UNIVERSITÀ
degli STUDI
di CATANIA

DATA ANALYSIS OF ELECTRIC VEHICLE DATASET

Data Analysis and Statistical Learning
Professor Antonio Punzo

AsAD ASLAM – Data analysis on electric vehicles

Matriculation Number: 1000053142
Enrollment Year: 2022/2023
Data Science – University of Catania

Contents

1. Introduction.

In this report, I describe the analysis I performed on the electric vehicles dataset. I have divided this report into 3 parts. The first part starts with a description of the dataset, and then I continue with the univariate analysis where I take generally about the characteristics of all the variables and analyse them in detail. In the second part, I described the principal component analysis. In the last part, I discuss the cluster analysis I performed, starting with the evaluation of cluster tendency and the determination of the optimal number of clusters. Then I explain how to perform hierarchical clustering and partitioning clustering, and devote the last part of each section to cluster validation.

2. Preliminary Analysis.

2.1. Dataset Description.

There are some datasets already installed in R. In this report, I am using an external dataset of electric vehicles. This dataset is available on Kaggle (https://www.kaggle.com/datasets/geoffnel/evs-one-electric-vehicle-dataset). While using any external dataset we have to use the ***read()*** function to load the dataset file.

```
{r}
vehicle = read.csv(file='Electric_Vehicles.csv')
```

Then I used the ***str()*** command by which we can look at the structure of the data.

```
{r}
str(vehicle)

 'data.frame':   98 obs. of  8 variables:
  $ Brand          : chr  "Tesla " "Volkswagen " "Polestar " "BMW " ...
  $ Model          : chr  "Model 3 Long Range Dual Motor" "ID.3 Pure" "2" "iX3 "
 ...
  $ AccelSec       : num  4.6 10 4.7 6.8 9.5 2.8 9.6 8.1 5.6 6.3 ...
  $ TopSpeed_KmH   : int  233 160 210 180 145 250 150 150 225 180 ...
  $ Range_Km       : int  450 270 400 360 170 610 190 275 310 400 ...
  $ Efficiency_WhKm: int  161 167 181 206 168 180 168 164 153 193 ...
  $ FastCharge_KmH : int  940 250 620 560 190 620 220 420 650 540 ...
  $ PriceEuro      : int  55480 30000 56440 68040 32997 105000 31900 29682 46380
 55000 ...
```

As we can see from the above, the data contains 98 observations and 8 variables. Each observation refers to an electric vehicle and provides information regarding it.

Brand: Company name of a car.

Model: Name of a card.

AccelSec: Numeric value shows the acceleration per second.

TopSpeed_KmH: Numeric value shows the top speed of a car in kilometres per hour.

Range_Km: Numeric value shows the total range speed of a car in kilometres.

Efficiency_whKm: Numeric value shows the efficiency of a car in kilowatts per hour.

FastCharge_KmH: Numeric value shows the fast charging capacity in kilometres per hour.

PriceEuro: Numeric Value shows the price of a car.

2.2. Univariate Analysis.

After executing the ***str()*** I can see that all variables are numerical except for *brand* and *model*. So, let's use the function to look at some values of the dataset. I am using a ***head()*** function which provides us with the top 6 rows of data.

```
head(vehicle)
```

```
##          Brand                        Model AccelSec TopSpeed_KmH Range_Km
## 1        Tesla  Model 3 Long Range Dual Motor      4.6          233      450
## 2 Volkswagen                      ID.3 Pure     10.0          160      270
## 3     Polestar                             2      4.7          210      400
## 4          BMW                           iX3      6.8          180      360
## 5        Honda                             e      9.5          145      170
## 6        Lucid                           Air      2.8          250      610
##   Efficiency_WhKm FastCharge_KmH PriceEuro
## 1             161            940     55480
## 2             167            250     30000
## 3             181            620     56440
## 4             206            560     68040
## 5             168            190     32997
## 6             180            620    105000
```

To check you have loaded data correctly, we can use a ***tail()*** function which provides us with the last 6 rows of the data. You can verify the number of the last row with the number of observations 98 we got previously in ***str().***

```
tail(vehicle)
```

```
##       Brand                          Model AccelSec TopSpeed_KmH Range_Km
## 93   Byton               M-Byte 72 kWh 2WD      7.5          190      325
## 94 Nissan                     Ariya 63kWh      7.5          160      330
## 95    Audi    e-tron S Sportback 55 quattro      4.5          210      335
## 96 Nissan             Ariya e-4ORCE 63kWh      5.9          200      325
## 97 Nissan  Ariya e-4ORCE 87kWh Performance      5.1          200      375
## 98   Byton               M-Byte 95 kWh 2WD      7.5          190      400
##    Efficiency_WhKm FastCharge_KmH PriceEuro
## 93             222            420     53500
## 94             191            440     45000
## 95             258            540     96050
## 96             194            440     50000
## 97             232            450     65000
## 98             238            480     62000
```

The number of observations is equal to the number of the last row in a dataset. So moving forward, I am assuming that there are no negative values in the dataset and that 6 variables are continuous. To check this, I applied a ***summary()*** method.

```
summary(vehicle)
```

```
##     Brand              Model              AccelSec       TopSpeed_KmH
##  Length:98          Length:98          Min.   : 2.100   Min.   :123.0
##  Class :character   Class :character   1st Qu.: 5.100   1st Qu.:150.0
##  Mode  :character   Mode  :character   Median : 7.300   Median :167.0
##                                        Mean   : 7.047   Mean   :181.7
##                                        3rd Qu.: 8.950   3rd Qu.:200.0
##                                        Max.   :14.000   Max.   :410.0
##     Range_Km     Efficiency_WhKm FastCharge_KmH    PriceEuro
##  Min.   :170.0   Min.   :104.0   Min.   :170.0   Min.   : 20129
##  1st Qu.:258.8   1st Qu.:168.0   1st Qu.:275.0   1st Qu.: 35000
##  Median :350.0   Median :181.0   Median :440.0   Median : 45000
##  Mean   :350.2   Mean   :189.9   Mean   :456.7   Mean   : 57325
##  3rd Qu.:407.5   3rd Qu.:206.0   3rd Qu.:560.0   3rd Qu.: 65465
##  Max.   :970.0   Max.   :273.0   Max.   :940.0   Max.   :215000
```

The summary function supports my assumptions where brand and model are character variables and others are continuous variables defined in [0, infinte].

Let’s discuss each variable in depth.

2.2.1. Brand.

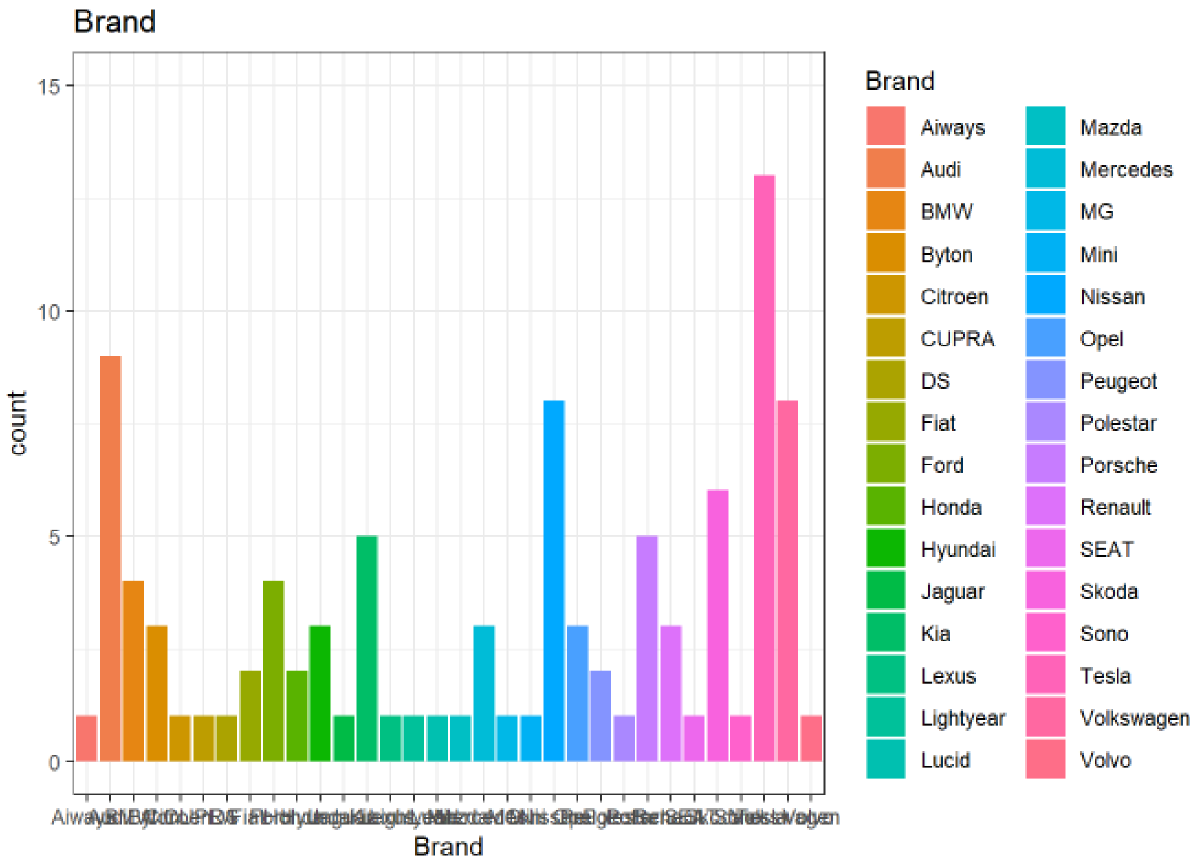
Model is the character variable which describes the company or maker name of an electric vehicle.

```
vehicle$Brand <- as.factor(vehicle$Brand)
summary(vehicle$Brand)
```

```
##     Aiways       Audi        BMW      Byton    Citroen      CUPRA
##          1          9          4          3          1          1
##         DS       Fiat       Ford      Honda    Hyundai     Jaguar
##          1          2          4          2          3          1
##        Kia      Lexus  Lightyear      Lucid      Mazda   Mercedes
##          5          1          1          1          1          3
##         MG       Mini     Nissan       Opel    Peugeot   Polestar
##          1          1          8          3          2          1
##    Porsche    Renault       SEAT      Skoda       Sono      Tesla
##          5          3          1          6          1         13
## Volkswagen      Volvo
##          8          1
```

To visualize the character data in the histogram to see the counts of data of any specific brand, we need to use ***as.factor()*** function.

```
Brand_Hist <-  ggplot(vehicle, aes(x=Brand, fill=Brand)) +
  theme_bw() +
  geom_bar() +
  ylim(0, 15) +
  labs(title = "Brand") +
  scale_x_discrete()
Brand_Hist
```



The above graph shows the counts of the electric cars of different brands. According to the graph, the most number of electric cars 13 are manufactured by Tesla. On the other hand, the least number of a car by any company is 1.

2.2.2. Model.

This variable contains all values different from each other because a company manufactures many different types of car and they differentiate it with the model name or a number. This dataset contains both alphanumeric. In this case, we can't plot any specific graph to visualize this data.

```
{r}
summary(vehicle$Model)
```

```
   Length     Class      Mode 
       98 character character 
```

2.2.3. AccelSec.

This is a numeric continuous variable, measured in acceleration per second. The basic statistical values are mentioned below.

```
{r}
summary(vehicle$AccelSec)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  2.100   5.100   7.300   7.047   8.950  14.000 
```

```
{r}
sd(vehicle$AccelSec)
```

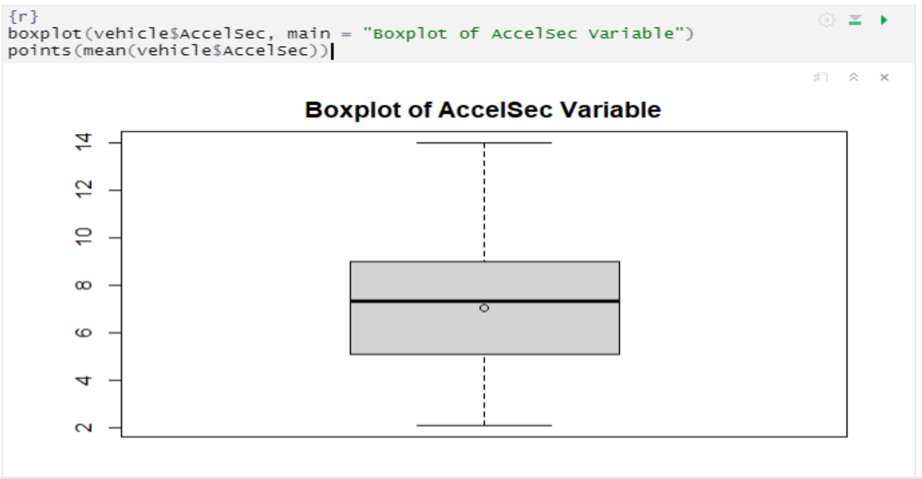
```
[1] 2.483895
```

```
{r}
var(vehicle$AccelSec)
```

```
[1] 6.169733
```

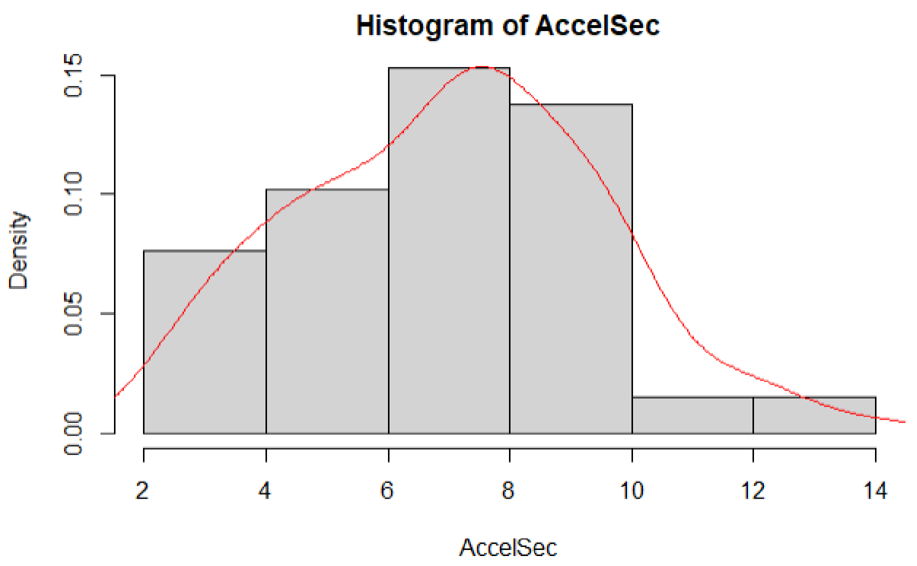
As we can see the median is greater than the mean, hence the distribution should be negatively skewed.

The boxplot of this variable is:



Boxplot gives us graphical information regarding the data and tells us how our data is well distributed in a dataset. The median divides the box into two parts. If we plot a box plot for our data the following plot appears. While reviewing the box plot we can observe that there are no outliers in this variable as we didn't see any dot lie outside the whiskers of the boxplot and it was also more clarified that the mean is below the median, outliers are the data points located outside the whiskers of the boxplot. Now we can plot the histogram along with skewness and kurtosis to look for the model that fits better the distribution:

```
{r}
hist(vehicle$AccelSec, freq=FALSE, main = "Histogram of AccelSec", xlab = "
AccelSec")
lines(density(vehicle$AccelSec), col="red")
```




```
{r}
skewness(vehicle$AccelSec)
```

```
[1] 0.1485048
```

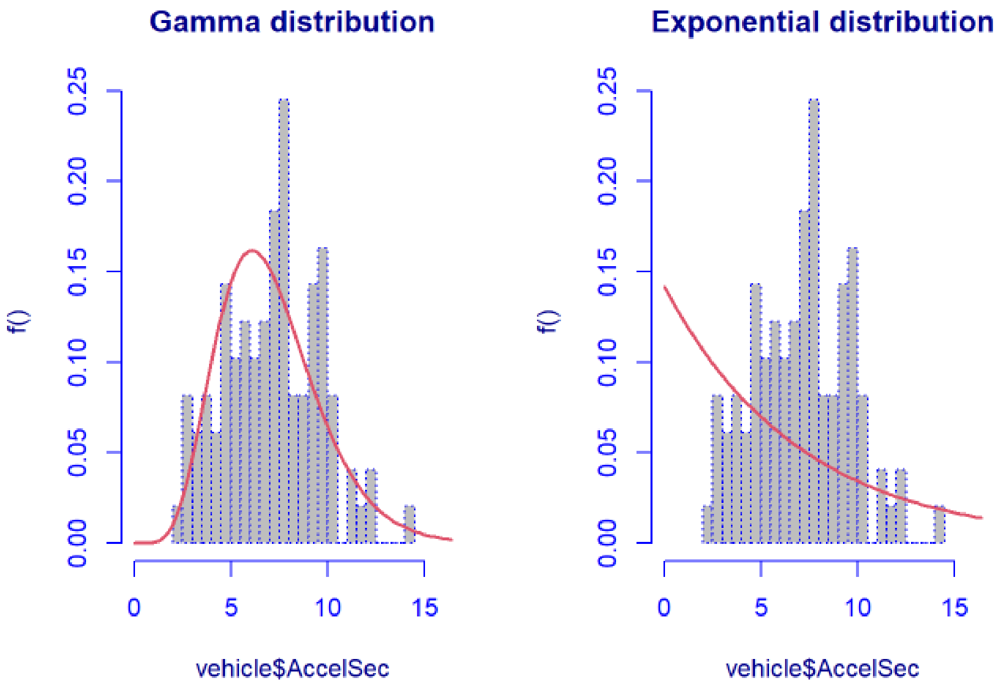
```
{r}
kurtosis(vehicle$AccelSec)
```

```
[1] 2.679723
```

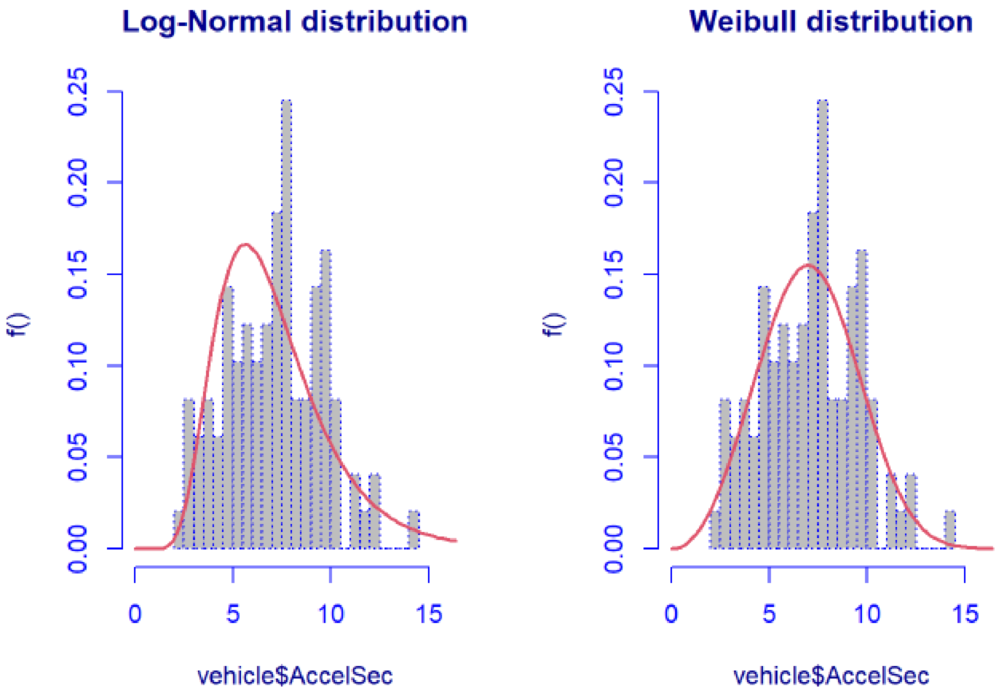
As we can see from the histogram it looks like a normal distribution but it is not because the skewness is not equal to exactly zero and according to the histogram, our distribution is positively-skewed (0.1485048). Hence our above prediction was wrong and kurtosis (2.679723) indicates that the distribution has less extreme outliers than a normal distribution.

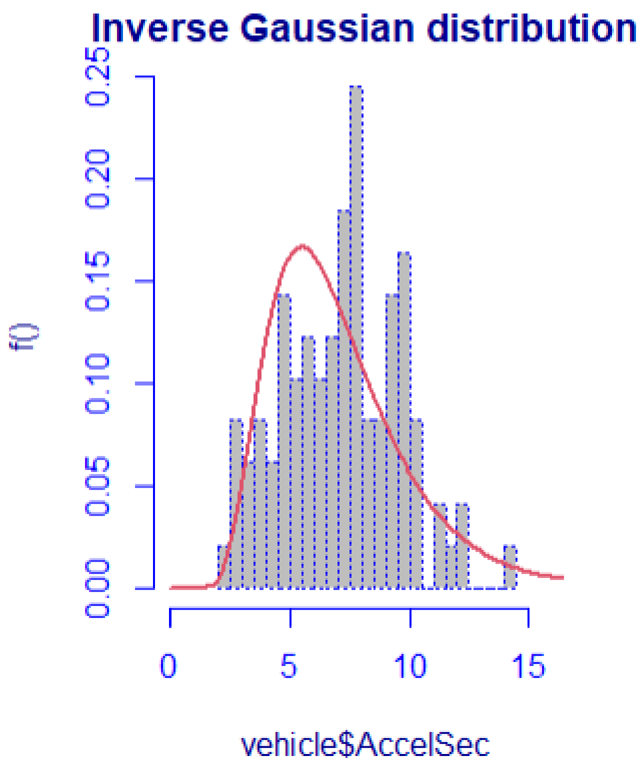
To fit this distribution, we are going to include in the comparison models which deal better with skewness:

```
par(mfrow=c(1,2))
fit.gamma <- histDist(vehicle$AccelSec, family=GA, nbins=30, main="Gamma distribution")
fit.EXP <- histDist(vehicle$AccelSec, family=EXP, nbins=30, main="Exponential distribution")
```




```
fit.LOGNO <- histDist(vehicle$AccelSec, family=LOGNO, nbins=30, main="Log-Normal distribution")
fit.WEI <- histDist(vehicle$AccelSec, family=WEI, nbins=30, main="Weibull distribution")
```






```r
data.frame(row.names=c("Gamma", "Exponentional", "Log-Normal", "Weibull","Inverse Gaussian"),
 DF=c(fit.gamma$df.fit, fit.EXP$df.fit, fit.LOGNO$df.fit, fit.WEI$df.fit, fit.IG$df.fit),
 LOGLIK=c(logLik(fit.gamma),logLik(fit.EXP),logLik(fit.LOGNO),logLik(fit.WEI),logLik(fit.IG)),
 AIC=c(AIC(fit.gamma), AIC(fit.EXP), AIC(fit.LOGNO),AIC(fit.WEI), AIC(fit.IG)),
 BIC=c(fit.gamma$sbc, fit.EXP$sbc, fit.LOGNO$sbc, fit.WEI$sbc, fit.IG$sbc))
```

```
##                  DF    LOGLIK      AIC      BIC
## Gamma             2 -228.8409 461.6818 466.8518
## Exponentional     1 -289.3541 580.7083 583.2933
## Log-Normal        2 -232.4413 468.8825 474.0525
## Weibull           2 -226.6509 457.3017 462.4717
## Inverse Gaussian  2 -232.8252 469.6503 474.8203
```

As we can see from the above chart with Likelihood, AIC and BIC, the model Weibull fits better the distribution for AccelSec.

The mixture of distribution:

It is possible to compute a mixture of two distributions to find the best mixture, the algorithm is repeated five times.

```
{r}
par(mfrow=c(1,3))
fit.GA.AcceSec <- gamlssMXfits(n = 5, vehicle$AccelSec~1, family = GA, K = 2, data
= vehicle)
par(mfrow=c(1,3))
fit.EXP.AcceSec <- gamlssMXfits(n = 5, vehicle$AccelSec~1, family = EXP, K = 2,
data = vehicle)
par(mfrow=c(1,3))
fit.LOGNO.AcceSec <- gamlssMXfits(n = 5, vehicle$AccelSec~1, family = LOGNO, K = 2,
data = vehicle)
par(mfrow=c(1,3))
fit.WEI.AcceSec <- gamlssMXfits(n = 5, vehicle$AccelSec~1, family = WEI, K = 2,
data = vehicle)
par(mfrow=c(1,3))
fit.IG.AcceSec <- gamlssMXfits(n = 5, vehicle$AccelSec~1, family = IG, K = 2, data
= vehicle)
```

```
data.frame(row.names=c("Gamma", "Exponentional", "Log-Normal", "Weibull","Inverse Gaussian","M.Gamma", "M.Exponentional",
"M.Log-Normal", "M.Weibull","M.Inverse Gaussian"),

 DF=c(fit.gamma$df.fit, fit.EXP$df.fit, fit.LOGNO$df.fit, fit.WEI$df.fit, fit.IG$df.fit, fit.GA.AcceSec$df.fit, fit.EXP.Acce
Sec$df.fit, fit.LOGNO.AcceSec$df.fit, fit.WEI.AcceSec$df.fit, fit.IG.AcceSec$df.fit),

 LOGLIK=c(logLik(fit.gamma),logLik(fit.EXP),logLik(fit.LOGNO),logLik(fit.WEI),logLik(fit.IG),logLik(fit.GA.AcceSec),logLik(f
it.EXP.AcceSec),logLik(fit.LOGNO.AcceSec),logLik(fit.WEI.AcceSec),logLik(fit.IG.AcceSec)),

 AIC=c(AIC(fit.gamma), AIC(fit.EXP), AIC(fit.LOGNO),AIC(fit.WEI), AIC(fit.IG),AIC(fit.GA.AcceSec), AIC(fit.EXP.AcceSec), AIC
(fit.LOGNO.AcceSec),AIC(fit.WEI.AcceSec), AIC(fit.IG.AcceSec)),

 BIC=c(fit.gamma$sbc, fit.EXP$sbc, fit.LOGNO$sbc, fit.WEI$sbc, fit.IG$sbc,fit.GA.AcceSec$sbc, fit.EXP.AcceSec$sbc, fit.LOGN
O.AcceSec$sbc, fit.WEI.AcceSec$sbc, fit.IG.AcceSec$sbc))
```

```
##                     DF    LOGLIK      AIC      BIC
## Gamma                2 -228.8409 461.6818 466.8518
## Exponentional        1 -289.3541 580.7083 583.2933
## Log-Normal           2 -232.4413 468.8825 474.0525
## Weibull              2 -226.6509 457.3017 462.4717
## Inverse Gaussian     2 -232.8252 469.6503 474.8203
## M.Gamma              5 -224.6183 459.2366 472.1614
## M.Exponentional      3 -289.3541 584.7083 592.4632
## M.Log-Normal         5 -224.4904 458.9809 471.9057
## M.Weibull            5 -225.9496 461.8992 474.8240
## M.Inverse Gaussian   5 -224.4566 458.9131 471.8380
```

In the first column, the family with "M." at the start indicate that it is a mixture. In the second column, it's possible to look at the number of parameters. The best fitting is performed through a mixture distribution with the Inverse Gaussian family but overall the best fit is weibull.

