UNIVERSITÀ degli STUDI di CATANIA | DIPARTIMENTO di ECONOMIA e IMPRESA

JUNE 30, 2021

LIVER DISEASE PREDICTION

This data set contains non-liver and liver patients' records collected from North East of Andhra Pradesh, India.

SYED MUHAMMAD ZAFFAR
SUBMITTED TO:
Prof. Antonio Punzo

Dataset

The Dataset I am using in my analysis is *LiverDisease*. This data is available on the link which is `https://www.kaggle.com/uciml/indian-liver-patient-records`. Patients with Liver disease have been continuously increasing because of excessive consumption of alcohol, inhale of harmful gases, intake of contaminated food, pickles and drugs. This dataset was used to evaluate prediction algorithms in an effort to reduce burden on doctors. Each row in the data provides relevant information about the patient. This dataset is used to predict whether a patient is likely to have liver disease based on the input parameters used in the dataset.

```
liverDisease <- read.csv("Liver_Disease.csv")
```

First we will see the actual dimensions of our dataset that how many objects and variables that the dataset have so, for this I will use the R basic function of dimension as follows:

```
dim(liverDisease)
```

```
## [1] 583  11
```

Here we can see that the total observations are `583` and the total number of attributes are `11`.

Exploration of Dataset

Now, we will see the complete structure of the dataset and also make some transformations in the dataset if needed so.

```
str(liverDisease)
```

```
## 'data.frame':    583 obs. of  11 variables:
##  $ Age                       : int  65 62 62 58 72 46 26 29 17 55 ...
##  $ Gender                    : chr  "Female" "Male" "Male" "Male" ...
##  $ Total_Bilirubin           : num  0.7 10.9 7.3 1 3.9 1.8 0.9 0.9 0.9 0.7
...
##  $ Direct_Bilirubin          : num  0.1 5.5 4.1 0.4 2 0.7 0.2 0.3 0.3 0.2
...
##  $ Alkaline_Phosphotase      : int  187 699 490 182 195 208 154 202 202
290 ...
##  $ Alamine_Aminotransferase  : int  16 64 60 14 27 19 16 14 22 53 ...
##  $ Aspartate_Aminotransferase: int  18 100 68 20 59 14 12 11 19 58 ...
##  $ Total_Protiens            : num  6.8 7.5 7 6.8 7.3 7.6 7 6.7 7.4 6.8
...
##  $ Albumin                   : num  3.3 3.2 3.3 3.4 2.4 4.4 3.5 3.6 4.1
3.4 ...
##  $ Albumin_and_Globulin_Ratio: num  0.9 0.74 0.89 1 0.4 1.3 1 1.1 1.2 1
...
##  $ Dataset                   : int  1 1 1 1 1 1 1 1 2 1 ...
```

The following description would help us to the to predict `Liver Disease`:

Age : Age of the patient $Gender$: Gender of the patient $Total_Bilirubin$: Is a substance produced during the normal breakdown of red blood cells $Direct_Bilirubin$: In the liver, bilirubin is changed into a form that your body can get rid of $Alkaline_Phosphotase$: is one kind enzyme found in your body, If you have liver disease $Alamine_Aminotransferase$: is an enzyme that increases in the blood when the liver is damaged $Aspartate_Aminotransferase$: is a blood test that checks for liver damage $Total_Protiens$: test measures the total amount albumin and globulin in your body $Albumin$: is a protein made by your liver $Albumin_and_Globulin_Ratio$: measures the protein ratio in your body $Dataset$: field used to split the data into two sets (patient with liver disease = 2, or no disease = 1)

