the Upper One-Sided HT formula.*
- *test value came out to be 25.105 and the c value came out to be 1.644 using test of mean with significance level of 0.05.*
- *Here test.value is greater than value, therefore the null hypothesis (H0) is rejected and alternate hypothesis H1 is accepted. That means the population mean would be at least 11000.*

*References*


- "[1] An Introduction to Statistical Methods and Data Analysis 7th Edition."
- "[2] Wickham, *R for Data Science : Import, Tidy, Transform, Visualize, and Model Data*."
- *https://study.dbs.ie/2122/msc-data/B9DA101/u6/index.html#/*
- *https://cran.r-project.org/web/packages/fitdistrplus/index.html*
- *https://www.r-bloggers.com/2022/09/the-multinomial-distribution-in-r/*
- *https://www.scaler.com/topics/r-rnorm/*
- *https://www.acsu.buffalo.edu/~adamcunn/probability/gamma.html#:~:text=The%20waiting%20time%20until%20%CE%B1,%22spread%22%20of%20the%20distribution*.