2.2.4. TopSpeed_KmH

This is a numeric continuous variable, measured in kilometres per hour. The basic statistical values are mentioned below.

```
{r}
summary(vehicle$TopSpeed_KmH)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  123.0   150.0   167.0   181.7   200.0   410.0 
```

```
{r}
sd(vehicle$TopSpeed_KmH)
```

```
[1] 43.24853
```

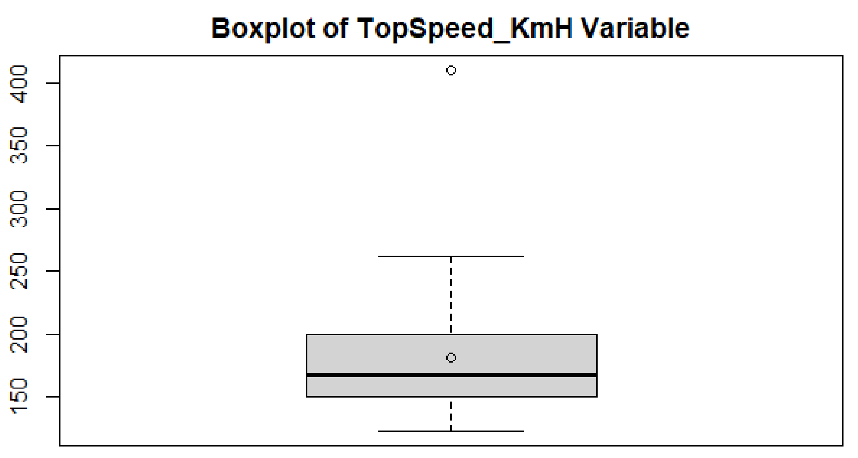
```
{r}
var(vehicle$TopSpeed_KmH)
```

```
[1] 1870.435
```

As we can see the median is less than the mean, hence the distribution should be positively skewed.

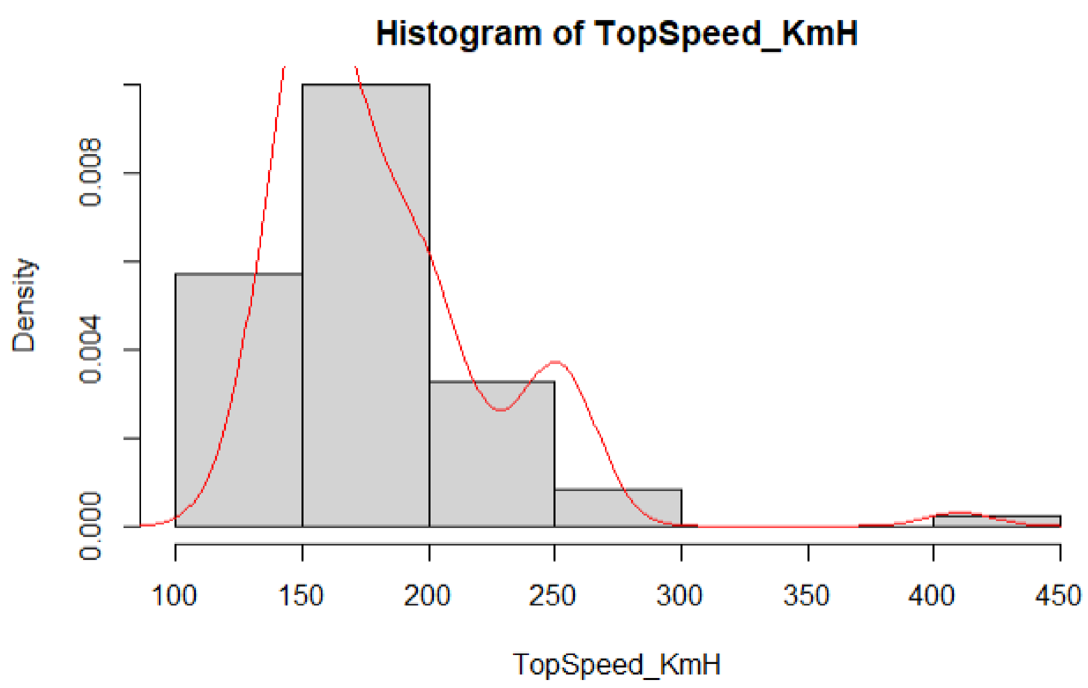
The boxplot of this variable is:

```
{r}
boxplot(vehicle$TopSpeed_KmH, main = "Boxplot of TopSpeed_KmH Variable")
points(mean(vehicle$TopSpeed_KmH))
```



While reviewing the box plot we can observe that there are some outliers in this variable as we see few dot lie outside the whiskers of the boxplot and it was also more clarified that the mean is above the median, outliers are the data points located outside the whiskers of the boxplot. Now we can plot the histogram along with skewness and kurtosis to look for the model that fits better the distribution:

```r
{r}
hist(vehicle$TopSpeed_KmH, freq=FALSE, main = "Histogram of TopSpeed_KmH", xlab = "
TopSpeed_KmH")
lines(density(vehicle$TopSpeed_KmH), col="red")
```




```r
{r}
skewness(vehicle$TopSpeed_KmH)
```

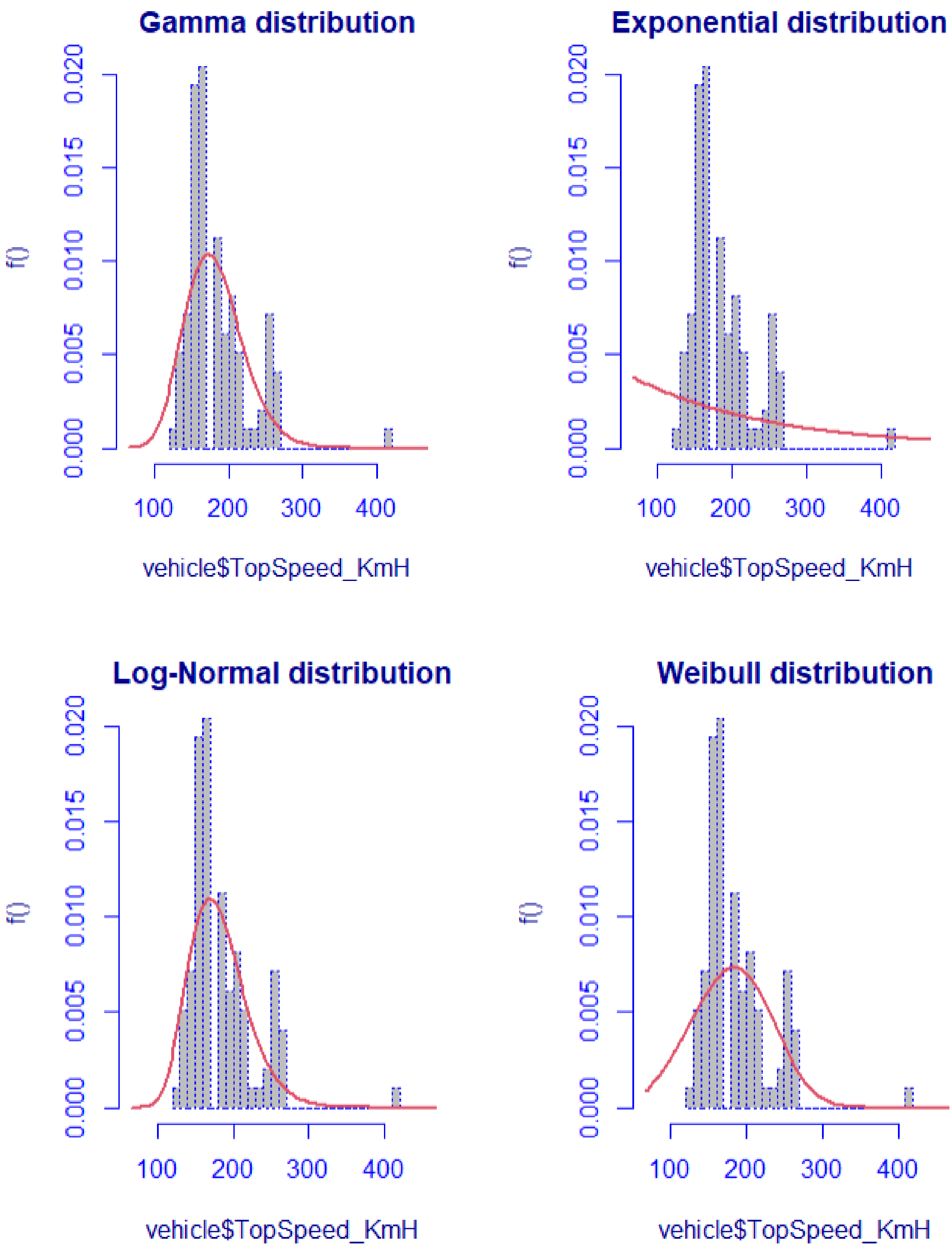
```
[1] 1.914129
```

```r
{r}
kurtosis(vehicle$TopSpeed_KmH)
```

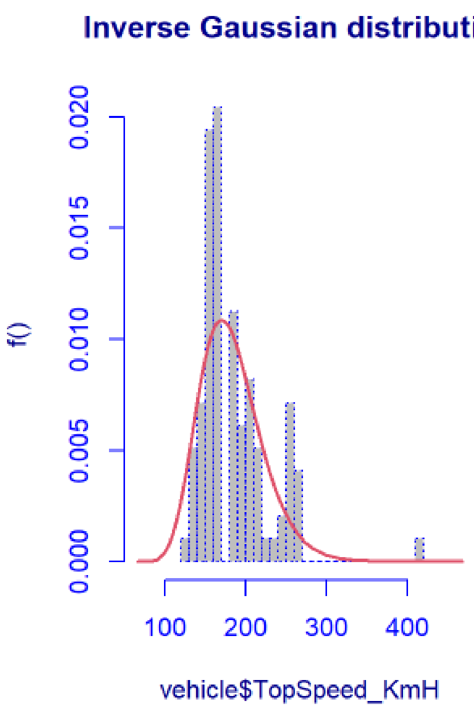
```
[1] 9.367375
```

As we can see from the histogram it is not a normal distribution because the skewness is not equal to exactly zero. According to the histogram, our distribution is positively-skewed (1.914129). Hence our above prediction was correct and kurtosis (9.367375) indicates that the distribution has more outliers than a normal distribution it is also known as leptokurtic.

To fit this distribution, we are going to include in the comparison models which deal better with skewness:


Gamma distribution
Exponential distribution
Log-Normal distribution
Weibull distribution
f()
0.000
0.005
0.010
0.015
0.020
100
200
300
400
vehicle$TopSpeed_KmH

```r
fit.IG <- histDist(vehicle$TopSpeed_KmH, family=IG, nbins=30, main="Inverse Gaussian distribution")
```




```r
data.frame(row.names=c("Gamma", "Exponentional", "Log-Normal", "Weibull","Inverse Gaussian"),
 DF=c(fit.gamma$df.fit, fit.EXP$df.fit, fit.LOGNO$df.fit, fit.WEI$df.fit, fit.IG$df.fit),
 LOGLIK=c(logLik(fit.gamma),logLik(fit.EXP),logLik(fit.LOGNO),logLik(fit.WEI),logLik(fit.IG)),
 AIC=c(AIC(fit.gamma), AIC(fit.EXP), AIC(fit.LOGNO),AIC(fit.WEI), AIC(fit.IG)),
 BIC=c(fit.gamma$sbc, fit.EXP$sbc, fit.LOGNO$sbc, fit.WEI$sbc, fit.IG$sbc))
```

```
##                  DF    LOGLIK       AIC       BIC
## Gamma             2 -497.4071  998.8142 1003.9841
## Exponentional     1 -607.8057 1217.6113 1220.1963
## Log-Normal        2 -493.6110  991.2221  996.3920
## Weibull           2 -516.2774 1036.5547 1041.7246
## Inverse Gaussian  2 -493.7563  991.5127  996.6826
```

As we can see from the above chart with Likelihood, AIC and BIC, the model log-normal fits better the distribution for TopSpeed.

The mixture of distribution:

It is possible to compute a mixture of two distributions to find the best mixture, the algorithm is repeated five times.

```
{r}
par(mfrow=c(1,3))
fit.GA.AcceSec <- gamlssMXfits(n = 5, vehicle$TopSpeed_KmH~1, family = GA, K = 2,
data = vehicle)
par(mfrow=c(1,3))
fit.EXP.AcceSec <- gamlssMXfits(n = 5, vehicle$TopSpeed_KmH~1, family = EXP, K = 2,
data = vehicle)
par(mfrow=c(1,3))
fit.LOGNO.AcceSec <- gamlssMXfits(n = 5, vehicle$TopSpeed_KmH~1, family = LOGNO, K
= 2, data = vehicle)
par(mfrow=c(1,3))
fit.WEI.AcceSec <- gamlssMXfits(n = 5, vehicle$TopSpeed_KmH~1, family = WEI, K = 2,
data = vehicle)
par(mfrow=c(1,3))
fit.IG.AcceSec <- gamlssMXfits(n = 5, vehicle$TopSpeed_KmH~1, family = IG, K = 2,
data = vehicle)
par(mfrow=c(1,3))
fit.GG.AcceSec <- gamlssMXfits(n = 5, vehicle$TopSpeed_KmH~1, family = GG, K = 2,
data = vehicle)
```

```
data.frame(row.names=c("Gamma", "Exponentional", "Log-Normal", "Weibull","Inverse Gaussian","M.Gamma", "M.Exponentional",
"M.Log-Normal", "M.Weibull","M.Inverse Gaussian" ),

 DF=c(fit.gamma$df.fit, fit.EXP$df.fit, fit.LOGNO$df.fit, fit.WEI$df.fit, fit.IG$df.fit, fit.GA.TopSpeed_KmH$df.fit, fit.EX
P.TopSpeed_KmH$df.fit, fit.LOGNO.TopSpeed_KmH$df.fit, fit.WEI.TopSpeed_KmH$df.fit, fit.IG.TopSpeed_KmH$df.fit),

 LOGLIK=c(logLik(fit.gamma),logLik(fit.EXP),logLik(fit.LOGNO),logLik(fit.WEI),logLik(fit.IG),logLik(fit.GA.TopSpeed_KmH),log
Lik(fit.EXP.TopSpeed_KmH),logLik(fit.LOGNO.TopSpeed_KmH),logLik(fit.WEI.TopSpeed_KmH),logLik(fit.IG.TopSpeed_KmH)),

 AIC=c(AIC(fit.gamma), AIC(fit.EXP), AIC(fit.LOGNO),AIC(fit.WEI), AIC(fit.IG),AIC(fit.GA.TopSpeed_KmH), AIC(fit.EXP.TopSpeed
_KmH), AIC(fit.LOGNO.TopSpeed_KmH),AIC(fit.WEI.TopSpeed_KmH), AIC(fit.IG.TopSpeed_KmH)),

 BIC=c(fit.gamma$sbc, fit.EXP$sbc, fit.LOGNO$sbc, fit.WEI$sbc, fit.IG$sbc,fit.GA.TopSpeed_KmH$sbc, fit.EXP.TopSpeed_KmH$sbc,
fit.LOGNO.TopSpeed_KmH$sbc, fit.WEI.TopSpeed_KmH$sbc, fit.IG.TopSpeed_KmH$sbc))
```

```
##                    DF    LOGLIK       AIC       BIC
## Gamma               2 -497.4071  998.8142 1003.9841
## Exponentional       1 -607.8057 1217.6113 1220.1963
## Log-Normal          2 -493.6110  991.2221  996.3920
## Weibull             2 -516.2774 1036.5547 1041.7246
## Inverse Gaussian    2 -493.7563  991.5127  996.6826
## M.Gamma             5 -484.3440  978.6880  991.6128
## M.Exponentional     3 -607.8057 1221.6113 1229.3662
## M.Log-Normal        5 -482.9450  975.8900  988.8148
## M.Weibull           5 -496.3728 1002.7456 1015.6705
## M.Inverse Gaussian  5 -483.0088  976.0177  988.9425
```

In the first column, the family with “M.” at the start indicate that it is a mixture. In the second column, it’s possible to look at the number of parameters. The best fitting is performed through a mixture distribution with the log-normal family.

2.2.5. Range_Km

This is a numeric continuous variable, measured in acceleration per second. The basic statistical values are mentioned below.

```
{r}
summary(vehicle$Range_Km)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  170.0   258.8   350.0   350.2   407.5   970.0 
```

```
{r}
sd(vehicle$Range_Km)

[1] 118.218
```

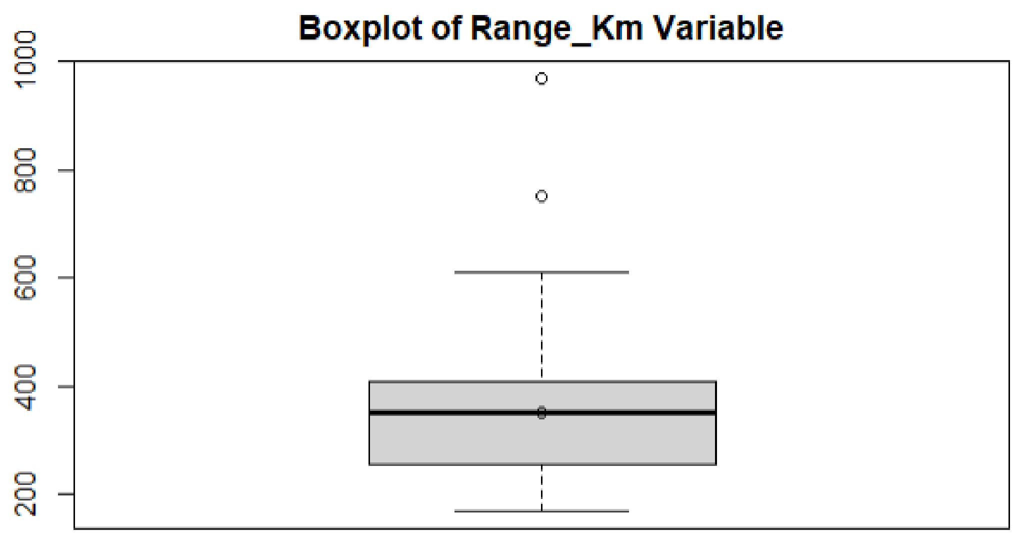
```
{r}
var(vehicle$Range_Km)

[1] 13975.49
```

As we can see the median is less than the mean, hence the distribution should be positively skewed.

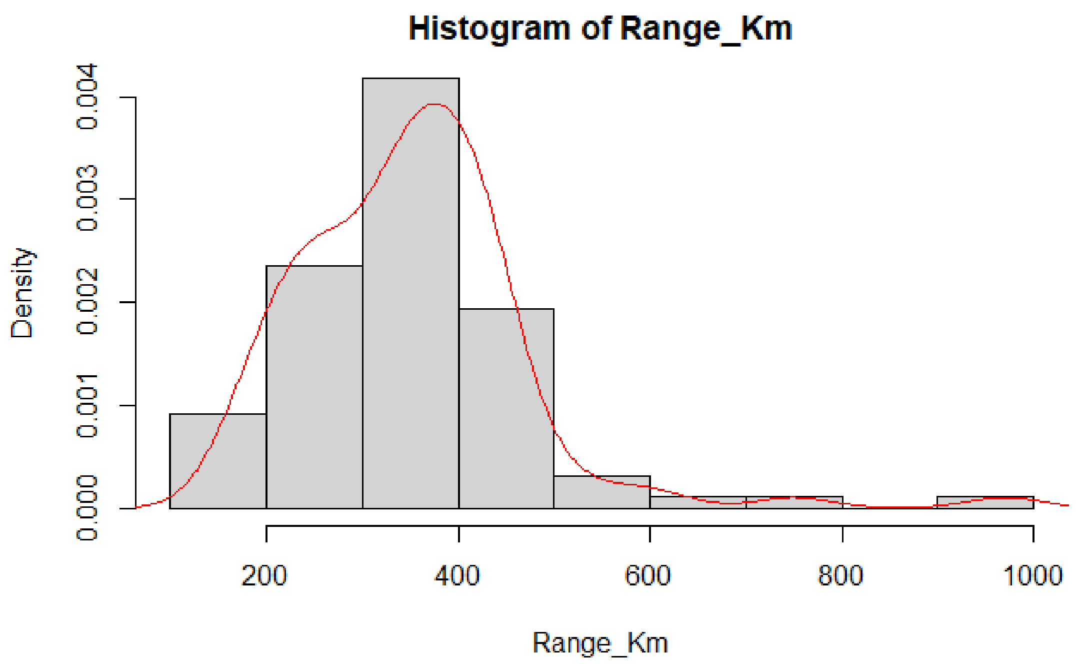
The boxplot of this variable is:

```
{r}
boxplot(vehicle$Range_Km, main = "Boxplot of Range_Km Variable")
points(mean(vehicle$Range_Km))
```



While reviewing the box plot we can observe that there are some outliers in this variable as we see few dots lie outside the whiskers of the boxplot and it was also more clarified that the mean is slightly above the median, outliers are the data points located outside the whiskers of the boxplot. Now we can plot the histogram along with skewness and kurtosis to look for the model that fits better the distribution:

```
{r}
hist(vehicle$Range_Km, freq=FALSE, main = "Histogram of Range_Km", xlab = "
Range_Km")
lines(density(vehicle$Range_Km), col="red")
```




```
{r}
skewness(vehicle$Range_Km)
```

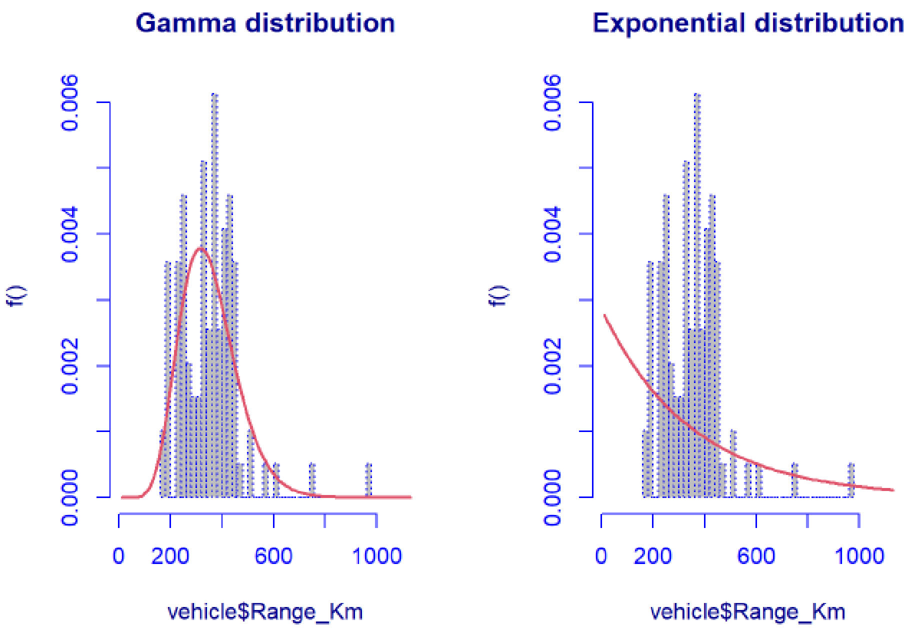
```
[1] 1.800251
```

```
{r}
kurtosis(vehicle$Range_Km)
```

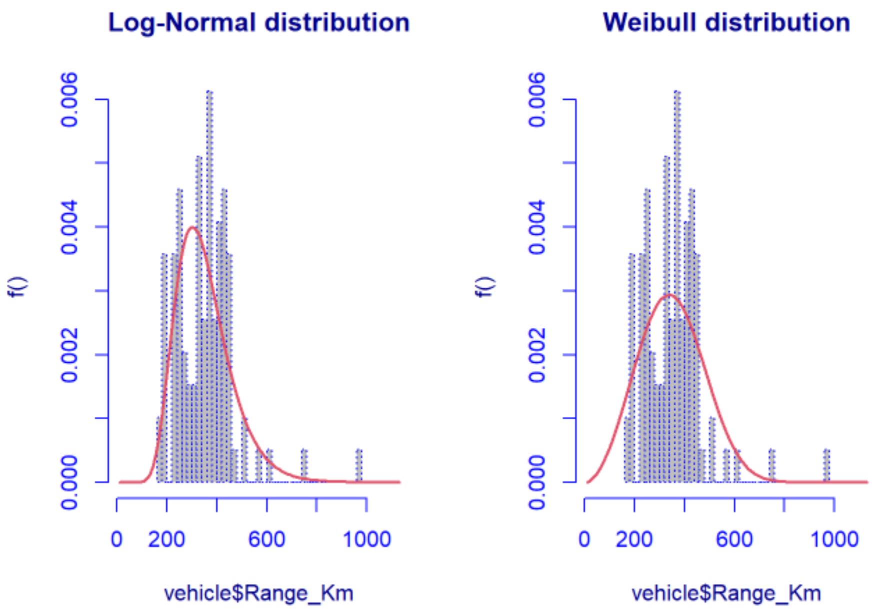
```
[1] 10.24374
```

As we can see from the histogram it is not a normal distribution because the skewness is not equal to exactly zero. According to the histogram, our distribution is positively-skewed (1.800251). Hence our above prediction was correct and kurtosis (10.24374) indicates that the distribution has more outliers than a normal distribution it is also known as leptokurtic.

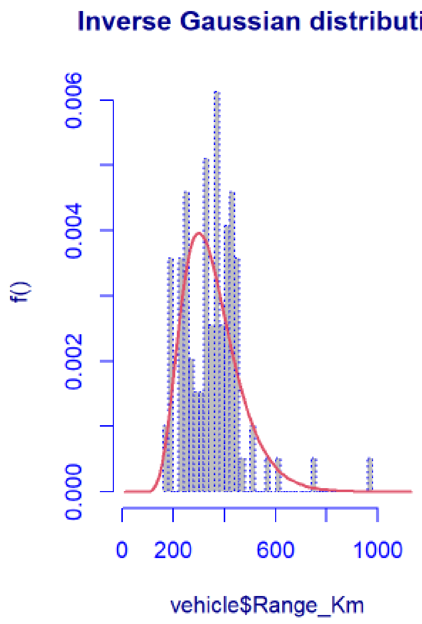
```
par(mfrow=c(1,2))
fit.gamma <- histDist(vehicle$Range_Km, family=GA, nbins=30, main="Gamma distribution")
fit.EXP <- histDist(vehicle$Range_Km, family=EXP, nbins=30, main="Exponential distribution")
```




```
fit.LOGNO <- histDist(vehicle$Range_Km, family=LOGNO, nbins=30, main="Log-Normal distribution")
fit.WEI <- histDist(vehicle$Range_Km, family=WEI, nbins=30, main="Weibull distribution")
```



```r
fit.IG <- histDist(vehicle$Range_Km, family=IG, nbins=30, main="Inverse Gaussian distribution")
```




```r
{r}
data.frame(row.names=c("Gamma", "Exponentional", "Log-Normal", "Weibull","Inverse Gaussian"),
 DF=c(fit.gamma$df.fit, fit.EXP$df.fit, fit.LOGNO$df.fit, fit.WEI$df.fit, fit.IG$df.fit),
 LOGLIK=c(logLik(fit.gamma),logLik(fit.EXP),logLik(fit.LOGNO),logLik(fit.WEI),logLik(fit.IG)),
 AIC=c(AIC(fit.gamma), AIC(fit.EXP), AIC(fit.LOGNO),AIC(fit.WEI), AIC(fit.IG)),
 BIC=c(fit.gamma$sbc, fit.EXP$sbc, fit.LOGNO$sbc, fit.WEI$sbc, fit.IG$sbc))
```

Description: df [5 × 4]

	DF <dbl>	LOGLIK <dbl>	AIC <dbl>	BIC <dbl>
Gamma	2	-596.3399	1196.680	1201.850
Exponentional	1	-672.1203	1346.241	1348.826
Log-Normal	2	-594.8122	1193.624	1198.794
Weibull	2	-607.6048	1219.210	1224.379
Inverse Gaussian	2	-595.0629	1194.126	1199.296

5 rows

As we can see from the above chart with Likelihood, AIC and BIC, the model log-normal fits better the distribution for Range_Km.