Now we will see the observations in the dataset as follows:

```
head(liverDisease)
```

```
##   Age Gender Total_Bilirubin Direct_Bilirubin Alkaline_Phosphotase
## 1  65 Female             0.7              0.1                  187
## 2  62   Male            10.9              5.5                  699
## 3  62   Male             7.3              4.1                  490
## 4  58   Male             1.0              0.4                  182
## 5  72   Male             3.9              2.0                  195
## 6  46   Male             1.8              0.7                  208
##   Alamine_Aminotransferase Aspartate_Aminotransferase Total_Protiens
Albumin
## 1                       16                         18            6.8
3.3
## 2                       64                        100            7.5
3.2
## 3                       60                         68            7.0
3.3
## 4                       14                         20            6.8
3.4
## 5                       27                         59            7.3
2.4
## 6                       19                         14            7.6
4.4
##   Albumin_and_Globulin_Ratio Dataset
## 1                       0.90       1
## 2                       0.74       1
## 3                       0.89       1
## 4                       1.00       1
## 5                       0.40       1
## 6                       1.30       1
```

We can see that the dataset doesn't require much transformation because the dataset is almost clean but changes need to be done before processing further.

Here I will change the column name of `dataset` into `Liver_Disease` and will also change the values in the column that patient with liver disease = 1 or no disease = 0 as follows:

```
names(liverDisease)[names(liverDisease) == "Dataset"] <- "Liver_Disease"

liverDisease$Liver_Disease[liverDisease$Liver_Disease == "1"] <- 0

liverDisease$Liver_Disease[liverDisease$Liver_Disease == "2"] <- 1
```

Now I will also change the `Gender` columns' observations into binary form as for `Male=1` and for `Female=0` as follows:

```
liverDisease$Gender[liverDisease$Gender == "Female"] <- 0

liverDisease$Gender[liverDisease$Gender == "Male"] <- 1
```

Here we can see that the column name and the values has been transformed successfully.Now I will make some transformation on the data for the clarity of the data, Here we can see that the `Gender` and `Liver_Disease` variables doesn't belong to a right data type so I will initially transform them to `Factor` as follows:

```
liverDisease$Gender <- factor(liverDisease$Gender)

liverDisease$Liver_Disease <- as.numeric(liverDisease$Liver_Disease)
liverDisease$Liver_Disease <- factor(liverDisease$Liver_Disease)
```

Here I will print the structure of the dataset and also check the if there is any missing values available in the dataset as follows:

```
str(liverDisease)

## 'data.frame':    583 obs. of  11 variables:
##  $ Age                       : int  65 62 62 58 72 46 26 29 17 55 ...
##  $ Gender                    : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 1 1
2 2 ...
##  $ Total_Bilirubin           : num  0.7 10.9 7.3 1 3.9 1.8 0.9 0.9 0.9 0.7
...
##  $ Direct_Bilirubin          : num  0.1 5.5 4.1 0.4 2 0.7 0.2 0.3 0.3 0.2
...
##  $ Alkaline_Phosphotase      : int  187 699 490 182 195 208 154 202 202
290 ...
##  $ Alamine_Aminotransferase  : int  16 64 60 14 27 19 16 14 22 53 ...
##  $ Aspartate_Aminotransferase: int  18 100 68 20 59 14 12 11 19 58 ...
##  $ Total_Protiens            : num  6.8 7.5 7 6.8 7.3 7.6 7 6.7 7.4 6.8
...
##  $ Albumin                   : num  3.3 3.2 3.3 3.4 2.4 4.4 3.5 3.6 4.1
3.4 ...
##  $ Albumin_and_Globulin_Ratio: num  0.9 0.74 0.89 1 0.4 1.3 1 1.1 1.2 1
...
##  $ Liver_Disease             : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1
2 1 ...
```

```
colSums(sapply(liverDisease, is.na))
```

```
##                         Age                      Gender
##                           0                           0
##             Total_Bilirubin           Direct_Bilirubin
##                           0                           0
##       Alkaline_Phosphotase   Alamine_Aminotransferase
##                           0                           0
## Aspartate_Aminotransferase             Total_Protiens
##                           0                           0
##                     Albumin Albumin_and_Globulin_Ratio
##                           0                           4
##               Liver_Disease
##                           0
```