The mixture of distribution:

It is possible to compute a mixture of two distributions to find the best mixture, the algorithm is repeated five times.

```
{r}
par(mfrow=c(1,3))
fit.GA.Range_Km <- gamlssMXfits(n = 5, vehicle$Range_Km~1, family = GA, K = 2, data
= vehicle)
par(mfrow=c(1,3))
fit.EXP.Range_Km <- gamlssMXfits(n = 5, vehicle$Range_Km~1, family = EXP, K = 2,
data = vehicle)
par(mfrow=c(1,3))
fit.LOGNO.Range_Km <- gamlssMXfits(n = 5, vehicle$Range_Km~1, family = LOGNO, K = 2
, data = vehicle)
par(mfrow=c(1,3))
fit.WEI.Range_Km <- gamlssMXfits(n = 5, vehicle$Range_Km~1, family = WEI, K = 2,
data = vehicle)
par(mfrow=c(1,3))
fit.IG.Range_Km <- gamlssMXfits(n = 5, vehicle$Range_Km~1, family = IG, K = 2, data
= vehicle)
par(mfrow=c(1,3))
```

```
{r}
data.frame(row.names=c("Gamma", "Exponentional", "Log-Normal", "Weibull","Inverse
Gaussian","M.Gamma", "M.Exponentional", "M.Log-Normal", "M.Weibull","M.Inverse
Gaussian" ),
 DF=c(fit.gamma$df.fit, fit.EXP$df.fit, fit.LOGNO$df.fit, fit.WEI$df.fit, fit.IG$df
.fit, fit.GA.Range_Km$df.fit, fit.EXP.Range_Km$df.fit, fit.LOGNO.Range_Km$df.fit,
fit.WEI.Range_Km$df.fit, fit.IG.Range_Km$df.fit),
 LOGLIK=c(logLik(fit.gamma),logLik(fit.EXP),logLik(fit.LOGNO),logLik(fit.WEI
),logLik(fit.IG),logLik(fit.GA.Range_Km),logLik(fit.EXP.Range_Km),logLik(fit.LOGNO
.Range_Km),logLik(fit.WEI.Range_Km),logLik(fit.IG.Range_Km)),
 AIC=c(AIC(fit.gamma), AIC(fit.EXP), AIC(fit.LOGNO),AIC(fit.WEI), AIC(fit.IG),AIC
(fit.GA.Range_Km), AIC(fit.EXP.Range_Km), AIC(fit.LOGNO.Range_Km),AIC(fit.WEI
.Range_Km), AIC(fit.IG.Range_Km)),
 BIC=c(fit.gamma$sbc, fit.EXP$sbc, fit.LOGNO$sbc, fit.WEI$sbc, fit.IG$sbc,fit.GA
.Range_Km$sbc, fit.EXP.Range_Km$sbc, fit.LOGNO.Range_Km$sbc, fit.WEI.Range_Km$sbc,
fit.IG.Range_Km$sbc))
```

Description: df [10 × 4]

	DF <dbl>	LOGLIK <dbl>	AIC <dbl>	BIC <dbl>
Gamma	2	-596.3399	1196.680	1201.850
Exponentional	1	-672.1203	1346.241	1348.826
Log-Normal	2	-594.8122	1193.624	1198.794
Weibull	2	-607.6048	1219.210	1224.379
Inverse Gaussian	2	-595.0629	1194.126	1199.296
M.Gamma	5	-591.5086	1193.017	1205.942
M.Exponentional	3	-672.1203	1350.241	1357.995
M.Log-Normal	5	-588.3858	1186.772	1199.696
M.Weibull	5	-590.3454	1190.691	1203.616
M.Inverse Gaus...	5	-588.4523	1186.905	1199.829

In the first column, the family with “M.” at the start indicate that it is a mixture. In the second column, it’s possible to look at the number of parameters. The best fitting is performed through a mixture distribution with the log-normal family.

2.2.6. Efficiency_whKm

This is a numeric continuous variable, measured in watt hour per Kilometre. The basic statistical values are mentioned below.

```
{r}
summary(vehicle$Efficiency_WhKm)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  104.0   168.0   181.0   189.9   206.0   273.0 
```

```
{r}
sd(vehicle$Efficiency_WhKm)

 [1] 30.05584
```

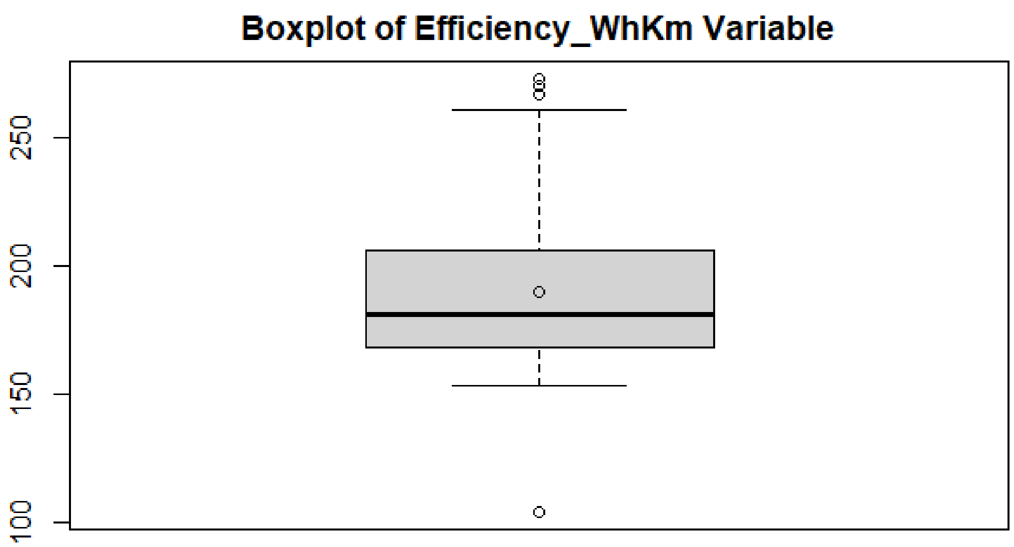
```
{r}
var(vehicle$Efficiency_WhKm)

 [1] 903.3534
```

As we can see the median is less than the mean, hence the distribution should be positively skewed.

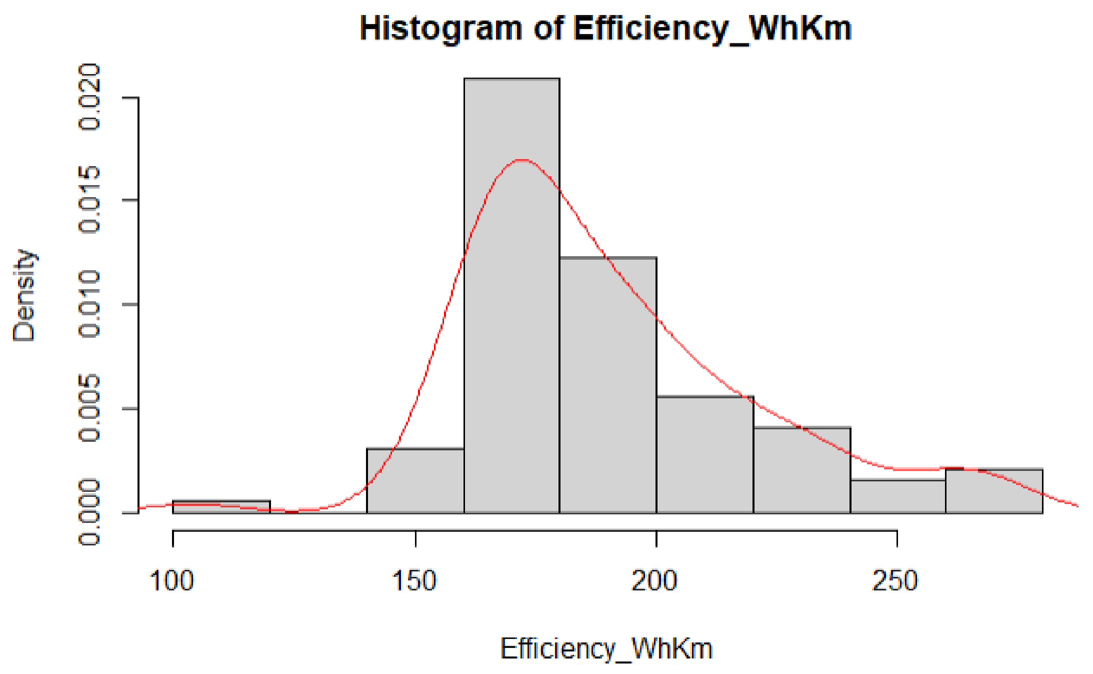
The boxplot of this variable is:

```
{r}
boxplot(vehicle$Efficiency_WhKm, main = "Boxplot of Efficiency_WhKm Variable")
points(mean(vehicle$Efficiency_WhKm))
```



While reviewing the box plot we can observe that there are some outliers in this variable as we see few dots lie outside the whiskers of the boxplot and it was also more clarified that the mean is above the median, outliers are the data points located outside the whiskers of the boxplot. Now we can plot the histogram along with skewness and kurtosis to look for the model that fits better the distribution:

```r
{r}
hist(vehicle$Efficiency_WhKm, freq=FALSE, main = "Histogram of Efficiency_WhKm",
xlab = "Efficiency_WhKm")
lines(density(vehicle$Efficiency_WhKm), col="red")
```




```r
{r}
skewness(vehicle$Efficiency_WhKm)
```

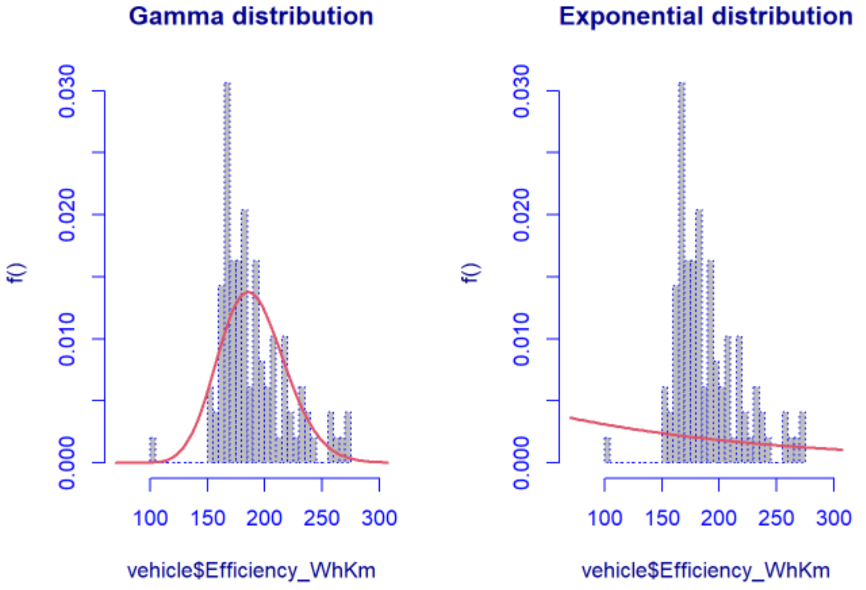
```
[1] 0.8014048
```

```r
{r}
kurtosis(vehicle$Efficiency_WhKm)
```

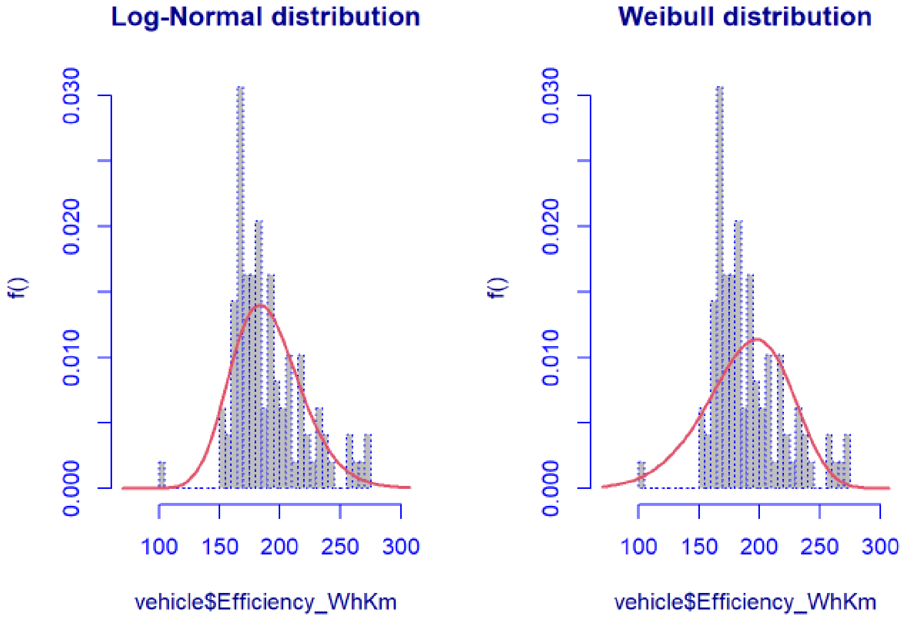
```
[1] 3.809173
```

As we can see from the histogram it is not a normal distribution because the skewness is not equal to exactly zero. According to the histogram, our distribution is positively-skewed (0.80.14048). Hence our above prediction was correct and kurtosis (3.809173) indicates that the distribution has more outliers than a normal distribution it is also known as leptokurtic.

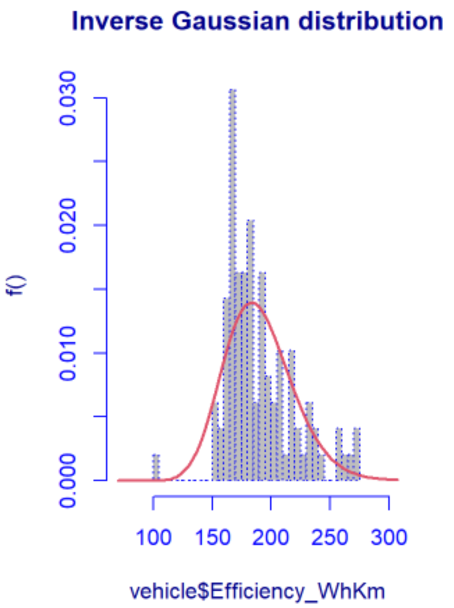
```
par(mfrow=c(1,2))
fit.gamma <- histDist(vehicle$Efficiency_WhKm, family=GA, nbins=30, main="Gamma distribution")
fit.EXP <- histDist(vehicle$Efficiency_WhKm, family=EXP, nbins=30, main="Exponential distribution")
```




```
fit.LOGNO <- histDist(vehicle$Efficiency_WhKm, family=LOGNO, nbins=30, main="Log-Normal distribution")
fit.WEI <- histDist(vehicle$Efficiency_WhKm, family=WEI, nbins=30, main="Weibull distribution")
```



```
fit.IG <- histDist(vehicle$Efficiency_WhKm, family=IG, nbins=30, main="Inverse Gaussian distribution")
```




```
{r}
data.frame(row.names=c("Gamma", "Exponentional", "Log-Normal", "Weibull","Inverse Gaussian"),
 DF=c(fit.gamma$df.fit, fit.EXP$df.fit, fit.LOGNO$df.fit, fit.WEI$df.fit, fit.IG$df.fit),
 LOGLIK=c(logLik(fit.gamma),logLik(fit.EXP),logLik(fit.LOGNO),logLik(fit.WEI),logLik(fit.IG)),
 AIC=c(AIC(fit.gamma), AIC(fit.EXP), AIC(fit.LOGNO),AIC(fit.WEI), AIC(fit.IG)),
 BIC=c(fit.gamma$sbc, fit.EXP$sbc, fit.LOGNO$sbc, fit.WEI$sbc, fit.IG$sbc))
```

Description: df [5 × 4]

	DF <dbl>	LOGLIK <dbl>	AIC <dbl>	BIC <dbl>
Gamma	2	-469.0996	942.1993	947.3692
Exponentional	1	-612.1399	1226.2798	1228.8648
Log-Normal	2	-468.3743	940.7486	945.9186
Weibull	2	-480.2388	964.4777	969.6476
Inverse Gaussian	2	-468.5113	941.0225	946.1924

5 rows

As we can see from the above chart with Likelihood, AIC and BIC, the model log-normal fits better the distribution for Efficiency_whKm

The mixture of distribution:

It is possible to compute a mixture of two distributions to find the best mixture, the algorithm is repeated five times.

```
{r}
par(mfrow=c(1,3))
fit.GA.Efficiency_WhKm <- gamlssMXfits(n = 5, vehicle$Efficiency_WhKm~1, family =
GA, K = 2, data = vehicle)
par(mfrow=c(1,3))
fit.EXP.Efficiency_WhKm <- gamlssMXfits(n = 5, vehicle$Efficiency_WhKm~1, family =
EXP, K = 2, data = vehicle)
par(mfrow=c(1,3))
fit.LOGNO.Efficiency_WhKm <- gamlssMXfits(n = 5, vehicle$Efficiency_WhKm~1, family
= LOGNO, K = 2, data = vehicle)
par(mfrow=c(1,3))
fit.WEI.Efficiency_WhKm <- gamlssMXfits(n = 5, vehicle$Efficiency_WhKm~1, family =
WEI, K = 2, data = vehicle)
par(mfrow=c(1,3))
fit.IG.Efficiency_WhKm <- gamlssMXfits(n = 5, vehicle$Efficiency_WhKm~1, family =
IG, K = 2, data = vehicle)
par(mfrow=c(1,3))
```

```
{r}
data.frame(row.names=c("Gamma", "Exponentional", "Log-Normal", "Weibull","Inverse
Gaussian","M.Gamma", "M.Exponentional", "M.Log-Normal", "M.Weibull","M.Inverse
Gaussian" ),
 DF=c(fit.gamma$df.fit, fit.EXP$df.fit, fit.LOGNO$df.fit, fit.WEI$df.fit, fit.IG$df
.fit, fit.GA.Efficiency_WhKm$df.fit, fit.EXP.Efficiency_WhKm$df.fit, fit.LOGNO
.Efficiency_WhKm$df.fit, fit.WEI.Efficiency_WhKm$df.fit, fit.IG.Efficiency_WhKm$df
.fit),
 LOGLIK=c(logLik(fit.gamma),logLik(fit.EXP),logLik(fit.LOGNO),logLik(fit.WEI
),logLik(fit.IG),logLik(fit.GA.Efficiency_WhKm),logLik(fit.EXP.Efficiency_WhKm
),logLik(fit.LOGNO.Efficiency_WhKm),logLik(fit.WEI.Efficiency_WhKm),logLik(fit.IG
.Efficiency_WhKm)),
 AIC=c(AIC(fit.gamma), AIC(fit.EXP), AIC(fit.LOGNO),AIC(fit.WEI), AIC(fit.IG),AIC
(fit.GA.Efficiency_WhKm), AIC(fit.EXP.Efficiency_WhKm), AIC(fit.LOGNO
.Efficiency_WhKm),AIC(fit.WEI.Efficiency_WhKm), AIC(fit.IG.Efficiency_WhKm)),
 BIC=c(fit.gamma$sbc, fit.EXP$sbc, fit.LOGNO$sbc, fit.WEI$sbc, fit.IG$sbc,fit.GA
.Efficiency_WhKm$sbc, fit.EXP.Efficiency_WhKm$sbc, fit.LOGNO.Efficiency_WhKm$sbc,
fit.WEI.Efficiency_WhKm$sbc, fit.IG.Efficiency_WhKm$sbc))
```

	DF <dbl>	LOGLIK <dbl>	AIC <dbl>	BIC <dbl>
Gamma	2	-469.0996	942.1993	947.3692
Exponentional	1	-612.1399	1226.2798	1228.8648
Log-Normal	2	-468.3743	940.7486	945.9186
Weibull	2	-480.2388	964.4777	969.6476
Inverse Gaussian	2	-468.5113	941.0225	946.1924
M.Gamma	5	-457.9782	925.9565	938.8813
M.Exponentional	3	-612.1399	1230.2798	1238.0347
M.Log-Normal	5	-458.7821	927.5642	940.4890
M.Weibull	5	-460.8481	931.6963	944.6211
M.Inverse Gaus...	5	-458.9539	927.9079	940.8327

1-10 of 10 rows

Previously we saw, according to the likelihood the best-fitted distribution was log-normal. But after applying the mixture of distribution, the best fitting is performed through a mixture distribution with the gamma family.

2.2.7. FastCharge_KmH

This is a numeric continuous variable, measured in Kilometre per hour. The basic statistical values are mentioned below.

```r
{r}
summary(vehicle$FastCharge_KmH)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  170.0   275.0   440.0   456.7   560.0   940.0 
```

```r
{r}
sd(vehicle$FastCharge_KmH)
```

```
[1] 201.2629
```

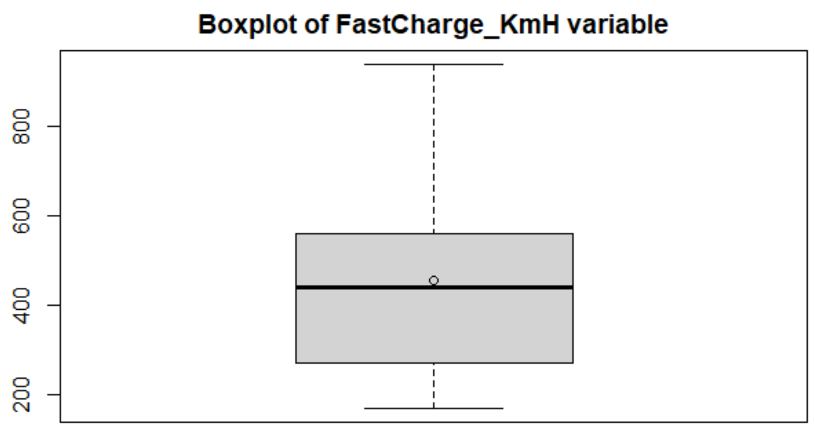
```r
{r}
var(vehicle$FastCharge_KmH)
```

```
[1] 40506.75
```

As we can see the median is less than the mean, hence the distribution should be positively skewed.

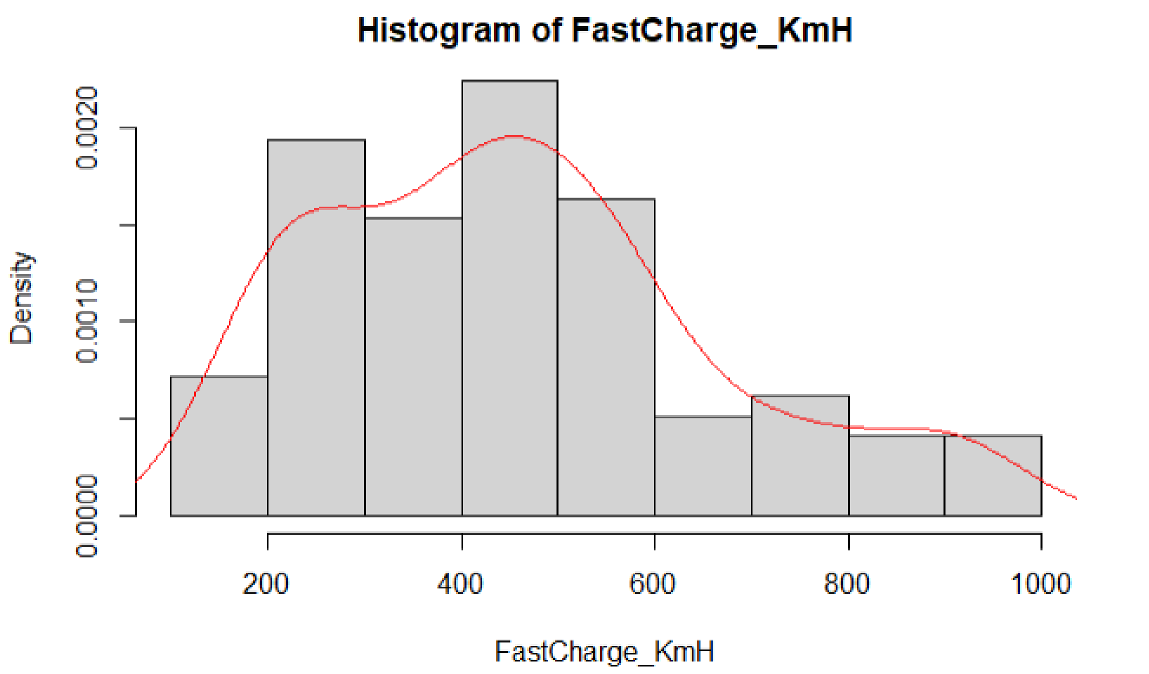
The boxplot of this variable is:

```r
{r}
boxplot(vehicle$FastCharge_KmH, main = "Boxplot of FastCharge_KmH variable")
points(mean(vehicle$FastCharge_KmH))
```



While reviewing the box plot we can observe that there are no outliers in this variable and it was also more clarified that the mean is above the median. Now we can plot the histogram along with skewness and kurtosis to look for the model that fits better the distribution:

```{r}
hist(vehicle$FastCharge_KmH, freq = FALSE, main = "Histogram of FastCharge_KmH",
xlab = "FastCharge_KmH")
lines(density(vehicle$FastCharge_KmH),col="red")
```




```{r}
skewness(vehicle$FastCharge_KmH)
```

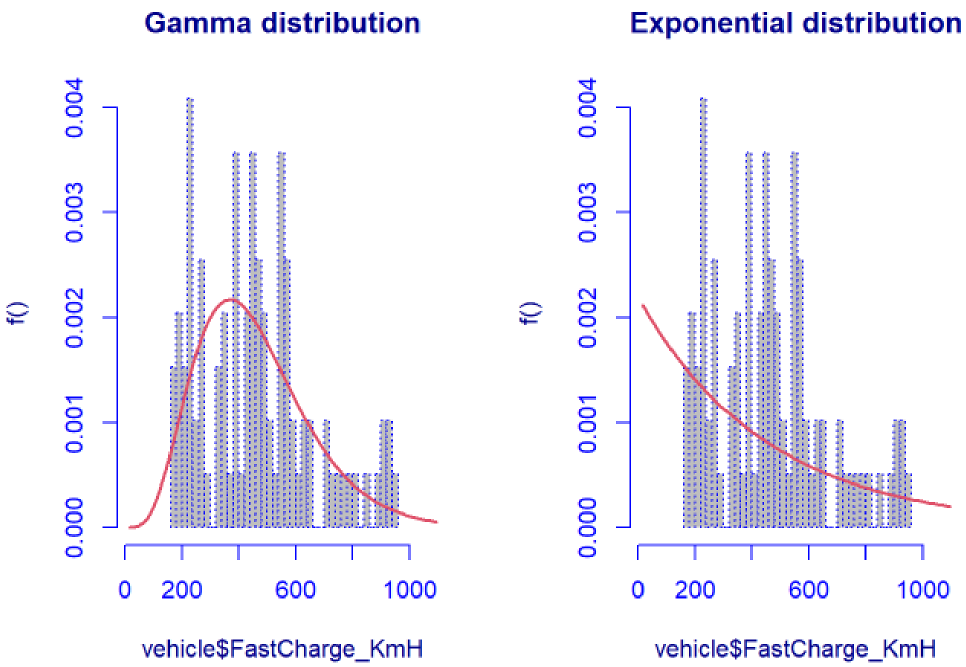
```
[1] 0.6397894
```

```{r}
kurtosis(vehicle$FastCharge_KmH)
```

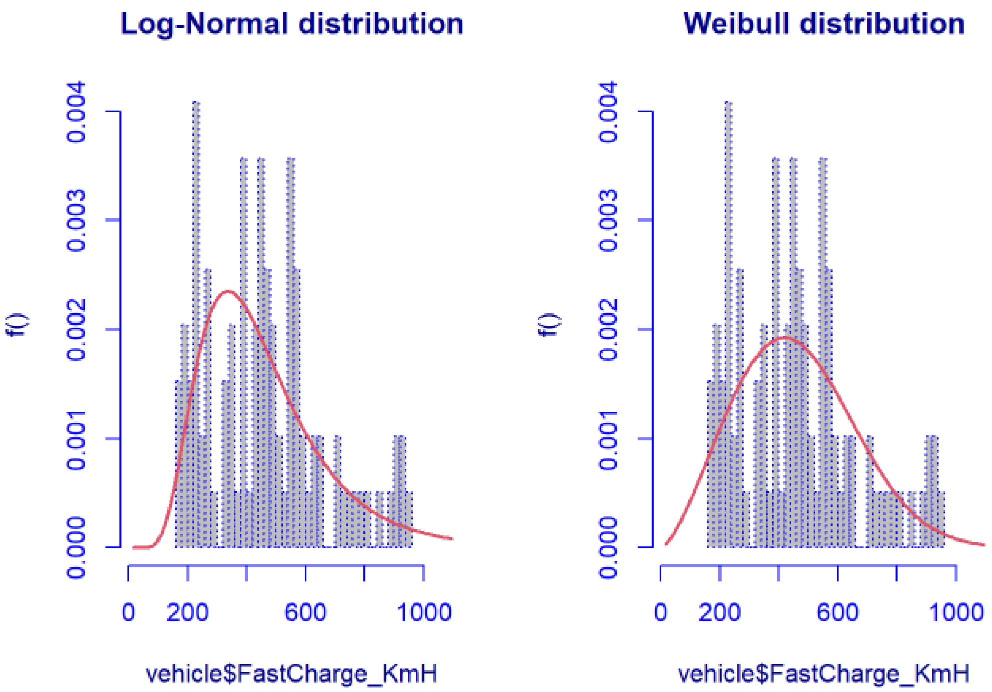
```
[1] 2.798588
```

As we can see from the histogram it is not a normal distribution because the skewness is not equal to exactly zero. According to the histogram, our distribution is positively-skewed (0.6397894). Hence our above prediction was correct and kurtosis is (2.798588). It is also known as platykurtic.

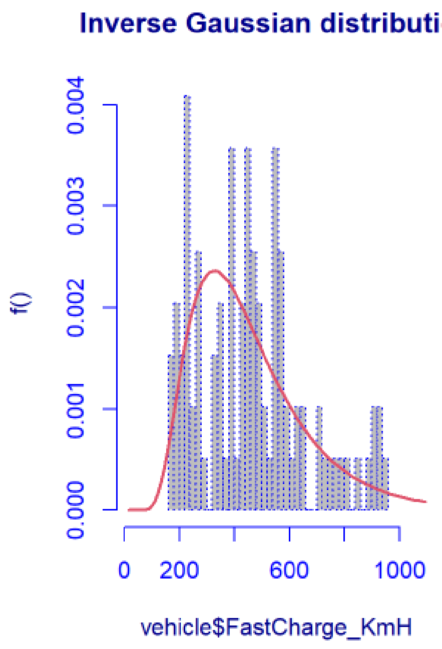
```
par(mfrow=c(1,2))
fit.gamma <- histDist(vehicle$FastCharge_KmH, family=GA, nbins=30, main="Gamma distribution")
fit.EXP <- histDist(vehicle$FastCharge_KmH, family=EXP, nbins=30, main="Exponential distribution")
```




```
fit.WEI <- histDist(vehicle$FastCharge_KmH, family=WEI, nbins=30, main="Weibull distribution")
```



```
fit.IG <- histDist(vehicle$FastCharge_KmH, family=IG, nbins=30, main="Inverse Gaussian distribution")
```




```
{r}
data.frame(row.names=c("Gamma", "Exponentional", "Log-Normal", "Weibull","Inverse Gaussian"),
 DF=c(fit.gamma$df.fit, fit.EXP$df.fit, fit.LOGNO$df.fit, fit.WEI$df.fit, fit.IG$df.fit),
 LOGLIK=c(logLik(fit.gamma),logLik(fit.EXP),logLik(fit.LOGNO),logLik(fit.WEI),logLik(fit.IG)),
 AIC=c(AIC(fit.gamma), AIC(fit.EXP), AIC(fit.LOGNO),AIC(fit.WEI), AIC(fit.IG)),
 BIC=c(fit.gamma$sbc, fit.EXP$sbc, fit.LOGNO$sbc, fit.WEI$sbc, fit.IG$sbc))
```

Description: df [5 × 4]

	DF <dbl>	LOGLIK <dbl>	AIC <dbl>	BIC <dbl>
Gamma	2	-651.9599	1307.920	1313.090
Exponentional	1	-698.1621	1398.324	1400.909
Log-Normal	2	-652.4292	1308.858	1314.028
Weibull	2	-654.0723	1312.145	1317.315
Inverse Gaussian	2	-651.9767	1307.953	1313.123

5 rows

As we can see from the above chart with Likelihood, AIC and BIC, the model gamma fits better the distribution for FastCharge_Km.

The mixture of distribution:

It is possible to compute a mixture of two distributions to find the best mixture, the algorithm is repeated five times.

```
{r}
par(mfrow=c(1,3))
fit.GA.FastCharge_KmH <- gamlssMXfits(n = 5, vehicle$FastCharge_KmH~1, family = GA,
K = 2, data = vehicle)
par(mfrow=c(1,3))
fit.EXP.FastCharge_KmH <- gamlssMXfits(n = 5, vehicle$FastCharge_KmH~1, family =
EXP, K = 2, data = vehicle)
par(mfrow=c(1,3))
fit.LOGNO.FastCharge_KmH <- gamlssMXfits(n = 5, vehicle$FastCharge_KmH~1, family =
LOGNO, K = 2, data = vehicle)
par(mfrow=c(1,3))
fit.WEI.FastCharge_KmH <- gamlssMXfits(n = 5, vehicle$FastCharge_KmH~1, family =
WEI, K = 2, data = vehicle)
par(mfrow=c(1,3))
fit.IG.FastCharge_KmH <- gamlssMXfits(n = 5, vehicle$FastCharge_KmH~1, family = IG,
K = 2, data = vehicle)
par(mfrow=c(1,3))
```

```
{r}
data.frame(row.names=c("Gamma", "Exponentional", "Log-Normal", "Weibull","Inverse
Gaussian","M.Gamma", "M.Exponentional", "M.Log-Normal", "M.Weibull","M.Inverse
Gaussian" ),
 DF=c(fit.gamma$df.fit, fit.EXP$df.fit, fit.LOGNO$df.fit, fit.WEI$df.fit, fit.IG$df
.fit, fit.GA.FastCharge_KmH$df.fit, fit.EXP.FastCharge_KmH$df.fit, fit.LOGNO
.FastCharge_KmH$df.fit, fit.WEI.FastCharge_KmH$df.fit, fit.IG.FastCharge_KmH$df.fit
),
 LOGLIK=c(logLik(fit.gamma),logLik(fit.EXP),logLik(fit.LOGNO),logLik(fit.WEI
),logLik(fit.IG),logLik(fit.GA.FastCharge_KmH),logLik(fit.EXP.FastCharge_KmH
),logLik(fit.LOGNO.FastCharge_KmH),logLik(fit.WEI.FastCharge_KmH),logLik(fit.IG
.FastCharge_KmH)),
 AIC=c(AIC(fit.gamma), AIC(fit.EXP), AIC(fit.LOGNO),AIC(fit.WEI), AIC(fit.IG),AIC
(fit.GA.FastCharge_KmH), AIC(fit.EXP.FastCharge_KmH), AIC(fit.LOGNO.FastCharge_KmH
),AIC(fit.WEI.FastCharge_KmH), AIC(fit.IG.FastCharge_KmH)),
 BIC=c(fit.gamma$sbc, fit.EXP$sbc, fit.LOGNO$sbc, fit.WEI$sbc, fit.IG$sbc,fit.GA
.FastCharge_KmH$sbc, fit.EXP.FastCharge_KmH$sbc, fit.LOGNO.FastCharge_KmH$sbc, fit
.WEI.FastCharge_KmH$sbc, fit.IG.FastCharge_KmH$sbc))
```

Description: df [10 x 4]

	DF <dbl>	LOGLIK <dbl>	AIC <dbl>	BIC <dbl>
Gamma	2	-651.9599	1307.920	1313.090
Exponentional	1	-698.1621	1398.324	1400.909
Log-Normal	2	-652.4292	1308.858	1314.028
Weibull	2	-654.0723	1312.145	1317.315
Inverse Gaussian	2	-651.9767	1307.953	1313.123
M.Gamma	5	-644.3234	1298.647	1311.572
M.Exponentional	3	-698.1621	1402.324	1410.079
M.Log-Normal	5	-643.8924	1297.785	1310.710
M.Weibull	5	-646.9592	1303.918	1316.843
M.Inverse Gaus...	5	-643.8076	1297.615	1310.540

1-10 of 10 rows

Previously we saw, according to the likelihood the best-fitted distribution was the gamma model. But after applying the mixture of distribution, the best fitting is performed through a mixture distribution with the inverse Gaussian family.

2.2.8. PriceEuro

This is a numeric continuous variable, measured in money(Euro). The basic statistical values are mentioned below.

```
{r}
summary(vehicle$PriceEuro)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  20129   35000   45000   57325   65465  215000 
```

```
{r}
sd(vehicle$PriceEuro)
```

```
[1] 34288.25
```

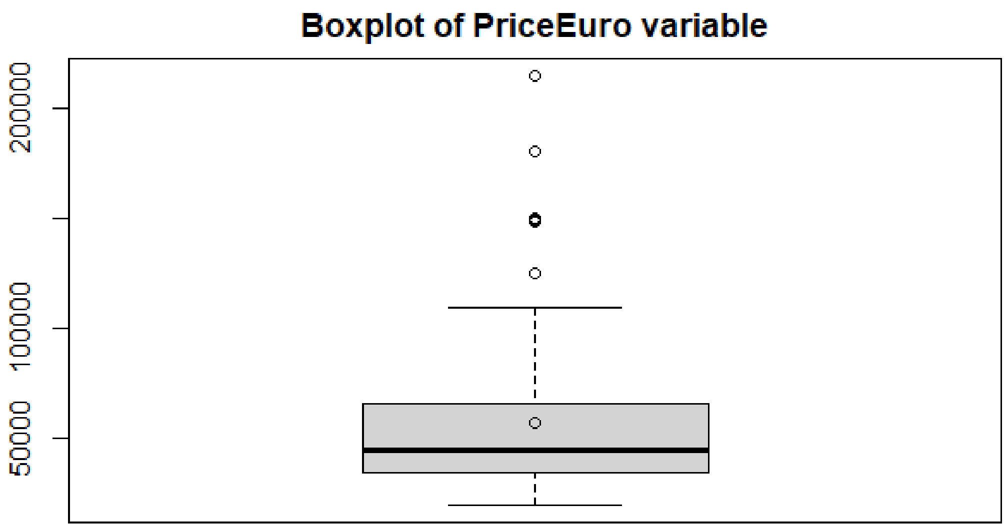
```
{r}
var(vehicle$PriceEuro)
```

```
[1] 1175683839
```

As we can see the median is less than the mean, hence the distribution should be positively skewed.

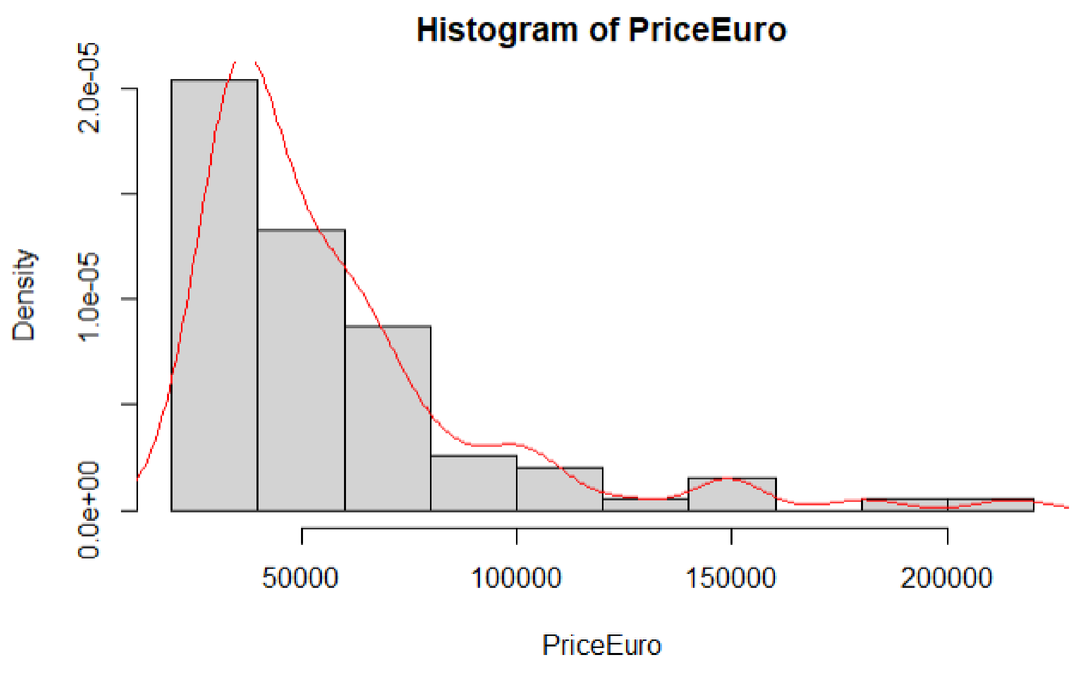
The boxplot of this variable is:

```
{r}
boxplot(vehicle$PriceEuro, main = "Boxplot of PriceEuro variable")
points(mean(vehicle$PriceEuro))
```



While reviewing the box plot we can observe that there are many outliers in this variable and it was also more clarified that the mean is above the median. Now we can plot the histogram along with skewness and kurtosis to look for the model that fits better the distribution:

```{r}
hist(vehicle$PriceEuro, freq = FALSE, main = "Histogram of PriceEuro", xlab =
"PriceEuro")
lines(density(vehicle$PriceEuro),col="red")
```




```{r}
skewness(vehicle$PriceEuro)
```

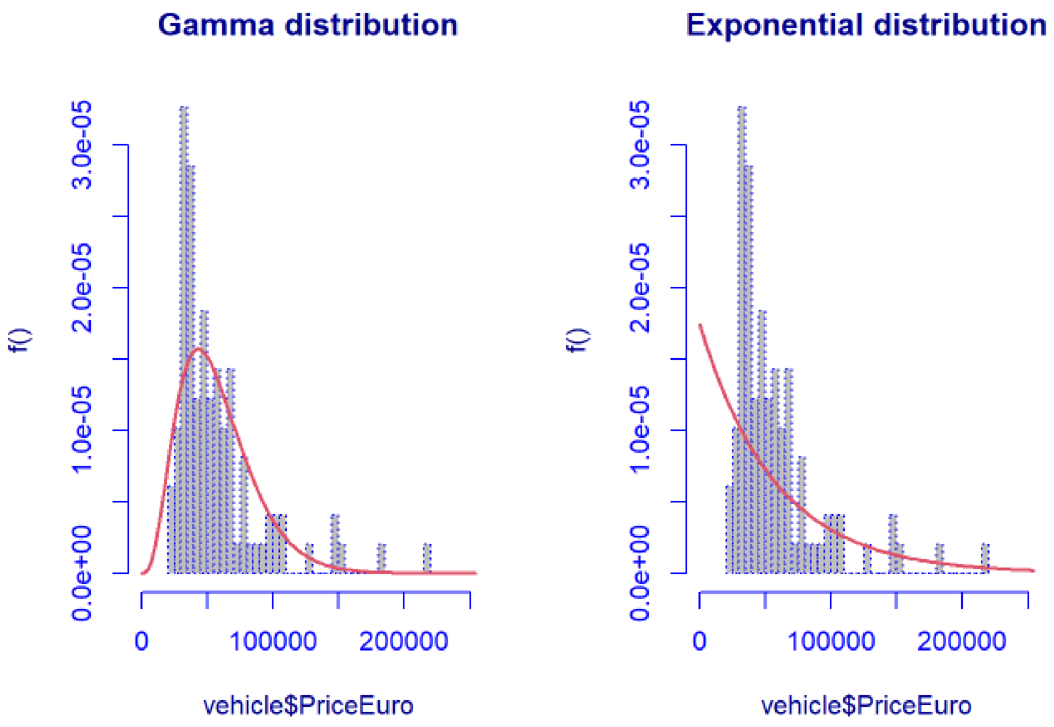
```
[1] 2.176277
```

```{r}
kurtosis(vehicle$PriceEuro)
```

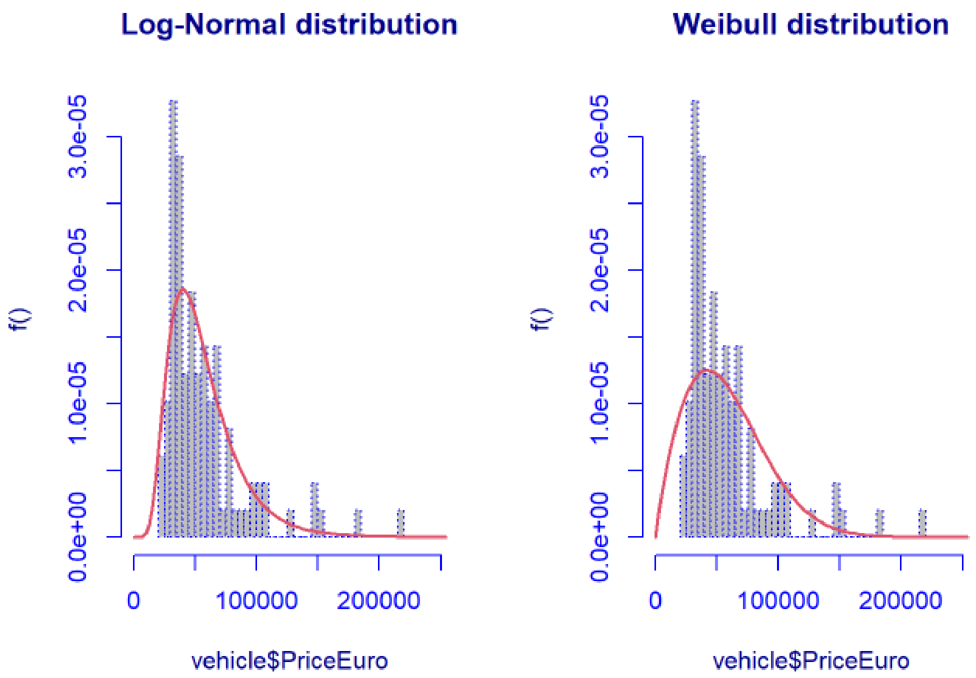
```
[1] 8.522458
```

As we can see from the histogram it is not a normal distribution because the skewness is not equal to exactly zero. According to the histogram, our distribution is positively-skewed (2.176277). Hence our above prediction was correct and kurtosis is (8.522458). It is greater than 3 so it is leptokurtic.

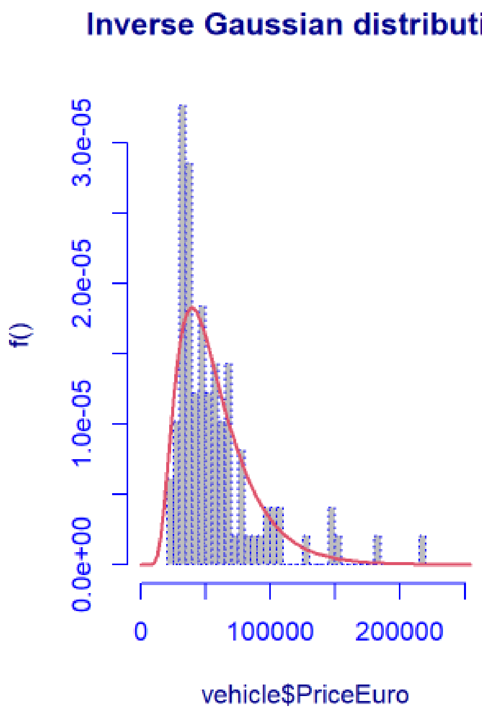
```
par(mfrow=c(1,2))
fit.gamma <- histDist(vehicle$PriceEuro, family=GA, nbins=30, main="Gamma distribution")
fit.EXP <- histDist(vehicle$PriceEuro, family=EXP, nbins=30, main="Exponential distribution")
```




```
fit.LOGNO <- histDist(vehicle$PriceEuro, family=LOGNO, nbins=30, main="Log-Normal distribution")
fit.WEI <- histDist(vehicle$PriceEuro, family=WEI, nbins=30, main="Weibull distribution")
```



```
fit.IG <- histDist(vehicle$PriceEuro, family=IG, nbins=30, main="Inverse Gaussian distribution")
```




```
{r}
data.frame(row.names=c("Gamma", "Exponentional", "Log-Normal", "Weibull","Inverse Gaussian"),
 DF=c(fit.gamma$df.fit, fit.EXP$df.fit, fit.LOGNO$df.fit, fit.WEI$df.fit, fit.IG$df.fit),
 LOGLIK=c(logLik(fit.gamma),logLik(fit.EXP),logLik(fit.LOGNO),logLik(fit.WEI),logLik(fit.IG)),
 AIC=c(AIC(fit.gamma), AIC(fit.EXP), AIC(fit.LOGNO),AIC(fit.WEI), AIC(fit.IG)),
 BIC=c(fit.gamma$sbc, fit.EXP$sbc, fit.LOGNO$sbc, fit.WEI$sbc, fit.IG$sbc))
```

Description: df [5 × 4]

	DF <dbl>	LOGLIK <dbl>	AIC <dbl>	BIC <dbl>
Gamma	2	-1135.449	2274.898	2280.068
Exponentional	1	-1171.736	2345.471	2348.056
Log-Normal	2	-1127.577	2259.155	2264.325
Weibull	2	-1145.280	2294.560	2299.730
Inverse Gaussian	2	-1127.255	2258.509	2263.679

5 rows

As we can see from the above chart with Likelihood, AIC and BIC, the model inverse Gaussian fits better the distribution for PriceEuro

The mixture of distribution:

It is possible to compute a mixture of two distributions to find the best mixture, the algorithm is repeated five times.

```r
{r}
par(mfrow=c(1,3))
fit.GA.PriceEuro <- gamlssMXfits(n = 5, vehicle$PriceEuro~1, family = GA, K = 2,
data = vehicle)
par(mfrow=c(1,3))
fit.EXP.PriceEuro <- gamlssMXfits(n = 5, vehicle$PriceEuro~1, family = EXP, K = 2,
data = vehicle)
par(mfrow=c(1,3))
fit.LOGNO.PriceEuro <- gamlssMXfits(n = 5, vehicle$PriceEuro~1, family = LOGNO, K =
2, data = vehicle)
par(mfrow=c(1,3))
fit.WEI.PriceEuro <- gamlssMXfits(n = 5, vehicle$PriceEuro~1, family = WEI, K = 2,
data = vehicle)
par(mfrow=c(1,3))
fit.IG.PriceEuro <- gamlssMXfits(n = 5, vehicle$PriceEuro~1, family = IG, K = 2,
data = vehicle)
```

```r
{r}
data.frame(row.names=c("Gamma", "Exponentional", "Log-Normal", "Weibull","Inverse
Gaussian","M.Gamma", "M.Exponentional", "M.Log-Normal", "M.Weibull","M.Inverse
Gaussian" ),
 DF=c(fit.gamma$df.fit, fit.EXP$df.fit, fit.LOGNO$df.fit, fit.WEI$df.fit, fit.IG$df
.fit, fit.GA.PriceEuro$df.fit, fit.EXP.PriceEuro$df.fit, fit.LOGNO.PriceEuro$df.fit
, fit.WEI.PriceEuro$df.fit, fit.IG.PriceEuro$df.fit),
 LOGLIK=c(logLik(fit.gamma),logLik(fit.EXP),logLik(fit.LOGNO),logLik(fit.WEI
),logLik(fit.IG),logLik(fit.GA.PriceEuro),logLik(fit.EXP.PriceEuro),logLik(fit
.LOGNO.PriceEuro),logLik(fit.WEI.PriceEuro),logLik(fit.IG.PriceEuro)),
 AIC=c(AIC(fit.gamma), AIC(fit.EXP), AIC(fit.LOGNO),AIC(fit.WEI), AIC(fit.IG),AIC
(fit.GA.PriceEuro), AIC(fit.EXP.PriceEuro), AIC(fit.LOGNO.PriceEuro),AIC(fit.WEI
.PriceEuro), AIC(fit.IG.PriceEuro)),
 BIC=c(fit.gamma$sbc, fit.EXP$sbc, fit.LOGNO$sbc, fit.WEI$sbc, fit.IG$sbc,fit.GA
.PriceEuro$sbc, fit.EXP.PriceEuro$sbc, fit.LOGNO.PriceEuro$sbc, fit.WEI
.PriceEuro$sbc, fit.IG.PriceEuro$sbc))
```

Description: df [10 × 4]

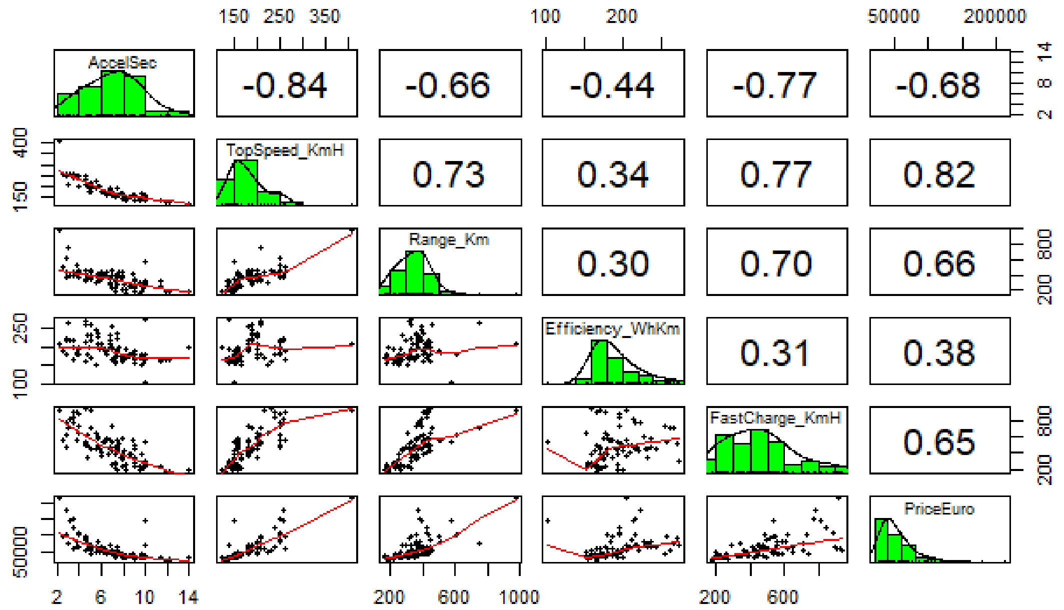
	DF <dbl>	LOGLIK <dbl>	AIC <dbl>	BIC <dbl>
Gamma	2	-1135.449	2274.898	2280.068
Exponentional	1	-1171.736	2345.471	2348.056
Log-Normal	2	-1127.577	2259.155	2264.325
Weibull	2	-1145.280	2294.560	2299.730
Inverse Gaussian	2	-1127.255	2258.509	2263.679
M.Gamma	5	-1122.393	2254.787	2267.712
M.Exponentional	3	-1171.736	2349.471	2357.226
M.Log-Normal	5	-1118.255	2246.511	2259.436
M.Weibull	5	-1127.743	2265.485	2278.410
M.Inverse Gaus...	5	-1118.151	2246.302	2259.227

Previously we saw, according to the likelihood the best-fitted distribution was the gamma model. But after applying the mixture of distribution, the best fitting is performed through a mixture distribution with the inverse Gaussian family.

3. Principal Component Analysis.

Before proceeding with the PCA, it is necessary to evaluate whether there is a correlation between the numerical variables of this dataset. Thus, the coefficients of correlation between variables are located in the upper triangle of the matrix and are used to determine whether PCA is useful or not, whereas the scatterplots of data are found in the lower triangle and the non-parametric density of the data are located on the main diagonal, both are used to determine whether CA is useful or not.

```
{r}
library(psych)
vehicle_pca <- vehicle[,3:8]
plot <- pairs.panels(vehicle_pca, hist.col="green", density = TRUE, ellipses =
FALSE)
```



In particular, if we look at the plot, we can see that there is a correlation found among most of the variables, as their values of coefficients of correlation are greater than zero. The highest coefficients of correlation in this data are 0.73 and 0.65 and the least coefficient of correlation is -0.87 only variable. So we can say that for our data we can apply Principal Component Analysis.

```
```{r}
apply(vehicle_pca, 2, mean)
```
```

```
       AccelSec    TopSpeed_KmH        Range_Km Efficiency_WhKm  FastCharge_KmH       PriceEuro 
       7.046939      181.653061      350.153061      189.867347      456.734694    57324.683673 
```

```
```{r}
apply(vehicle_pca, 2, var)
```
```

```
       AccelSec    TopSpeed_KmH        Range_Km Efficiency_WhKm  FastCharge_KmH       PriceEuro 
   6.169733e+00    1.870435e+03    1.397549e+04    9.033534e+02    4.050675e+04    1.175684e+09 
```

The variables are very different from each other. Now it is better to standardize the variable to have zero mean and unitary variance.

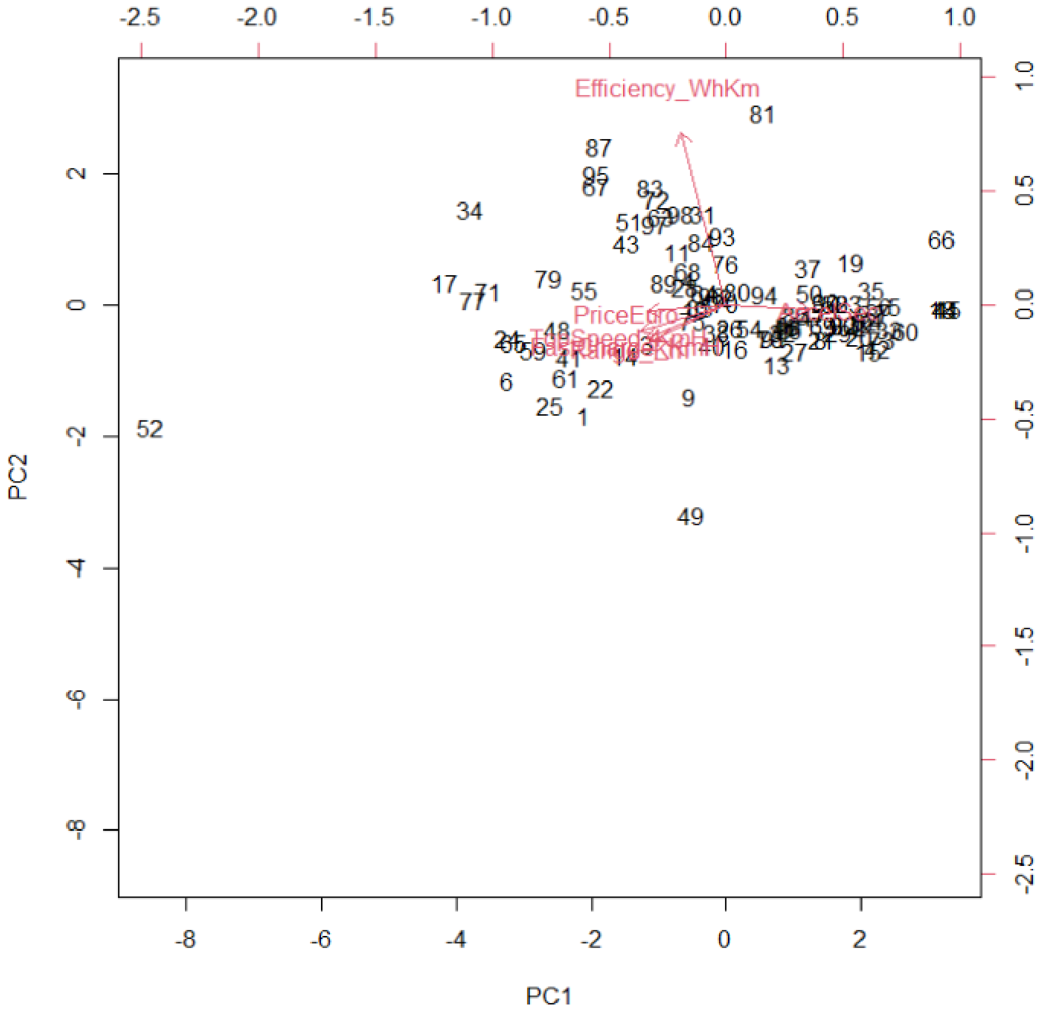
3.1. Computing PCs:

```
{r}
pr.out = prcomp(vehicle[,3:8], scale=TRUE)
pr.out$rotation[,1:4]
```

```
                     PC1         PC2         PC3         PC4
AccelSec       0.4434977 -0.02070974  0.35262586  0.37843399
TopSpeed_KmH  -0.4600929 -0.14547133  0.14699181 -0.26489525
Range_Km      -0.4099149 -0.19978606  0.03910994  0.85554567
Efficiency_WhKm -0.2462651  0.95016431 -0.04036327  0.13418077
FastCharge_KmH -0.4271999 -0.18552572 -0.54587668 -0.03250404
PriceEuro     -0.4247915 -0.03553483  0.74357803 -0.18867655
```

According to the above PC computing, We scaled our data to compute 4 PCs using prcomp. After analysis, it shows us a minimum of 2 and a maximum of 3 PCs will be enough.

3.2. Biplot



We can also show in a 2D graphical representation, that is biplot, the dataset on the cartesian space spanned by the first two principal components. In this graphic, the arrows represent the original variables (PC loadings) and the points are sample units (PC scores). The angle between the variables reflects the correlation matrix (low angles imply a high positive correlation such as TopSpeed and Range_Km, 90° degrees angles imply the absence of correlation, and near by 180° degrees imply a perfect negative correlation such as AccelSec and TopSpeed).

3.3. Selecting the number of principal components

To select the number of principal components, three heuristic methods are proposed.

3.3.1. CPVE – Cumulative proportion of variance explained

According to this approach, the first q principal components that explain at least 80% of the total variance are retained.

```
{r}
(PVE <- vehicle_eigen$values/sum(vehicle_eigen$values))
PVE <- round(PVE,3)
```

```
[1] 0.687 0.137 0.062 0.060 0.037 0.017
```

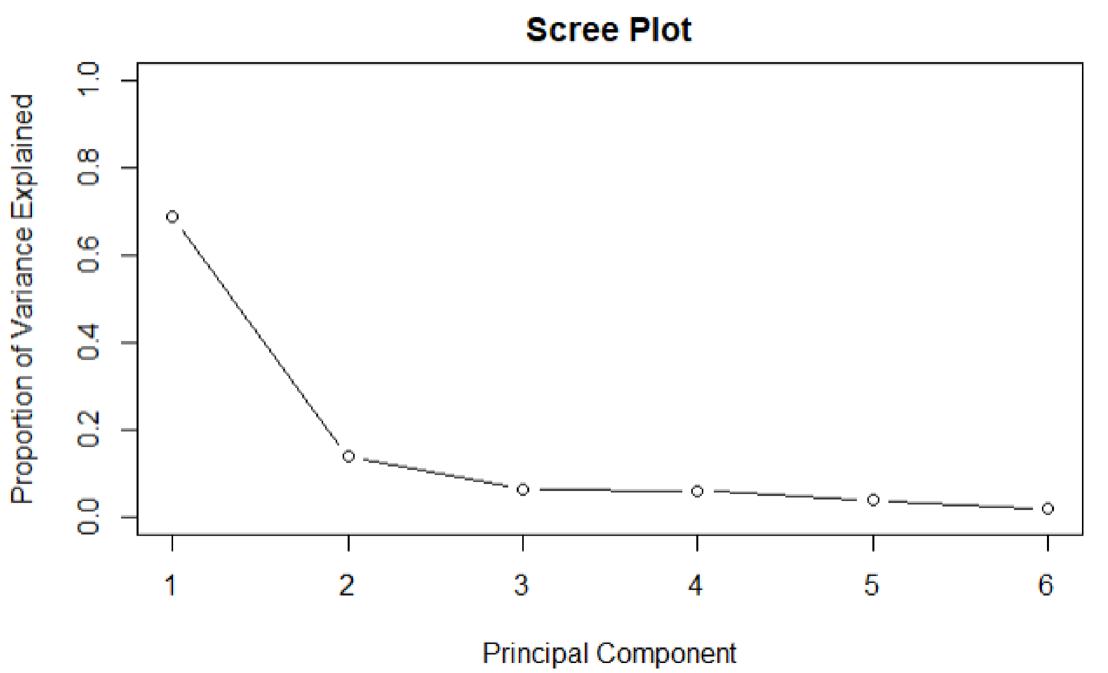
```
{r}
cumsum(PVE)
```

```
[1] 0.687 0.824 0.886 0.946 0.983 1.000
```

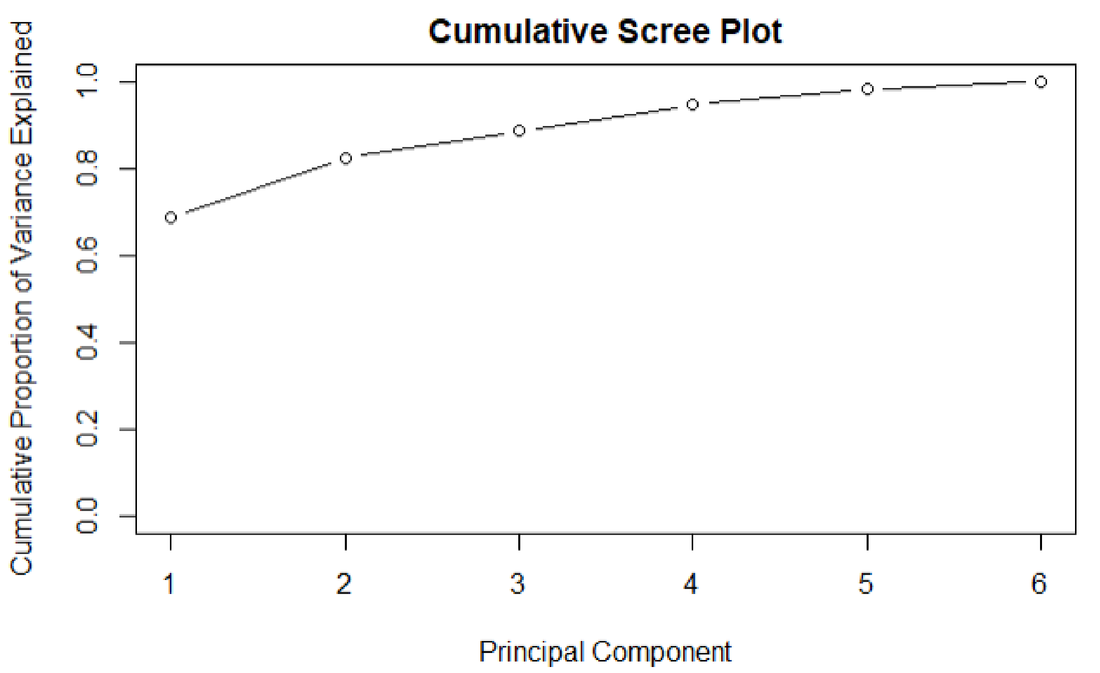
According to the method, the first two principal components are retained because together they are 82% of the total variance.

3.3.2. Scree Plot

```
{r}
plot(PVE, xlab="Principal Component", ylab="Proportion of Variance Explained",
main="Scree Plot", ylim=c(0,1), type='b')
```




```
{r}
plot(cumsum(PVE), xlab="Principal Component", main="Cumulative Scree Plot",
ylab="Cumulative Proportion of Variance Explained", ylim=c(0,1),type='b')
```



In the “Proportion of variance Explained” plot, the elbow point is not clear and it may be at q=2. it seems reasonable to retain the first two principal components.

3.3.3. Kaiser's rule

According to Kaiser's rule, only the first principal components can represent the phenomena: this heuristic method says, for standardized data, to keep as many PCs as are the ones with variance greater than 1:

```
{r}
vehicle_eigen$values[vehicle_eigen$values > 1]

 [1] 4.121681
```

Kaiser's rule suggests the first principal component

3.4. PCA Result

The cumulative PVE rule suggesting to retain the first two Principal components. While Kaiser's rule suggests retaining the only first principal component. However, the scree plot suggests retaining the first two. So the reduction from d=6 to 2PCs, while explaining 82% of the total variability, is a good compromise.

4. Cluster Analysis.

The purpose of performing cluster analysis is to identify the pattern of similar units with the vehicle dataset. In simple words, the clustering analysis's goal is to segregate groups with similar traits and assign them into clusters. There are some methods to achieve this goal. The first step is to calculate a certain type of distance between pairs of units and build the distance matrix. We do this because the concept of dissimilarity is fundamental in this type of analysis since the units most "similar" to each other will be grouped within the same cluster, while between the different clusters there must be a high dissimilarity. Let us compute the Euclidean distance:

```
```{r}
dist.eucl <- dist(vehicle_cluster_scaled, method = "euclidean")
eucl <- round(as.matrix(dist.eucl)[1:6, 1:6], 2)
rownames(eucl) <- c("AccelSec","TopSpeed_KmH","Range_Km","Efficiency_WhKm","FastCharge_KmH","PriceEuro
")
colnames(eucl) <- c("AccelSec","TopSpeed_KmH","Range_Km","Efficiency_WhKm","FastCharge_KmH","PriceEuro
")
eucl

```
```

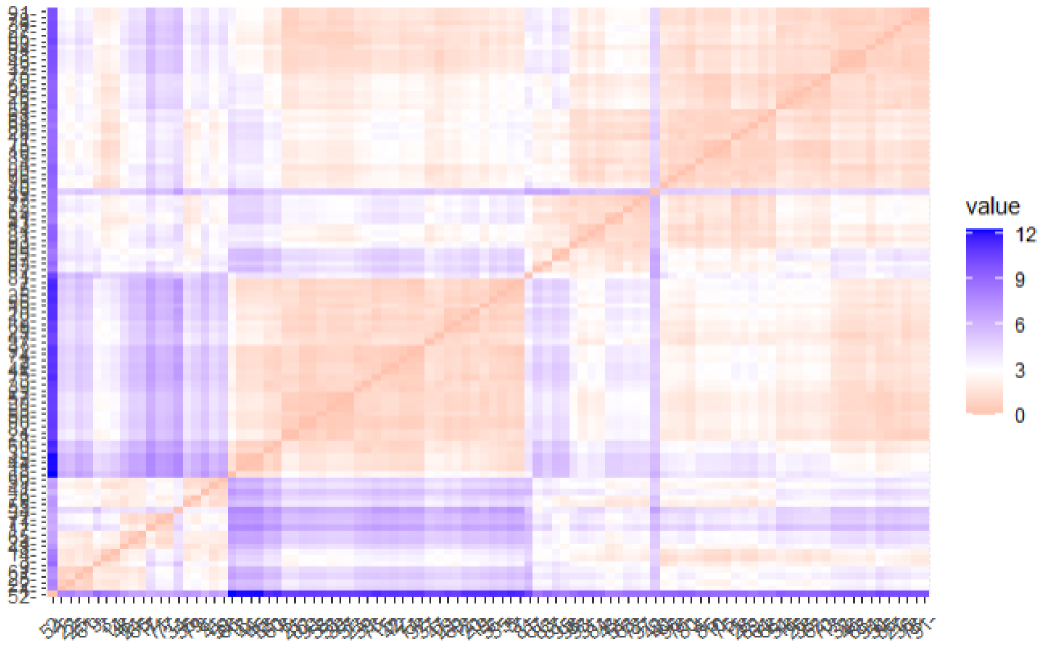
	AccelSec	TopSpeed_KmH	Range_Km	Efficiency_WhKm	FastCharge_KmH	PriceEuro
AccelSec	0.00	4.72	1.85	2.97	5.29	2.74
TopSpeed_KmH	4.72	0.00	3.36	2.78	0.99	5.42
Range_Km	1.85	3.36	0.00	1.49	3.87	2.57
Efficiency_WhKm	2.97	2.78	1.49	0.00	3.23	3.42
FastCharge_KmH	5.29	0.99	3.87	3.23	0.00	6.01
PriceEuro	2.74	5.42	2.57	3.42	6.01	0.00

According to this distance matrix, relative to a subset of data, the most distant observations are TopSpeed_KmH and FastCHarge_KmH, while the most similar ones are PriceEuro and FastCharge_KmH. We can also try to compute the Manhattan distance:

	AccelSec	TopSpeed_KmH	Range_Km	Efficiency_WhKm	FastCharge_KmH	PriceEuro
AccelSec	0.00	9.76	3.28	6.62	10.99	6.14
TopSpeed_KmH	9.76	0.00	7.46	6.46	1.81	12.31
Range_Km	3.28	7.46	0.00	3.35	8.63	4.92
Efficiency_WhKm	6.62	6.46	3.35	0.00	7.63	7.58
FastCharge_KmH	10.99	1.81	8.63	7.63	0.00	13.48
PriceEuro	6.14	12.31	4.92	7.58	13.48	0.00

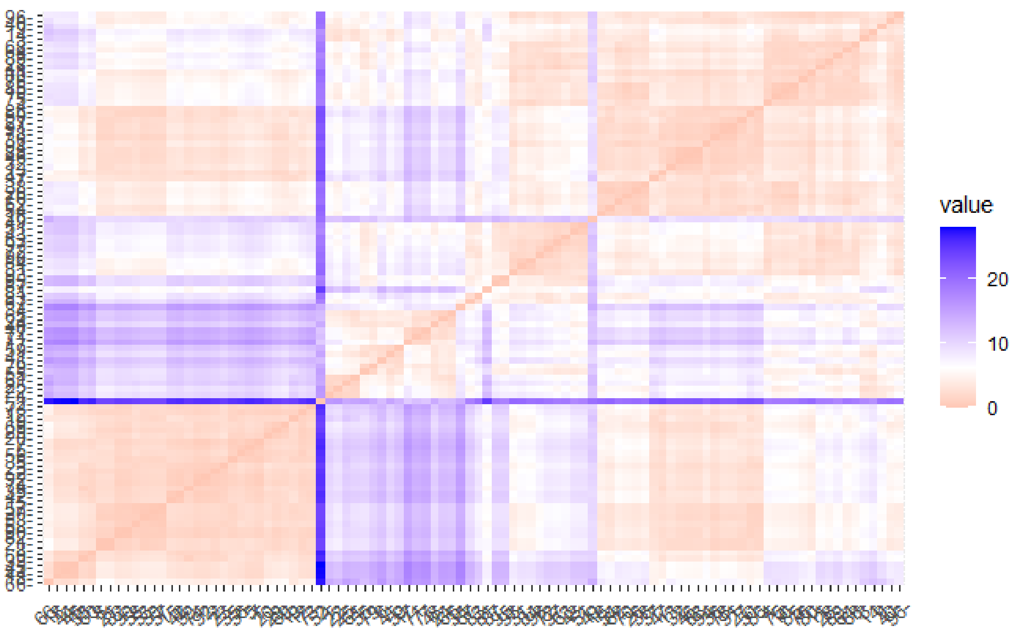
Also according to this distance matrix, the most distant observations are TopSpeed_KmH and FastCHarge_KmH, while the most similar are PriceEuro and FastCharge_KmH. Having obtained the same results, we can say that the dataset probably does not contain outliers ("probably", because these results are relative to a subset of data).

```
{r}
fviz_dist(dist.eucl)
```



This is the distance matrix, based on the Euclidean distance. The level of its colours is proportional to the value of the dissimilarity between observations: red stands for high similarity; blue indicates low similarity. According to the Visual method, which uses the Euclidean distance as the distance between units, the data should not contain a noticeable clustering structure. While not sure of the presence of a clustering structure, the analysis continues.

```
{r}
fviz_dist(dist.man)
```



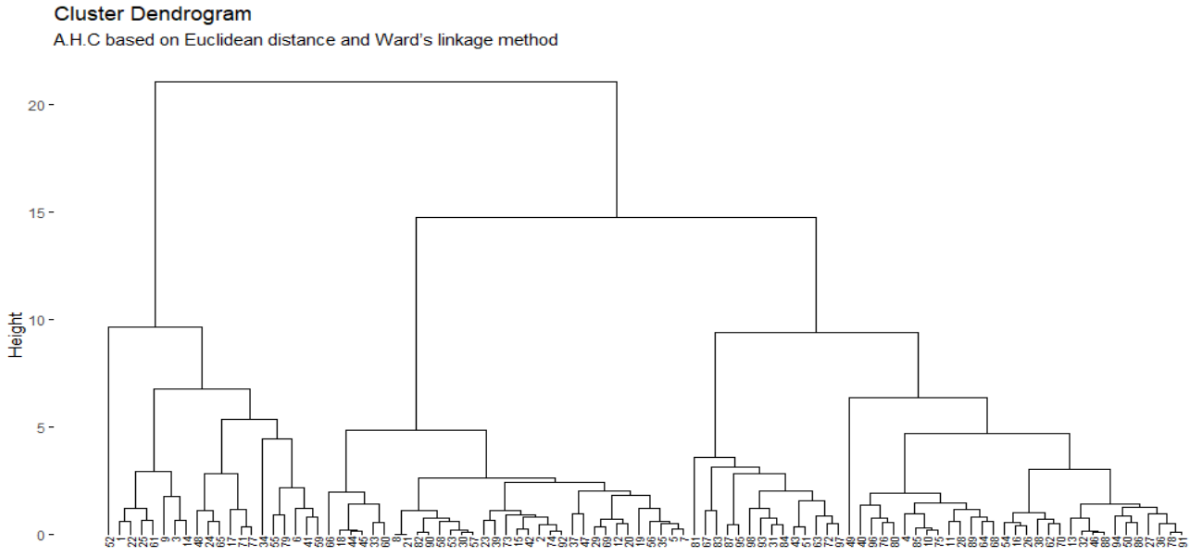
Now, we can try some clustering algorithms.

4.1. Agglomerative Hierarchical Clustering.

4.1.1. A.H.C based on Euclidean distance and Ward's linkage method.

res.hc <- hclust(d = dist.eucl, method = "ward.D2")

fviz_dend(res.hc, cex = 0.5,subtitle = "A.H.C based on Euclidean distance and Ward's linkage method")



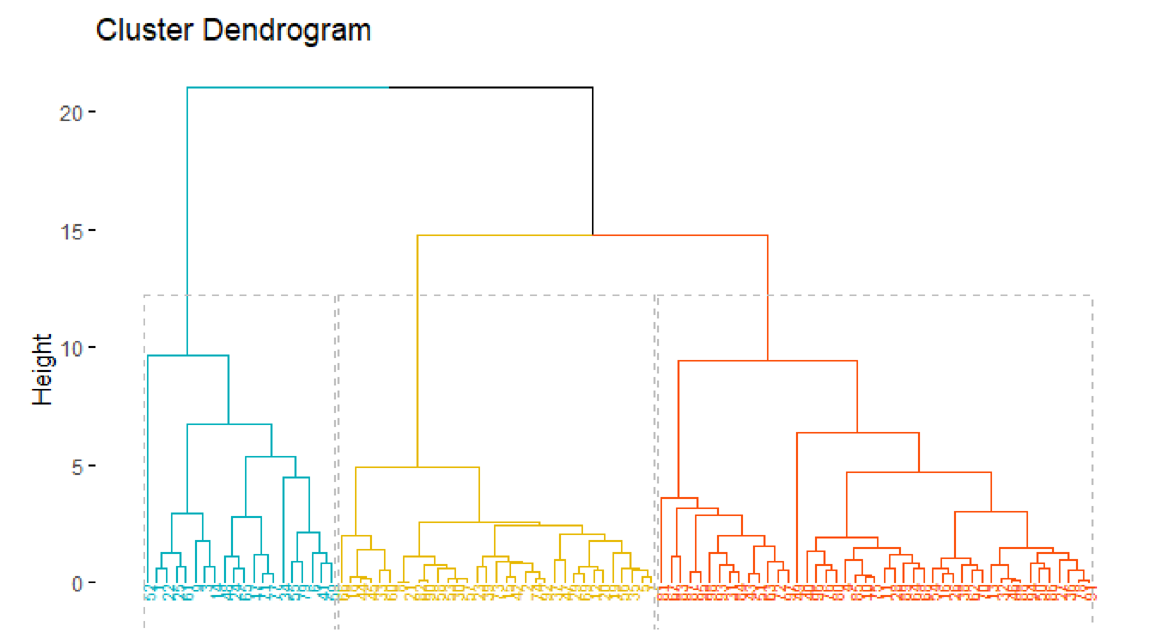
We have to look at the height in correspondence of which these units are merged together for the first time (minimum height). This is explained by the cophenetic distance, which is compared to the original one, to understand if the clustering method is good or not: it is good if the cophenetic distance between units in the cluster tree is well correlated to the distance between units in the original distance matrix.

```
{r}
res.coph <- cophenetic(res.hc)
cor(dist.eucl, res.coph)

[1] 0.6289154
```

The result is 0.62, which is not such a good result. This means that this clustering method does not preserve the true original distances between units very well. Now, if we want to see clusters, we have to cut the hierarchical tree, for example specifying the number of groups that we want.

```
{r}
grp <- cutree(res.hc, k = 3)
fviz_dend(res.hc, k = 3, cex = 0.5, k_colors = c("#00AFBB", "#E7B800", "#FC4E07"),
color_labels_by_k = TRUE, rect = TRUE)
```



We can see that each cluster is identified by a specific colour. Moreover, the last cluster is the bigger one, as we can see from this R code:

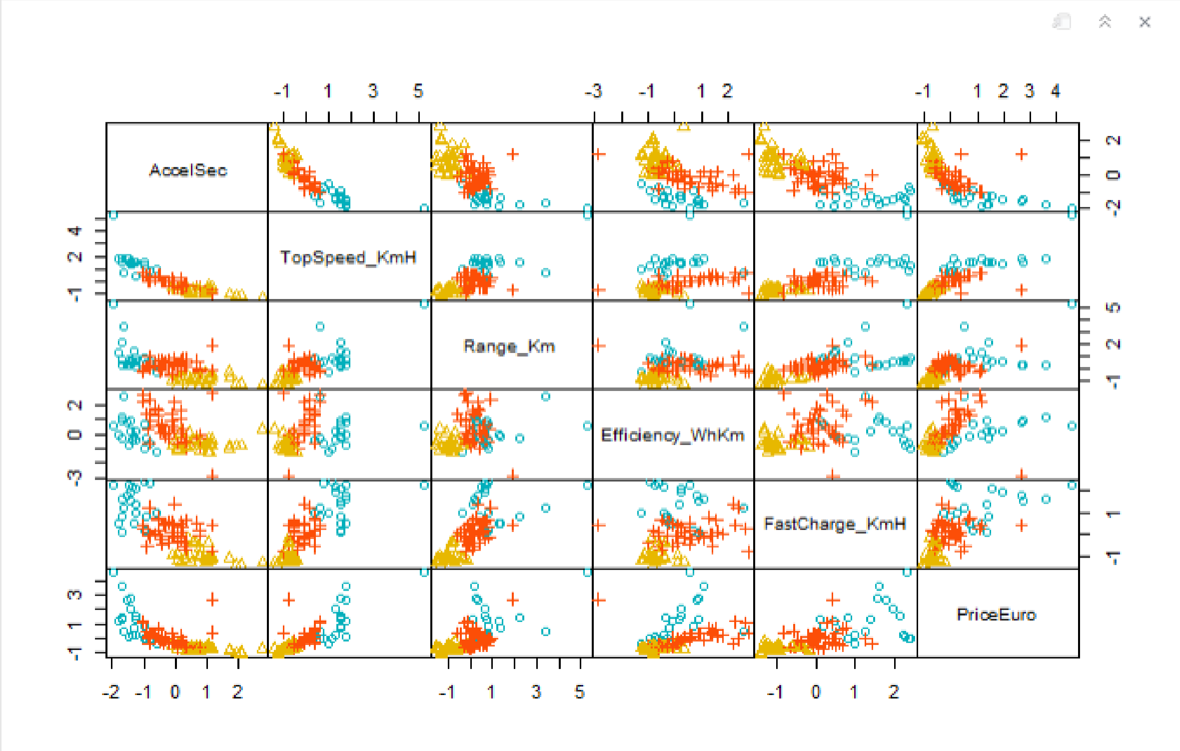
```
{r}
table(grp)
```

```
grp
 1  2  3
20 33 45
```

Cluster 3 contains more units than clusters 1 and 2, which contain 20 and 33 units respectively.

We can visualize the clustering results in the original space via the matrix of pairwise scatterplots:

```{r}
pairs(vehicle_cluster_scaled, gap=0, pch=grp, col=c("#00AFBB", "#E7B800", "#FC4E07")[grp])
```

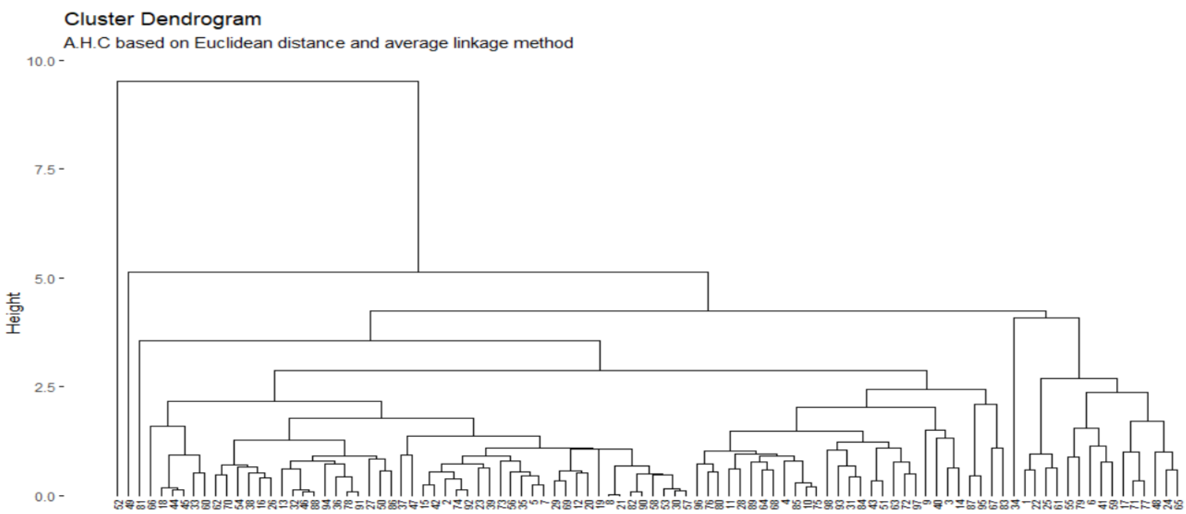


now we can see not only the clustering structure but also the different clusters, identified by the different colours used by a hierarchical clustering algorithm.

4.1.2. A.H.C based on Euclidean distance and Average linkage method.

res.hc2 <- hclust(d = dist.eucl, method = "average")

fviz_dend(res.hc2, cex = 0.5,subtitle = "A.H.C based on Euclidean distance and average linkage method")



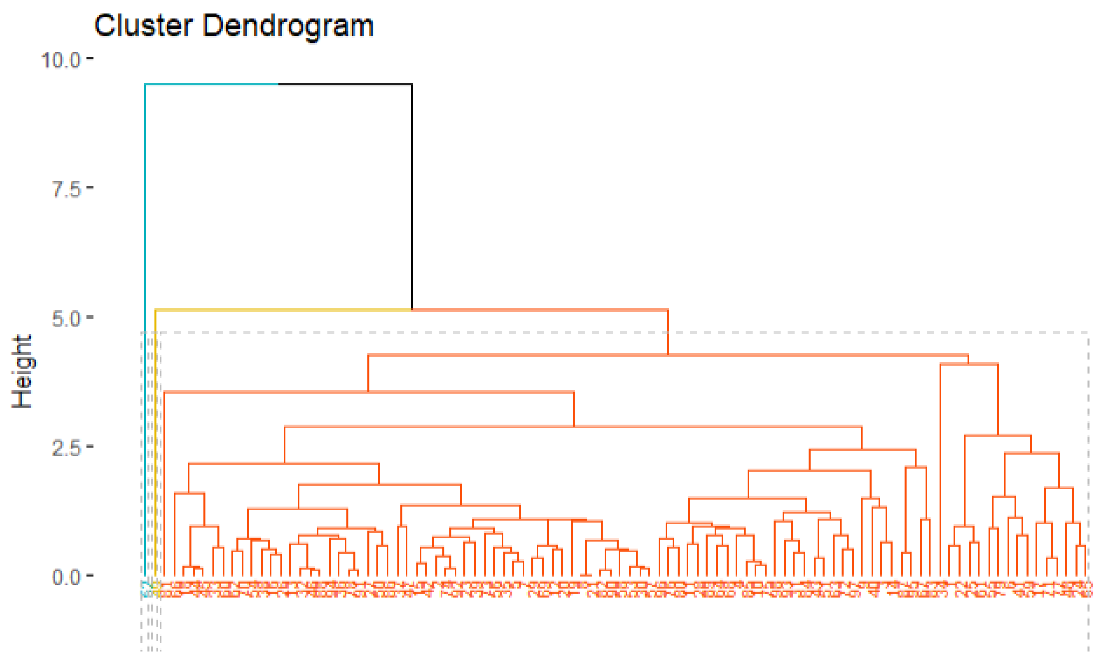
```
{r}
res.coph2 <- cophenetic(res.hc2)
cor(dist.eucl, res.coph2)
```

```
[1] 0.8450374
```

The result is 0.84, which is very good. So, this clustering method preserves the true original distances between units quite well, however, better than the previous one, with Ward's linkage method.

Now, to see clusters, we have to cut the hierarchical tree, for example specifying the number of groups that we want.

```
{r}
grp <- cutree(res.hc2, k = 3)
fviz_dend(res.hc2, k = 3, cex = 0.5, k_colors = c("#00AFBB", "#E7B800", "#FC4E07"),
color_labels_by_k = TRUE, rect = TRUE)
```



We can see that almost all of the values shifted to cluster 1 which was previously in clusters 2 and 3. Cluster 1 is bigger than the other two.

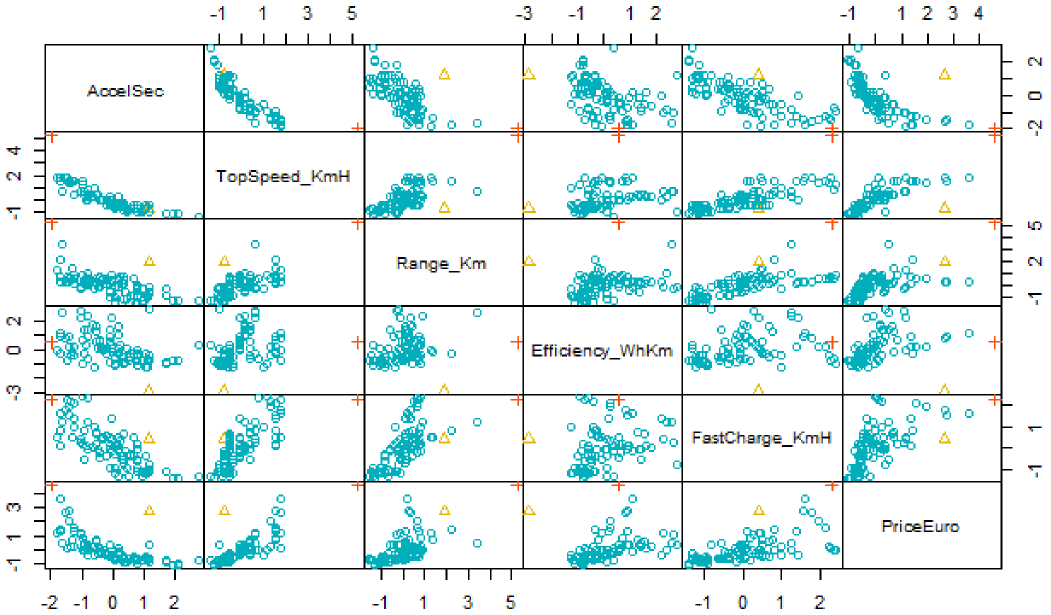
```
{r}
table(grp)
```

```
grp
 1  2  3
96  1  1
```

Cluster 1 contains more units than clusters 2 and 3, which contain 2 and 3 units respectively.

We can visualize the clustering results in the original space via the matrix of pairwise scatterplots:

```
{r}
pairs(vehicle_cluster_scaled, gap=0, pch=grp, col=c("#00AFBB", "#E7B800", "#FC4E07")[grp])
```

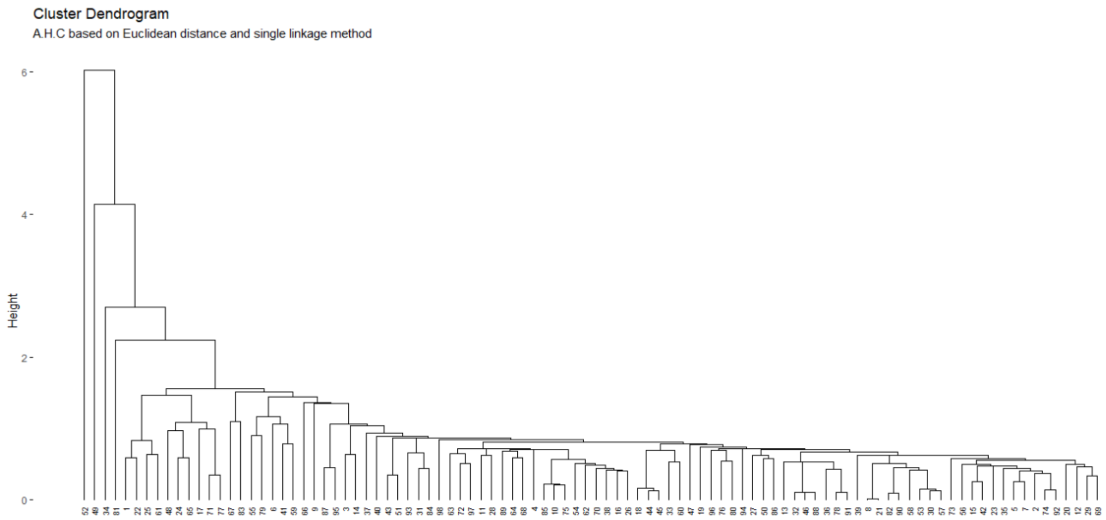


This is the original matrix of scatterplots which carries on the same observations as one of the previous algorithms, with the major difference, determined by the argument "grp" of the “pairs “ function, an argument based on the hierarchical algorithm with the average linkage method - in this case, it creates only big cluster and two small clusters with single data.

4.1.3. A.H.C based on Euclidean distance and Single linkage method.

res.hc3 <- hclust(d = dist.eucl, method = "single")

fviz_dend(res.hc3, cex = 0.5,subtitle = "A.H.C based on Euclidean distance and single linkage method")



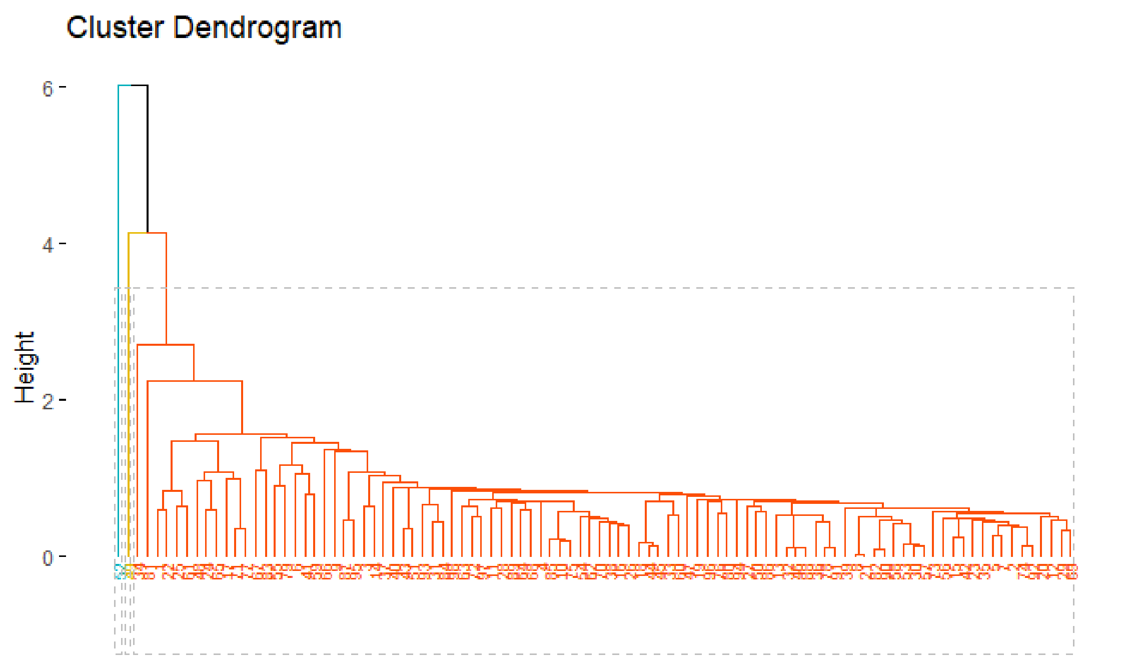
Even just looking at the dendrogram we understand that the distribution of data is fairly ambiguous. So, let's check for the correlation between the cophenetic distance and the original one:

```
{r}
res.coph3 <- cophenetic(res.hc3)
cor(dist.eucl, res.coph3)

[1] 0.7576896
```

The result is 0.75 which is a good one. This means that this clustering method does preserve the true original distances between units.

```
{r}
grp <- cutree(res.hc3, k = 3)
fviz_dend(res.hc3, k = 3, cex = 0.5, k_colors = c("#00AFBB", "#E7B800", "#FC4E07"),
color_labels_by_k = TRUE, rect = TRUE)
```



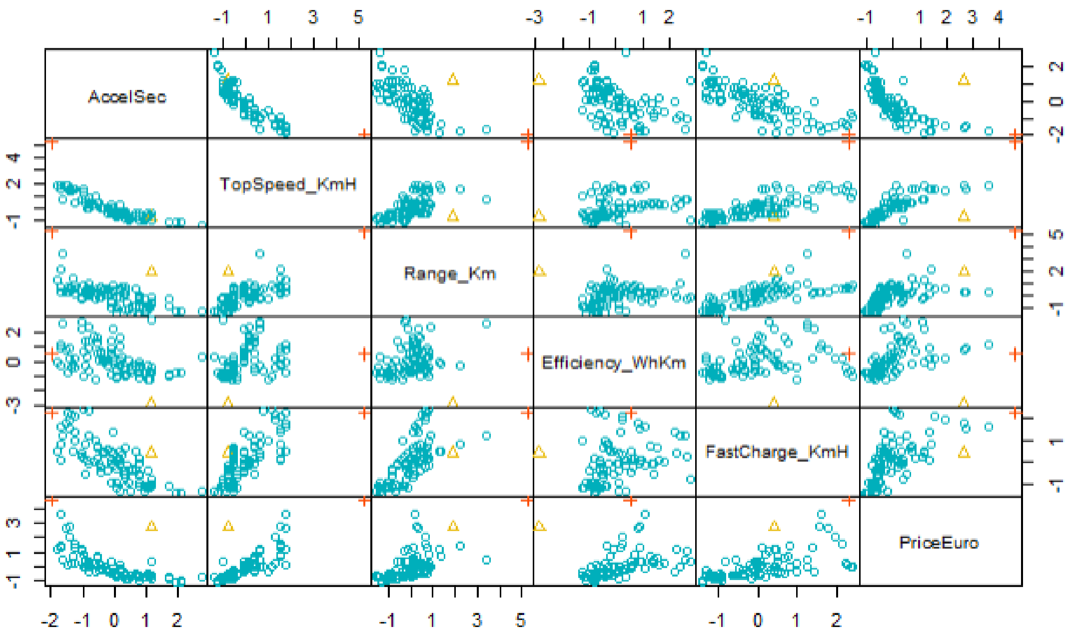
As we can see, almost all units are grouped in a single cluster and we can check this also through the R code which gives us the size of the cluster:

```
{r}
table(grp)

grp
 1  2  3
96  1  1
```

As per cophenetic distance, it was a good one but this is evident also in the clustering results in the original space:

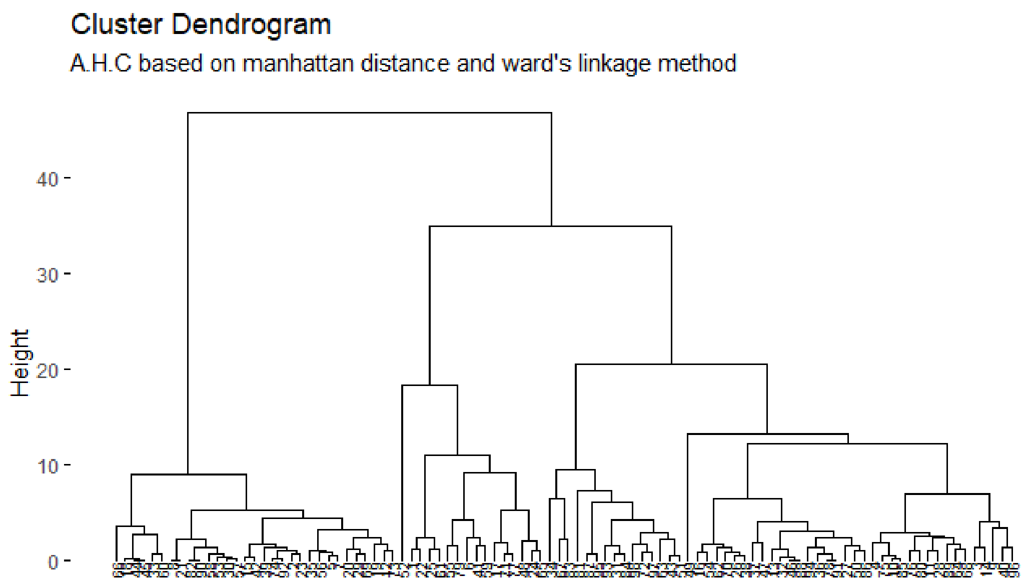
```r
{r}
pairs(vehicle_cluster_scaled, gap=0, pch=grp, col=c("#00AFBB", "#E7B800", "#FC4E07"
)[grp])
```



In each pairwise of the scatterplot, all the observations are the ones of the first cluster, plus the two of the other two clusters.

4.1.4. A.H.C based on Manhattan distance and Ward's linkage method.

```r
{r}
res.hc4 <- hclust(d = dist.man, method = "ward.D2")
fviz_dend(res.hc4, cex = 0.5,subtitle = "A.H.C based on manhattan distance and
ward's linkage method")
```



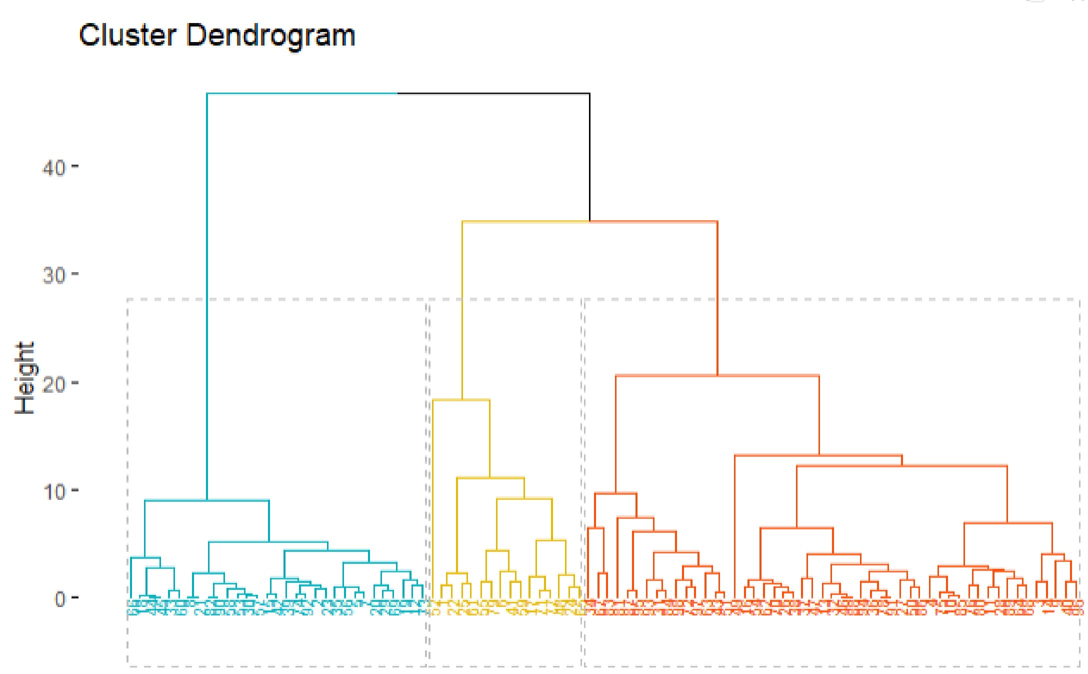
```{r}
res.coph4 <- cophenetic(res.hc4)
cor(dist.man, res.coph4)
```

```
[1] 0.5123185
```

The correlation between the two distances is not so high: 0.51. This means that this clustering method does not preserve the true original distances between units very well.

Now, let's see the clusters:

```{r}
grp <- cutree(res.hc4, k = 3)
fviz_dend(res.hc4, k = 3, cex = 0.5, k_colors = c("#00AFBB", "#E7B800", "#FC4E07"),
color_labels_by_k = TRUE, rect = TRUE)
```



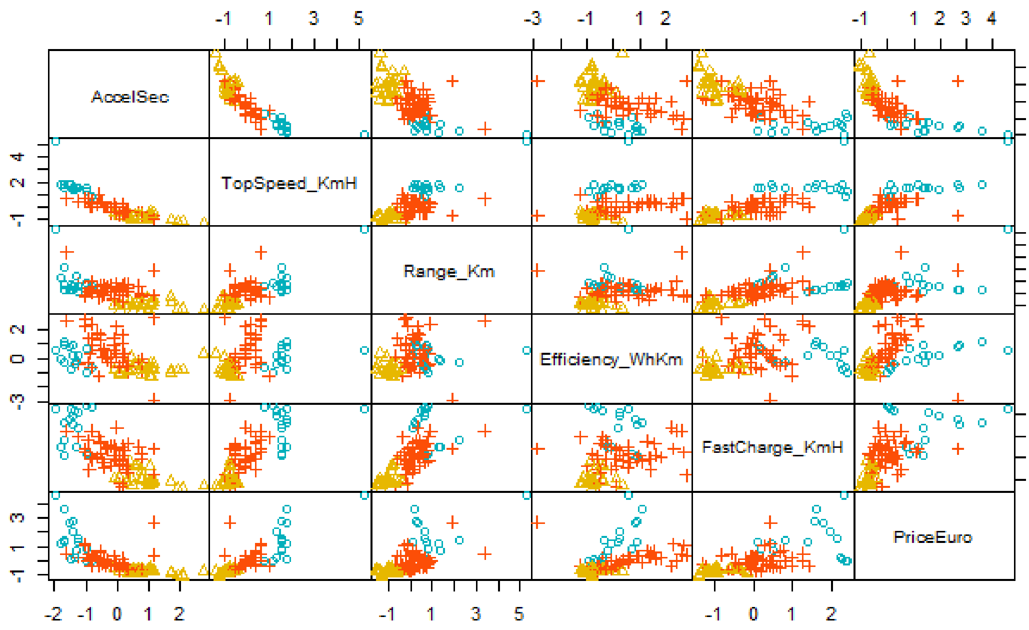
Here we have 3 clusters, underlined by 3 different colours. As regards the size of the cluster:

```{r}
table(grp)
```

```
grp
 1  2  3
16 31 51
```

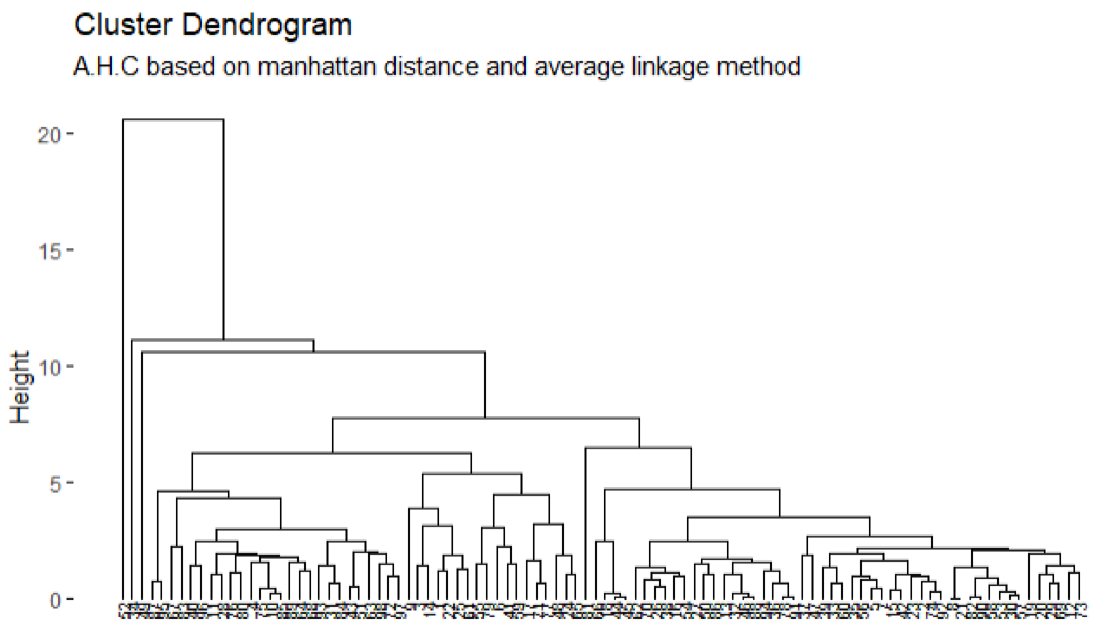
In this case, the biggest cluster is the third one, which contains 51 units.

```
{r}
pairs(vehicle_cluster_scaled, gap=0, pch=grp, col=c("#00AFBB", "#E7B800", "#FC4E07")[grp])
```



4.1.5. A.H.C based on Manhattan distance and the Average linkage method.

```
{r}
res.hc5 <- hclust(d = dist.man, method = "average")
fviz_dend(res.hc5, cex = 0.5,subtitle = "A.H.C based on manhattan distance and average linkage method")
```



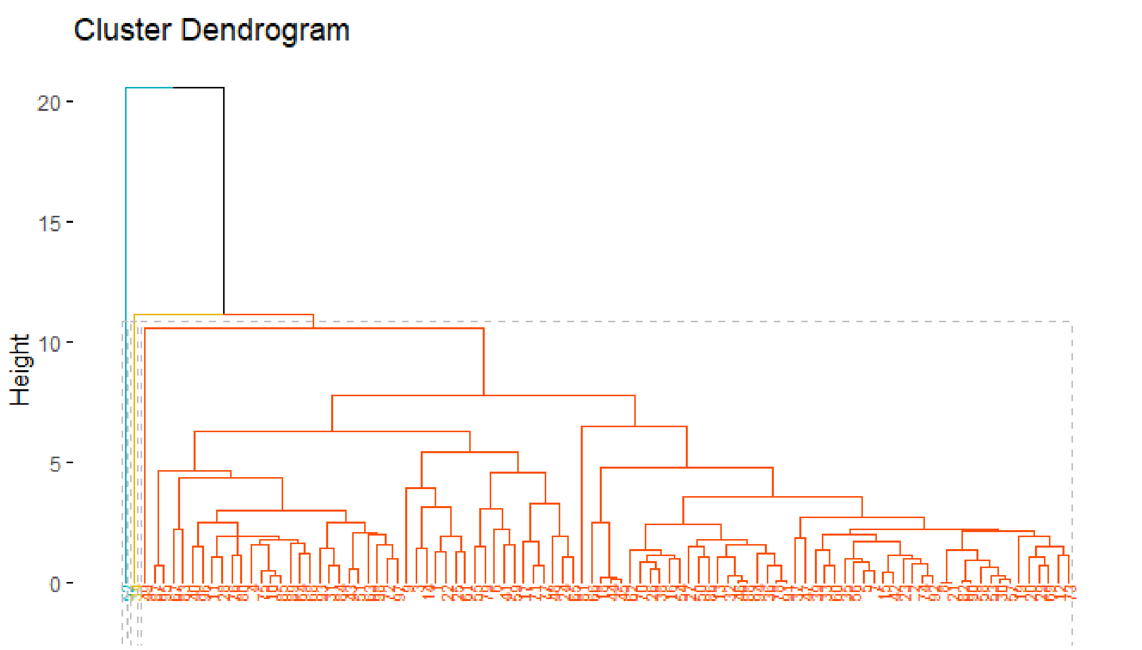
Let's check the correlation between the cophenetic distance and the original one:

```
{r}
res.coph5 <- cophenetic(res.hc5)
cor(dist.man, res.coph5)

 [1] 0.7956087
```

The result is 0.79, which is very good enough. So, this clustering method preserves the true original distances between units quite well.

```
{r}
grp <- cutree(res.hc5, k = 3)
fviz_dend(res.hc5, k = 3, cex = 0.5, k_colors = c("#00AFBB", "#E7B800", "#FC4E07"),
color_labels_by_k = TRUE, rect = TRUE)
```



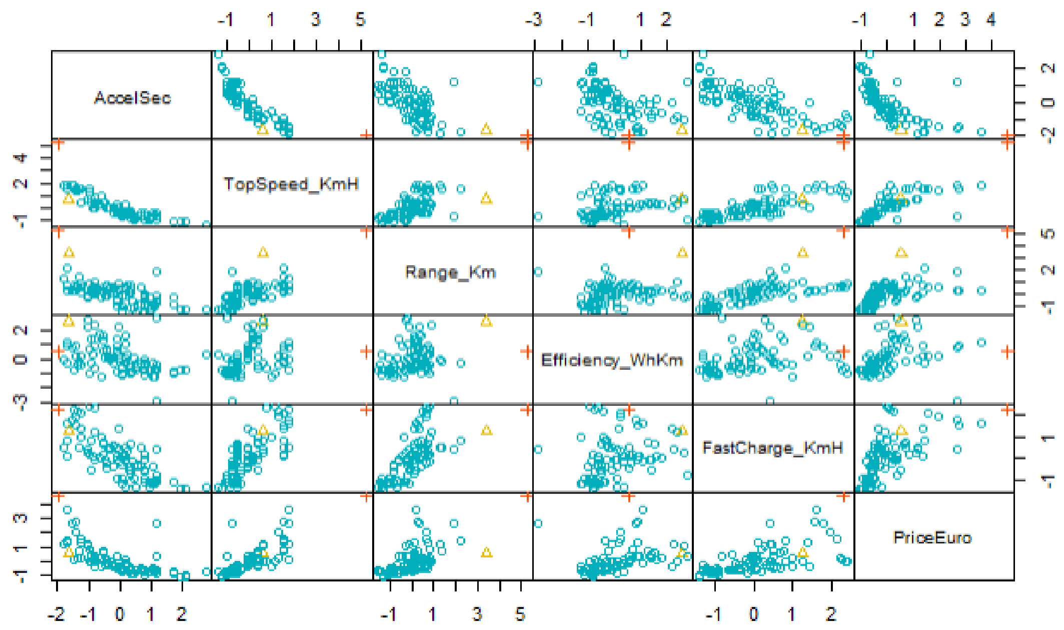
As we can see, almost all units are grouped in a single cluster and we can check this also through the R code which gives us the size of the cluster:

```
{r}
table(grp)

 grp
  1  2  3
 96  1  1
```

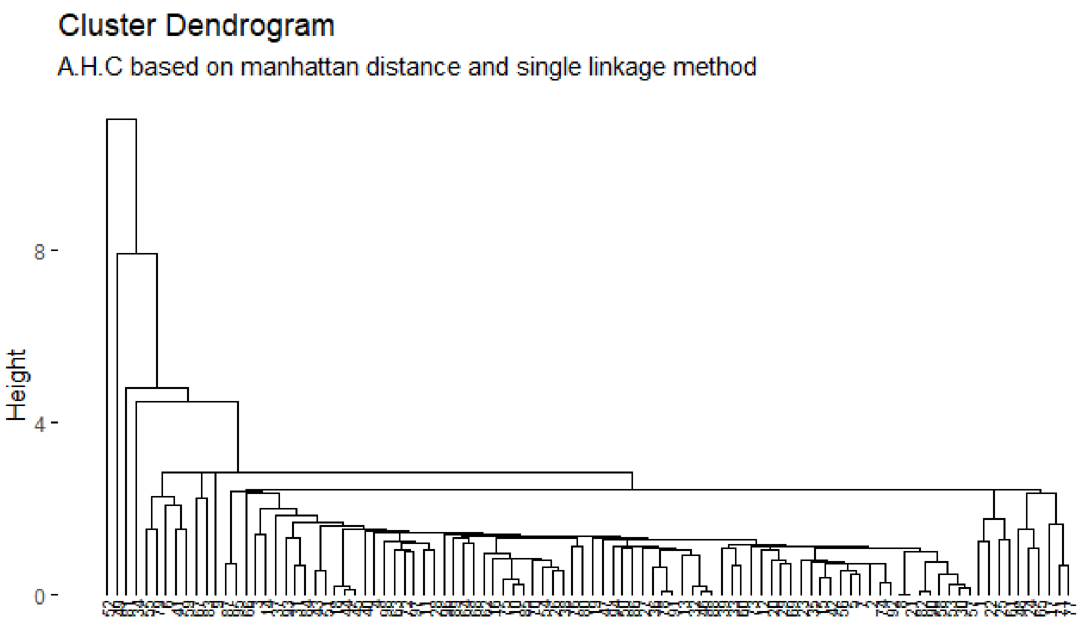
As per cophenetic distance, it was a good one but this is evident also in the clustering results in the original space:

```
{r}
pairs(vehicle_cluster_scaled, gap=0, pch=grp, col=c("#00AFBB", "#E7B800", "#FC4E07"
)[grp])
```



4.1.6. A.H.C based on Manhattan distance and the Single linkage method.

```
{r}
res.hc6 <- hclust(d = dist.man, method = "single")
fviz_dend(res.hc6, cex = 0.5,subtitle = "A.H.C based on manhattan distance and
single linkage method")
```



As we already know, the proximity of two units along the horizontal axis can not be used as a criterion for their similarity. For this purpose, we have to use the cophenetic distance e we have to compare it to the original one, to understand if the algorithm is good.

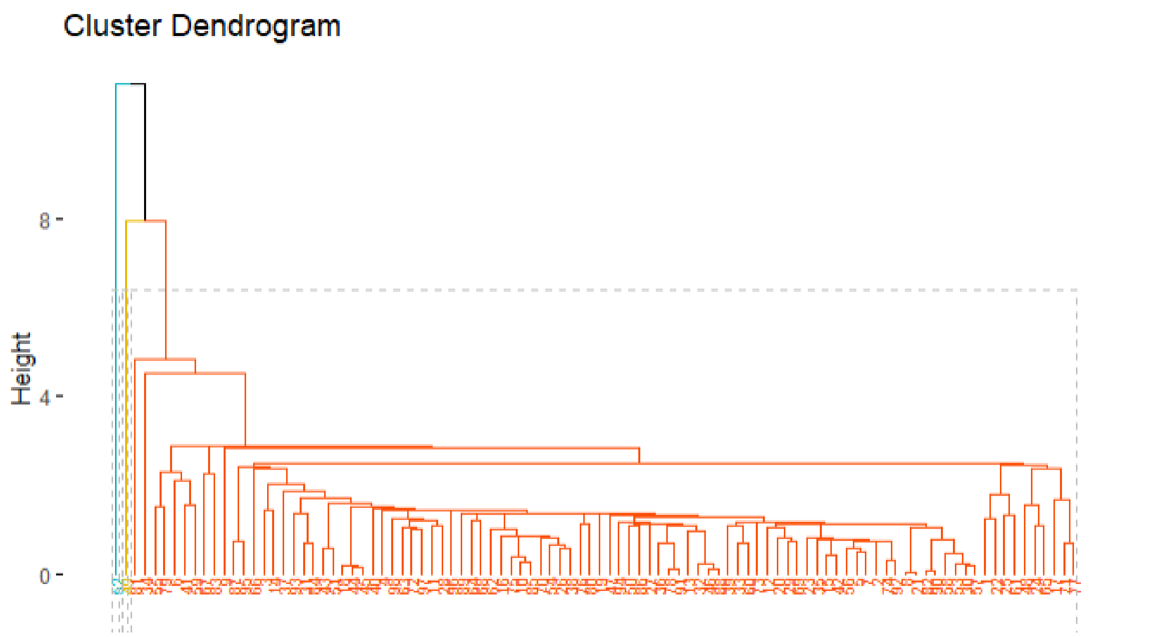
```
{r}
res.coph6 <- cophenetic(res.hc6)
cor(dist.man, res.coph6)

 [1] 0.6898203
```

The result is 0.68, which is not good.

Checking for clusters, we find out that the data partition is not good:

```
{r}
grp <- cutree(res.hc6, k = 3)
fviz_dend(res.hc6, k = 3, cex = 0.5, k_colors = c("#00AFBB", "#E7B800", "#FC4E07"),
color_labels_by_k = TRUE, rect = TRUE)
```



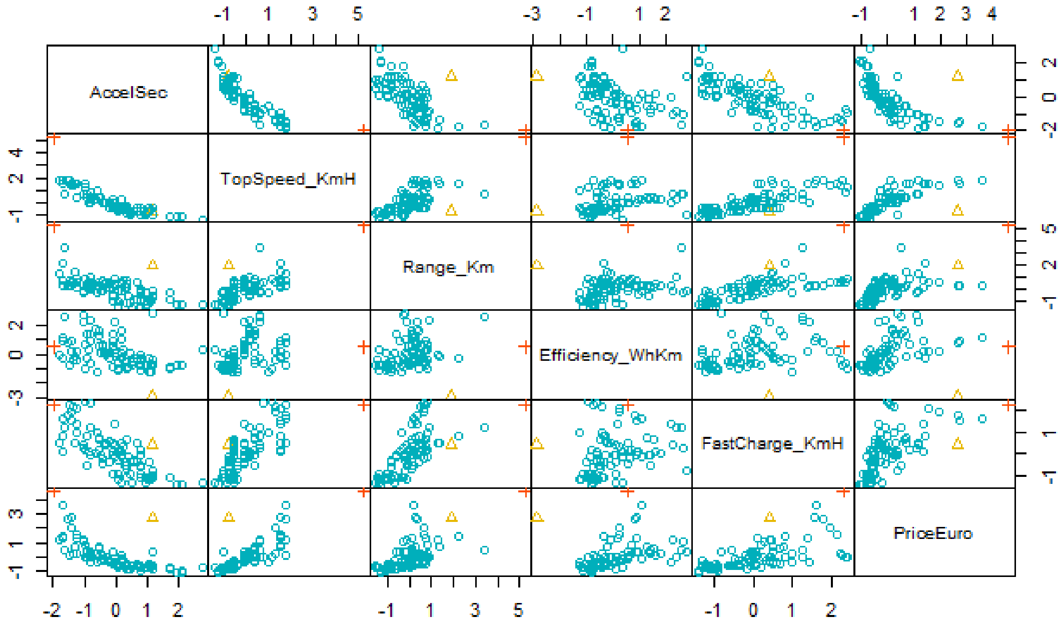
Almost all units are grouped in a single cluster and we can see this also through the R code which gives us the size of the cluster:

```
{r}
table(grp)

 grp
  1  2  3
 96  1  1
```

Moreover, if we look at the clustering results in the original space, we can see that in each pairwise of the scatterplot all the observations are the ones of the first cluster, plus the two of the other two clusters:

```r
{r}
pairs(vehicle_cluster_scaled, gap=0, pch=grp, col=c("#00AFBB", "#E7B800", "#FC4E07")[grp])
```



After trying all these different combinations, we can conclude that the best combination is the one with the Euclidean distance and the Average linkage method. In fact, if we look at the results of the correlation between the cophenetic distance and the original one for each combination, we see that the highest result was obtained precisely for the above combination.

4.2. Partitioning Clustering.

4.2.1. K-means; k=2.

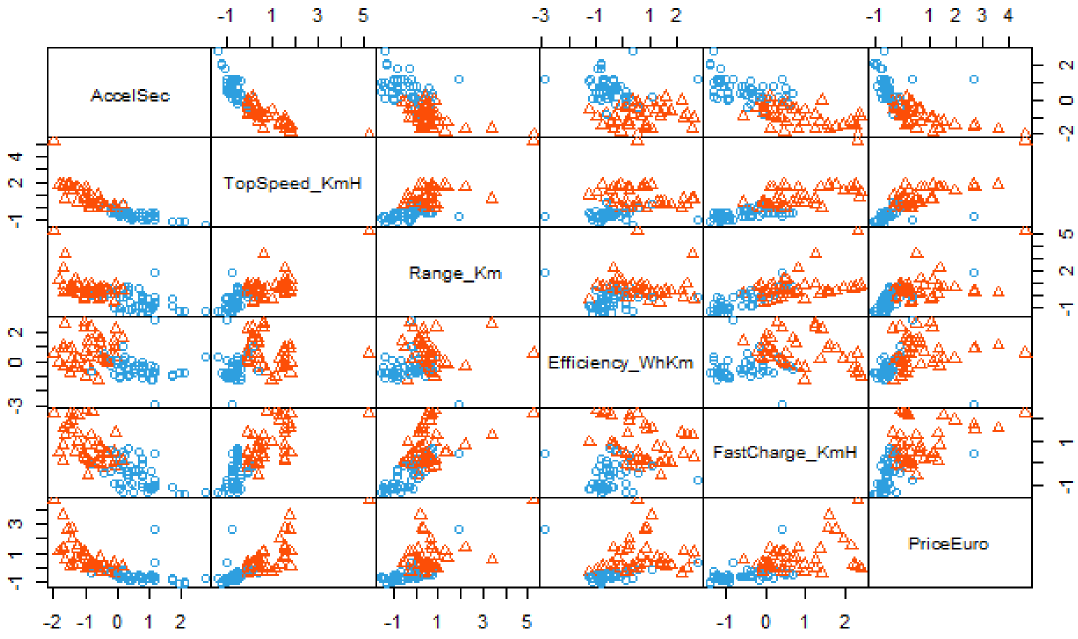
```r
{r}
km.res <- kmeans(vehicle_cluster_scaled, 2, nstart = 25)
km.res$size
```

```
[1] 56 42
```

Running this k-means algorithm we obtain two clusters: the first is the bigger one, containing 56 units, while the second is the smallest with 42 units.

We can visualize the cluster results in the original space:

```
{r}
cl <- km.res$cluster
pairs(vehicle_cluster_scaled, gap=0, pch=cl, col=c("#2E9FDF", "#FC4E07")[cl])
```



From this plot, we can understand what variables are useful to find clusters. But we can see that the scatterplot contains overlapping in all variables. Due to this, it is more difficult for the algorithm to find clusters.

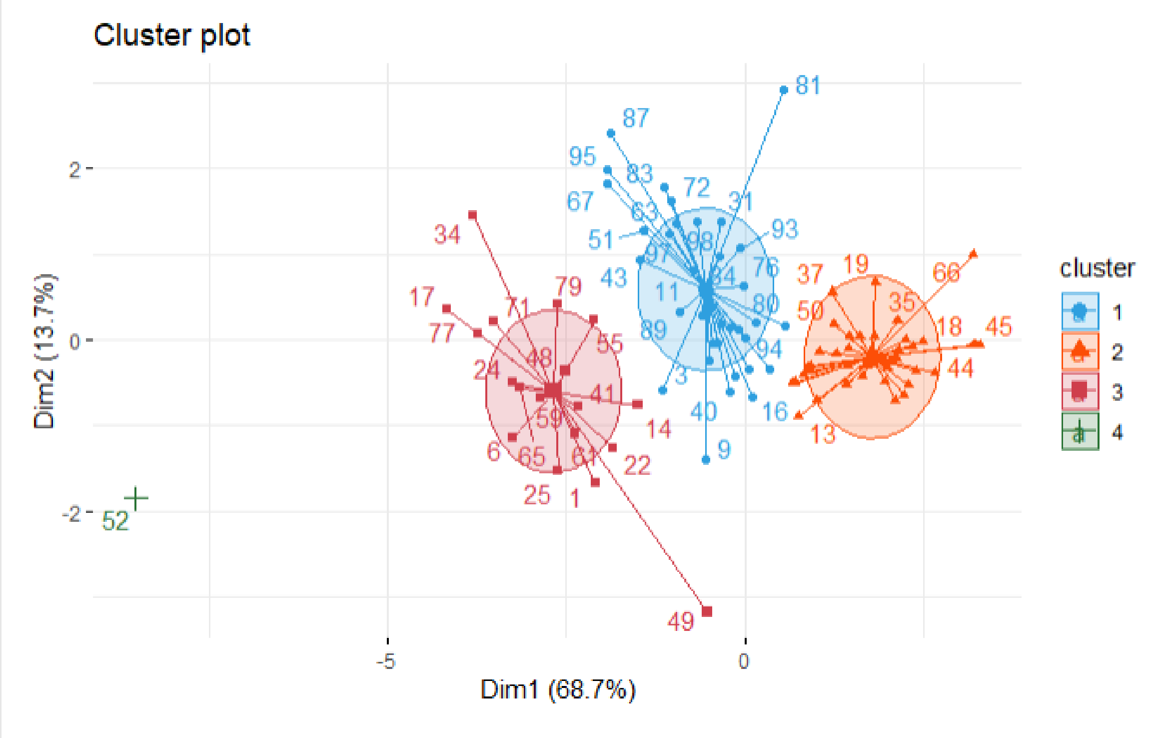
4.2.2. K-means; k=4.

```
{r}
km.res2 <- kmeans(vehicle_cluster_scaled, 4, nstart = 25)
km.res2$size
```

```
[1] 36 43 18  1
```

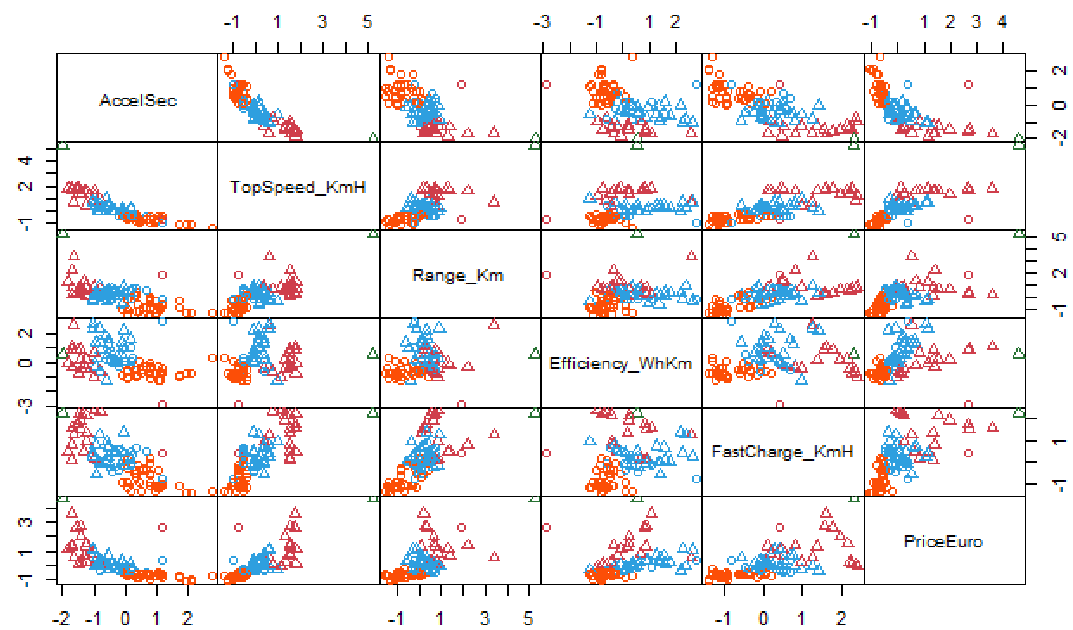
This time we have 4 clusters, the largest being the third, as we can see by the R code

Contrary to what was done in the previous case, let's visualize clustering results in the space spanned by the first two principal components:



We can see that separation between clusters is almost clear but we can visualize the cluster results in the original space:

```
{r}
cl2 <- km.res2$cluster
pairs(vehicle_cluster_scaled, gap=0, pch=cl, col=c("#2E9FDF", "#FC4E07","#CF3E4B",
"#287233")[cl2])
```



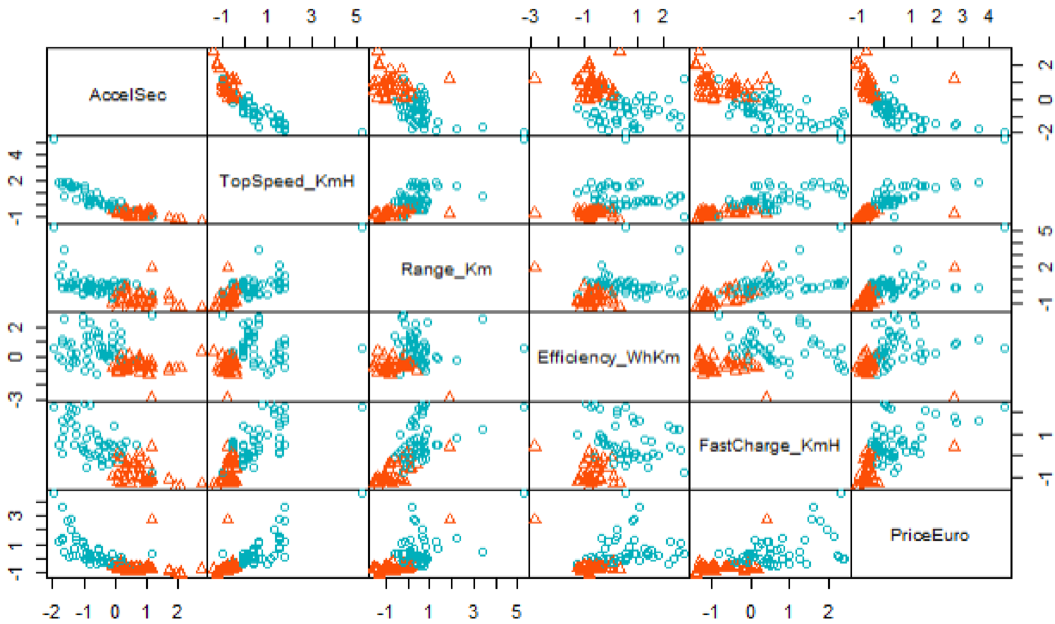
According to both of the above clustering methods, we can see that in both of the cases it is difficult to separate and identify the cluster to due overlapping.

4.2.3. K-medoids; k=2.

```
{r}
#install.packages("cluster")
library(cluster)
pam.res <- pam(vehicle_cluster_scaled, 2)
pam.res$medoids
```

```
Warning: package 'cluster' was built under R version 4.2.2        AccelSec
TopSpeed_KmH    Range_Km Efficiency_WhKm FastCharge_KmH     PriceEuro
[1,] -0.5422689     0.4242211  0.5908318         0.5700275      0.2149691
0.005113015
[2,]  0.7862899    -0.7318876 -0.8471897       -0.7275574     -0.6296973
-0.654004974
```

```
{r}
cl3 <- pam.res$cluster
pairs(vehicle_cluster_scaled, gap=0, pch=cl3, col= c("#00AFBB", "#FC4E07")[cl3] )
```



This is the graphical visualization of clustering results in the original space. Looking at the scatterplot between pairs of variables we can understand which are more useful to identify clusters. If we look at the scatterplot between variables we see a high overlap between clusters. It is almost difficult to find a cluster with this data.

4.3. Cluster Validation.

Assessing Cluster Tendency:

So far we have applied several clustering methods blindly, without knowing whether the data contains clusters. When you run an algorithm, it divides the data into clusters regardless. So, let's check for the clustering tendency. We can assess if there are clusters from both a statistical point of view and from a graphical one, using, respectively, the Hopkins statistic and the VAT algorithm.

4.3.1. Hopkins Statistic.

```
{r}
#install.packages("hopkins")
library(clustertend)
library(hopkins)
hopkins(vehicle_cluster_scaled, n = nrow(vehicle_cluster_scaled)-1)
```

```
Warning: Package `clustertend` is deprecated.  Use package `hopkins` instead.$H
[1] 0.1572238
```

```
{r}
random_df = apply(vehicle_cluster_scaled, 2, function(x){runif(length(x), min(x
),(max(x)))})
random_df=as.data.frame(random_df)
scaled.random_df=scale(random_df)
pairs(scaled.random_df, gap = 0, pch = 21)

library(clustertend)
set.seed(123)
hopkins(vehicle_cluster_scaled, n=nrow(vehicle_cluster_scaled)-1)

hopkins(scaled.random_df, n=nrow(scaled.random_df)-1)
```

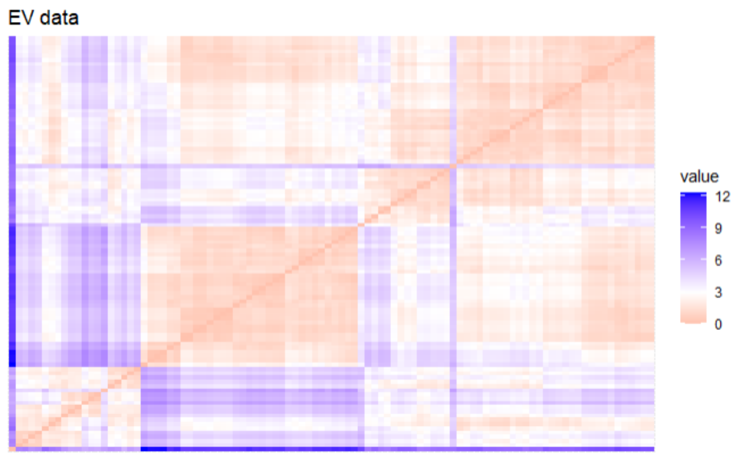
```
Warning: Package `clustertend` is deprecated.  Use package `hopkins` instead.$H
[1] 0.4874045
```

According to the Hopkins statistic (H), the vehicle_cluster_scaled dataset is not so well clusterable, because the H value, 0.15, is different from 0.48, as for the random dataset, which has an H value equal to 0.48 it is not clusterable. Moreover, the underlying assumption of the Hopkins statistic is that the situation without clusters is represented by a uniform distribution: if the dataset is close to the uniform distribution, the Hopkins statistic suggests that there are no clusters; if it is different from the uniform distribution, the Hopkins statistic suggests that there are clusters. There are other configurations without clusters, the reason why it is necessary to underline is that for H=0.5 there are no clusters concerning the assumption that the configuration without clusters is the uniform one.

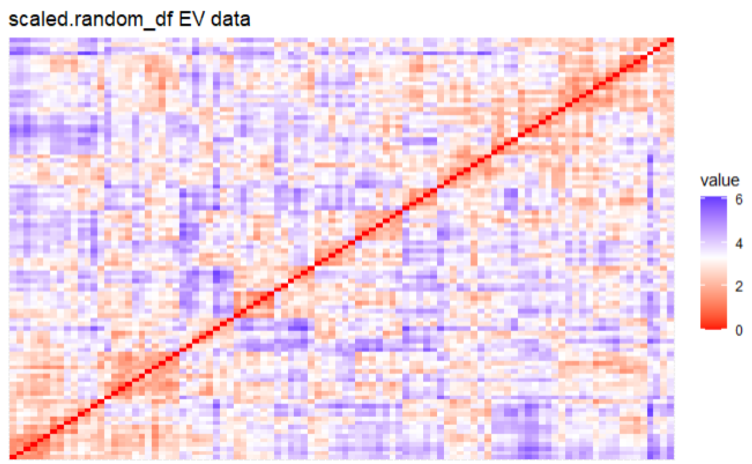
Calculate Hopkins statistics for given data. Calculated values 0-0.3 indicate regularly-spaced data. Values around 0.5 indicate random data. Values 0.7-1 indicate clustered data.

4.3.2. VAT Algorithm.

```
{r}
fviz_dist(dist(vehicle_cluster_scaled), show_labels = FALSE)+ labs(title = "EV
data")
```




```
{r}
fviz_dist(dist(scaled.random_df), show_labels = FALSE)+ labs(title = "scaled
.random_df EV data")
```

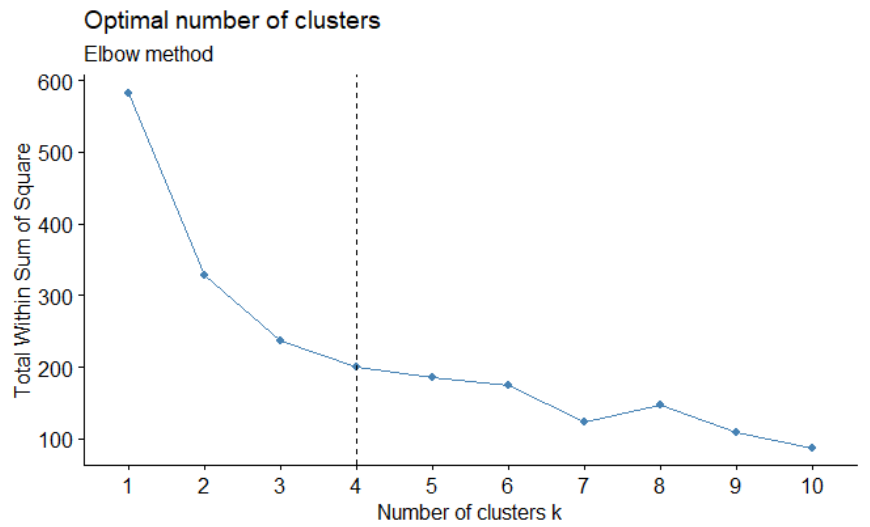


The dissimilarity matrix image confirms that there is a weak cluster structure in the standardized dataset, but not in the randomized one.

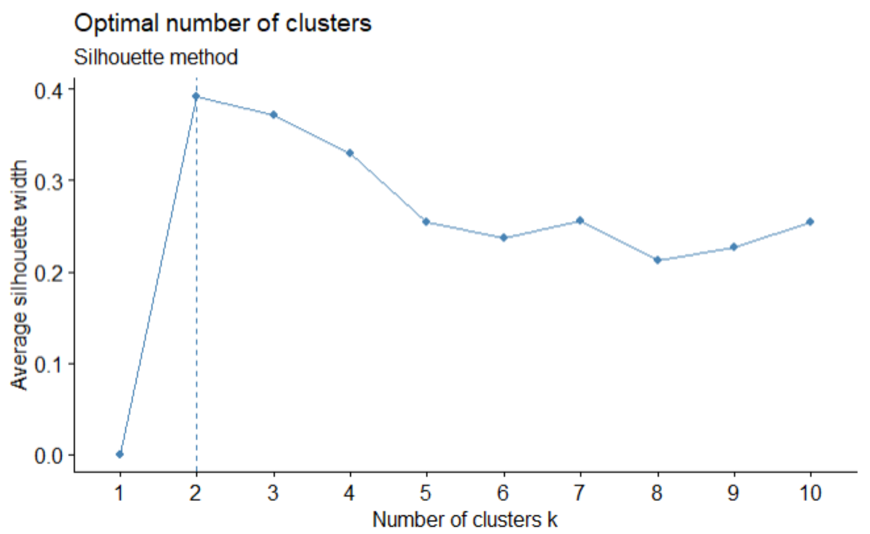
Determining the optimal number of clusters:

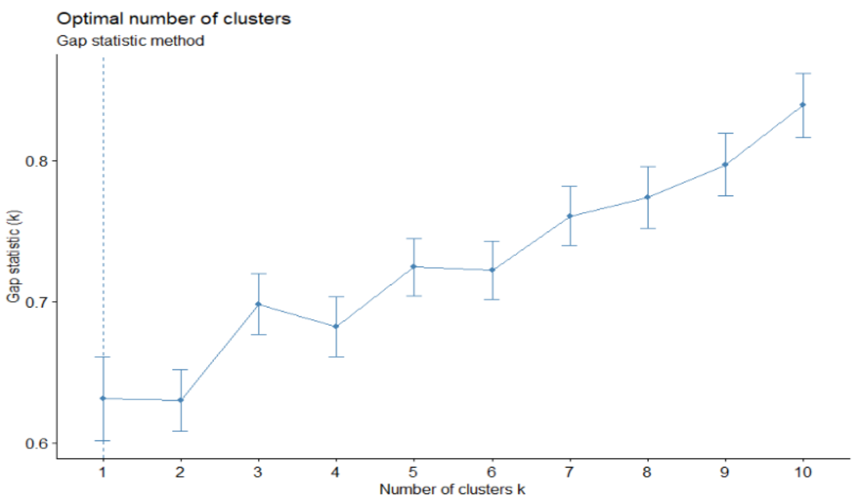
Several methods can be used for determining the optimal number of clusters. Using the fviz_nbclust function we obtain the following results:

```{r}
fviz_nbclust(vehicle_cluster_scaled, kmeans, method = "wss") + geom_vline
(xintercept = 4, linetype = 2)+labs(subtitle = "Elbow method")
```




```{r}
fviz_nbclust(vehicle_cluster_scaled, kmeans, method = "silhouette") + labs(subtitle
= "Silhouette method")
```





In order to find the optimal number of clusters, three indices were used. The elbow method suggested 3 or 4 numbers of the cluster. Silhouette method suggests 2 clusters. Gap statistics 1 clusters. It is difficult chose results from above three methods, so I decided to proceed by identifying 2 clusters.

Using the NbClust function we obtain the following results:

nb <- NbClust(vehicle_cluster_scaled, distance = "euclidean", min.nc = 2,max.nc = 10, method = "kmeans")

```
*** : The Hubert index is a graphical method of determining the number of clusters.
                In the plot of Hubert index, we seek a significant knee that corresponds to a
                significant increase of the value of the measure i.e the significant peak in Hubert
                index second differences plot.

*** : The D index is a graphical method of determining the number of clusters.
                In the plot of D index, we seek a significant knee (the significant peak in Dindex
                second differences plot) that corresponds to a significant increase of the value of
                the measure.

*******************************************************************
* Among all indices:
* 7 proposed 2 as the best number of clusters
* 5 proposed 3 as the best number of clusters
* 3 proposed 4 as the best number of clusters
* 2 proposed 6 as the best number of clusters
* 3 proposed 9 as the best number of clusters
* 3 proposed 10 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  2


*******************************************************************
```

In this case the best solution, according to 7 indices, is K=2; the second-best solution, according to 5 indices, is K=3.

4.4. Cluster Statistics.

Cluster statistics are used to evaluate the goodness of clustering results. It can be categorized into 3 classes: internal, external and relative cluster validation. The internal cluster validation allows us to estimate the optimal number of clusters and to select the appropriate clustering algorithm, using two indices: the Silhouette Width and the Dunn index.

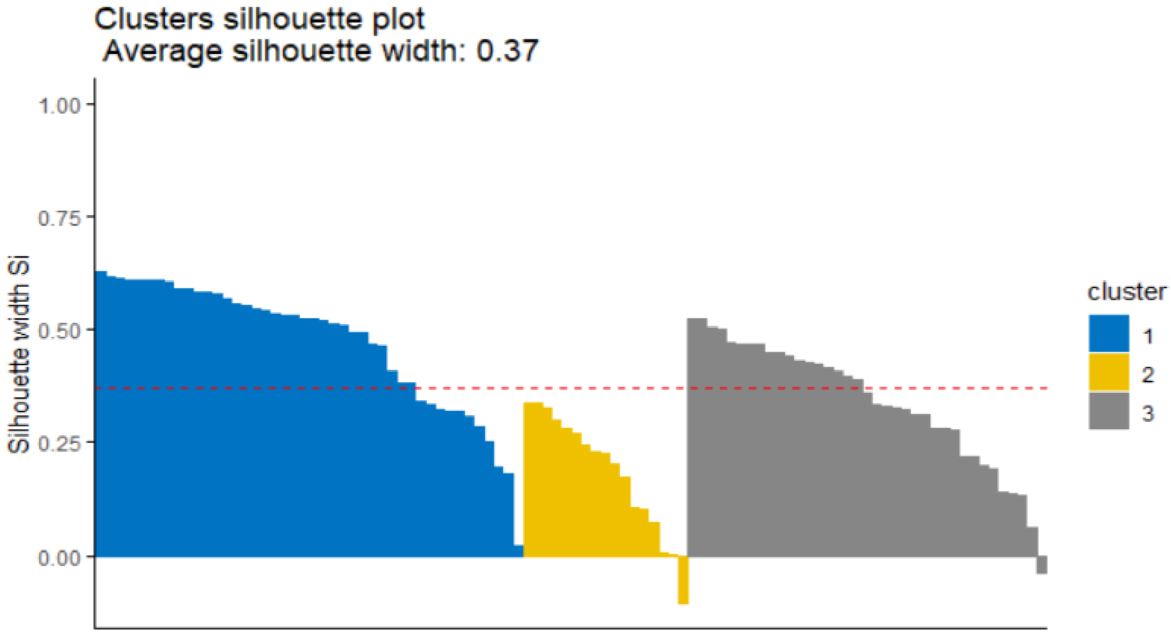
4.4.1. Silhouette Width.

km.res <- eclust(vehicle_cluster_scaled, "kmeans", k = 3, nstart = 25, graph = FALSE)

fviz_silhouette(km.res, palette = "jco", ggtheme = theme_classic())

	cluster <fctr>	size <int>	ave.sil.width <dbl>
1	1	44	0.47
2	2	17	0.18
3	3	37	0.34

3 rows



The average silhouette width is 0.37. The first cluster participates in the overall silhouette width with a greater weight because there are more units in it. Looking at the plot, we can see that in cluster 1 most of the units have a silhouette width higher than the average one; in cluster 2 there are all of the units have a silhouette width lower than the average one and some have a negative silhouette width; in cluster 3 some units have some high silhouette width, others are low one.

As regards the units with a negative silhouette value, this means that they are not in the right cluster, so we have to find their neighbour clusters:

```
{r}
sil <- km.res$silinfo$widths[, 1:3]
neg_sil_index_aver.eu<- which(sil[, "sil_width"] < 0)
sil[neg_sil_index_aver.eu, , drop = FALSE]
```

Description: df [2 × 3]

	cluster <fctr>	neighbor <dbl>	sil_width <dbl>
55	2	3	-0.10581320
94	3	1	-0.03702106

2 rows

According to the silhouette method, 55 should be in cluster 3; 94 should be in cluster 1.

4.4.2. Dunn Index.

The Dunn index aims to identify sets of clusters that are compact, with a small variance between members of the cluster, and well separated. A higher Dunn index indicates better clustering, so it should be maximized.

```
{r}
km_stats <- cluster.stats(dist(vehicle_cluster_scaled), km.res$cluster)
km_stats$dunn
```

```
[1] 0.06050309
```

4.5. Choosing the best Clustering Algorithms.

To choose the best pair of clustering algorithms and an optimal number of clusters, we can use both internal measures and stability ones.

4.5.1. Internal Measures.

clmethods <- c("hierarchical","kmeans","pam")

intern <- clValid(vehicle_cluster_scaled, nClust = 2:6, clMethods = clmethods, validation = "internal")

summary(intern)

```
Clustering Methods:
 hierarchical kmeans pam

Cluster sizes:
 2 3 4 5 6

Validation Measures:
                                 2       3       4       5       6

hierarchical Connectivity   2.9290  5.8579 13.3587 14.6849 17.6139
             Dunn           0.7732  0.5310  0.2477  0.2477  0.2477
             Silhouette     0.6763  0.4485  0.4226  0.4093  0.3328
kmeans       Connectivity  22.6909 19.9024 22.3397 21.1540 34.8853
             Dunn           0.0605  0.0632  0.0751  0.1234  0.1185
             Silhouette     0.3917  0.3968  0.4009  0.3838  0.3395
pam          Connectivity   5.0329 22.0405 40.6353 41.6484 40.4790
             Dunn           0.0679  0.0124  0.0186  0.0509  0.0780
             Silhouette     0.3878  0.3702  0.2908  0.2371  0.2545

Optimal Scores:
```

	Score <dbl>	Method <chr>	Clusters <chr>
Connectivity	2.9290	hierarchical	2
Dunn	0.7732	hierarchical	2
Silhouette	0.6763	hierarchical	2

We are comparing 3 clustering methods, hierarchical, k-means and pam, using 3 indices as connectivity, Dunn and silhouette width. According to this, the best solution is 2 clusters, which corresponds to the hierarchical method.

4.5.2. Stability Measures.

```
{r}
clmethods <- c("hierarchical","kmeans","pam")
stab <- clValid(vehicle_cluster_scaled, nClust = 2:6, clMethods = clmethods
,validation = "stability")
optimalScores(stab)
```

Description: df [4 x 3]

	Score <dbl>	Method <chr>	Clusters <chr>
APN	0.003366295	hierarchical	2
AD	1.550186474	pam	6
ADM	0.025367440	hierarchical	2
FOM	0.645058041	pam	6

For APN and ADM measures, hierarchical clustering with K = 2 clusters again gives the best score. For the other measures, PAM with K = 6 clusters has the best score.

4.6. Model-Based Clustering.

The model-based clustering is based on statistical models, thus trying to solve the disadvantage of traditional clustering methods, which are heuristic and not based on formal models. Now, we are going to implement the Model-Based Clustering via parsimonious Gaussian mixtures models. Gaussian mixtures are the ones most used, because they can model several density functions, by means of the suitable parameters, which are automatically estimated by the function used to fit the model. Moreover, we are referring to "parsimonious configurations" in the sense that we want to explain the reality by using a simple model, namely a model with a low number of parameters, according to the parsimony criterion. However, mixture models that use Gaussian distribution are complex ones, so we are in need to define a parsimonious version, reducing the number of parameters by means of the eigen-decomposition. Basically, we must find the right compromise between the fit of the model and the number of parameters (that is, the complexity of the model): comparing two criteria, if the increase in the fit is small, it is not worth choosing the more complex model. It is something related to the concept of penalization: the better the model fits the data, the higher the number of parameters it uses, which does not respect the parsimony criterion. Therefore, in some sense, the fitting of the model has to be penalized, taking into account the number of parameters, until, as already said, the right compromise is found. Finally, we will select the number of clusters and the best parsimonious configuration, by means of a selection criterion, for example the Bayesian Information Criterion (BIC):

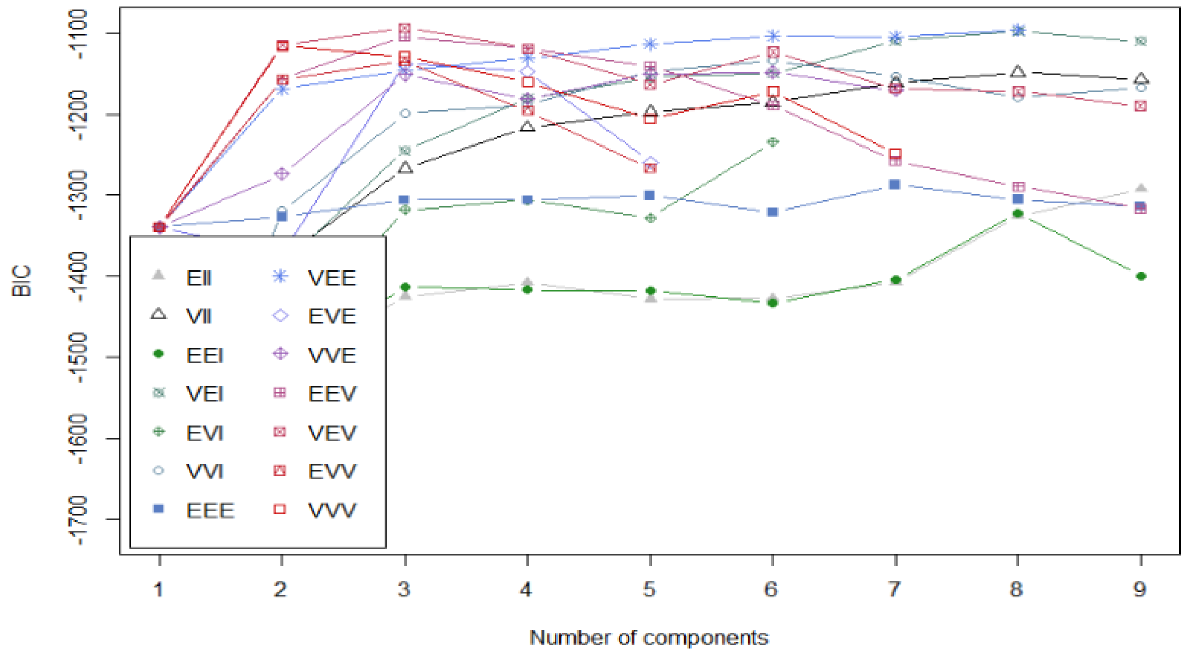
```
{r}
install.packages("mclust")
library(mclust)
mod <- Mclust(vehicle_cluster_scaled, G = 1:9, modelNames = NULL)
summary(mod$BIC)
```

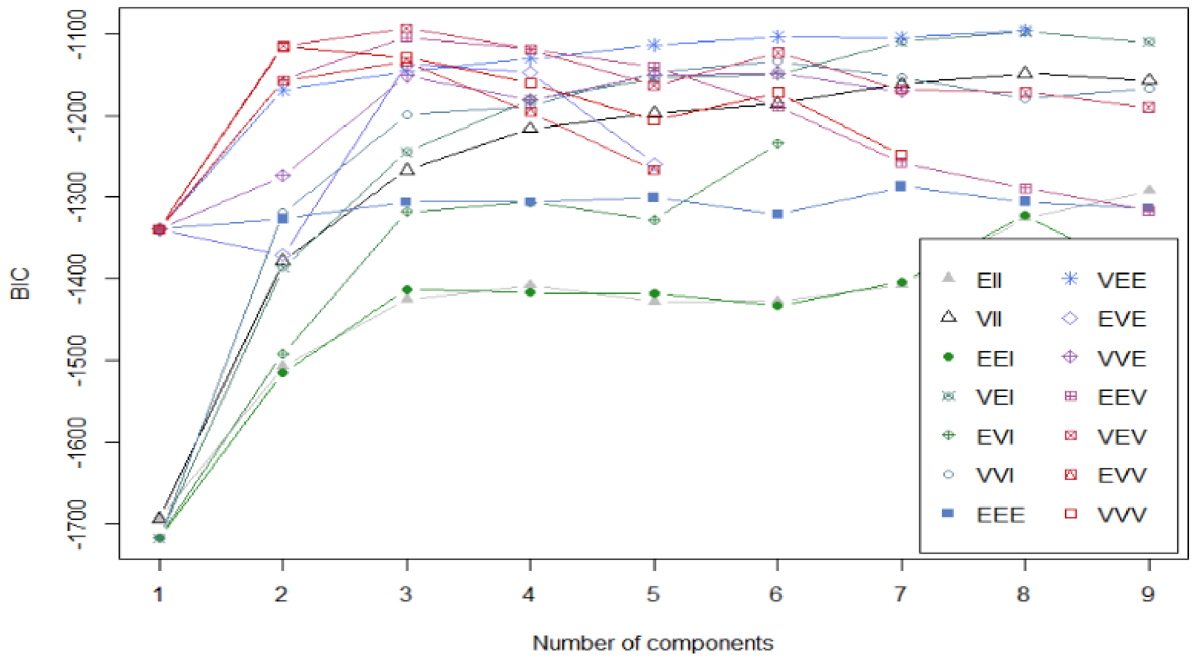
```
Best BIC values:
             VEV,3        VEE,8        VEI,8
BIC      -1092.877 -1095.481272 -1097.792955
BIC diff     0.000    -2.603782    -4.915465
```

Here we have the best 3 models according to the Bayesian Information Criterion (BIC): the best model is VEV (means that clusters have variable volume, variable shape and same orientation) with 3 clusters; the second best is VEE with 8 clusters and the last model is VEI with 8 clusters.

plot(mod, what = "BIC", ylim = range(mod$BIC, na.rm = TRUE), legendArgs = list(x = "bottomleft"))

plot(mod, what = "BIC", ylim = range(mod$BIC, na.rm = TRUE), legendArgs = list(x = "bottomright"))





In this plot, which is the graphical counterpart of the results of the previous code, we have the BIC curves for the penalized models; we have a different curve for each number of clusters and for each parsimonious configuration we have a different symbol. The maximum in this configuration is related to 8 clusters and VEI model.

```
{r}
summary(mod)
```


```
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust VEV (ellipsoidal, equal shape) model with 3 components:

Clustering table:
 1  2  3
22 44 32
```

Description: df [1 × 5]

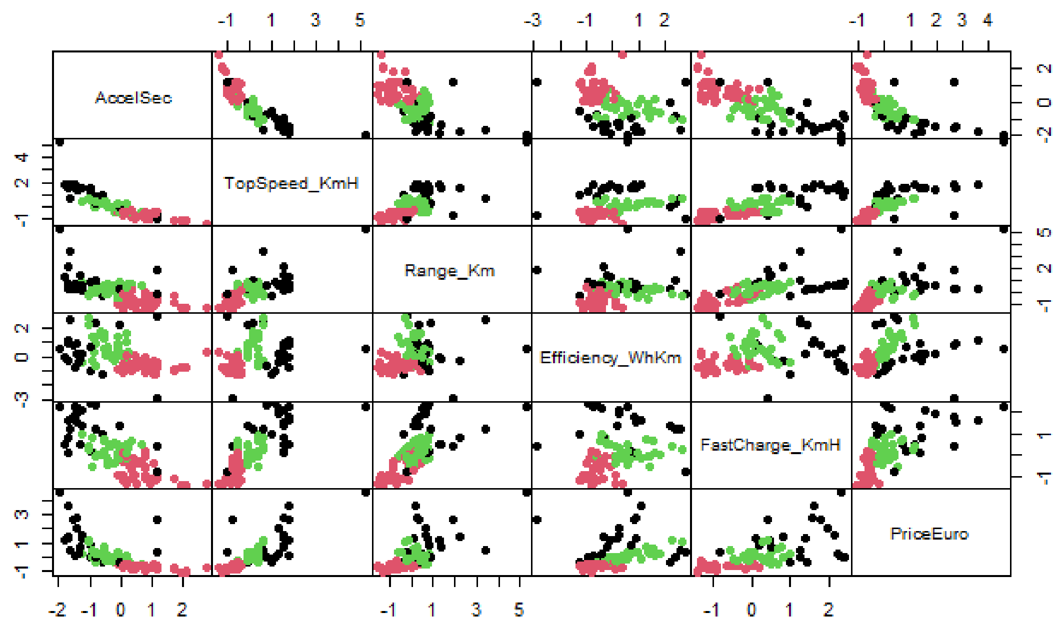
log-likelihood <dbl>	n <int>	df <dbl>	BIC <dbl>	ICL <dbl>
-379.0874	98	73	-1092.877	-1095.474

1 row

Here we can see that BIC = -1092.877 is the best value for BIC, which must be maximized (in fact, this is the less negative value). Moreover, according to this best model, VEV, the first cluster is characterized by 22 observations, the second one by 44 and the last one by 32.

Finally, we can graphically visualize the clustering results:

```
{r}
pairs(vehicle_cluster_scaled, gap=0, pch = 16, col = mod$classification)
```



Colours arise by the best model-based clustering model, VEV with 3 clusters. So, we can see that there is a clustering structure also by the result of Model-Based Clustering.