Here we can see that the `Albumin_and_Globulin_Ratio` has 4 missing values so here we will decide with the help of percentage data missing, whether we will omit these `NAs` or change them with the mean or median values as follows:

```
sum(is.na(liverDisease)) / (nrow(liverDisease) *ncol(liverDisease))
```

```
## [1] 0.000623733
```

Here the missing data percentage is almost zero so we will omit the NAs from the dataset as follows:

```
liver <- na.omit(liverDisease)

str(liver)
```

```
## 'data.frame':    579 obs. of  11 variables:
##  $ Age                       : int  65 62 62 58 72 46 26 29 17 55 ...
##  $ Gender                    : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 1 1
2 2 ...
##  $ Total_Bilirubin           : num  0.7 10.9 7.3 1 3.9 1.8 0.9 0.9 0.9 0.7
...
##  $ Direct_Bilirubin          : num  0.1 5.5 4.1 0.4 2 0.7 0.2 0.3 0.3 0.2
...
##  $ Alkaline_Phosphotase      : int  187 699 490 182 195 208 154 202 202
290 ...
##  $ Alamine_Aminotransferase  : int  16 64 60 14 27 19 16 14 22 53 ...
##  $ Aspartate_Aminotransferase: int  18 100 68 20 59 14 12 11 19 58 ...
##  $ Total_Protiens            : num  6.8 7.5 7 6.8 7.3 7.6 7 6.7 7.4 6.8
...
##  $ Albumin                   : num  3.3 3.2 3.3 3.4 2.4 4.4 3.5 3.6 4.1
3.4 ...
##  $ Albumin_and_Globulin_Ratio: num  0.9 0.74 0.89 1 0.4 1.3 1 1.1 1.2 1
...
##  $ Liver_Disease             : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1
2 1 ...
```

```
##  - attr(*, "na.action")= 'omit' Named int [1:4] 210 242 254 313
##   ..- attr(*, "names")= chr [1:4] "210" "242" "254" "313"
```

After removing the NAs from the dataset so now the total observations are 579 and total attributes are 11.

The first type of analysis we do is Univariate analysis, we will keep going with the preliminary analysis, as to understand that if it is useful to run a Principle Component Analysis and ClusteringAnalysis on the data set.

Univariate Analysis

Age

Lets explore the Age variable:

```
head(liver$Age)
```

```
## [1] 65 62 62 58 72 46
```

```
length(liver$Age)
```

```
## [1] 579
```

```
table(liver$Age)
```

```
##
##  4  6  7  8 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31
##  2  1  2  1  1  1  2  4  2  1  3  5 11  2  3  7  9  3  5  5 14  5  8  7 10
8
## 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56
57
## 20 15  8 11 11  9 21  6 17  5 21  4  3 24 16  6 20 11 23  9  7  6  8 18  4
7
## 58 60 61 62 63 64 65 66 67 68 69 70 72 73 74 75 78 84 85 90
## 14 34  5  9  2  6 17 12  1  4  2  9  8  2  4 14  1  1  1  1
```

The table() function in R returns the absolute frequencies for each patient-specified value in the data set.

```
unique(liver$Age)
```

```
##  [1] 65 62 58 72 46 26 29 17 55 57 64 74 61 25 38 33 40 51 63 34 20 84 52
30 48
## [26] 47 45 42 50 85 35 21 32 31 54 37 66 60 19 75 68 70 49 14 13 18 39 27
36 24
## [51] 28 53 15 56 44 41  7 22  8  6  4 43 23 12 69 16 78 11 73 67 10 90
```

```
length(unique(liver$Age))
```

```
## [1] 72
```

```
min(liver$Age)
```

```
## [1] 4
```

```
max(liver$Age)
```

```
## [1] 90
```

Here the total observation of Age is 579 and is a continuous variable that range lies between 4-890. Now, we will see the summary of the Age variable, for further analysis as follows:

```
summary(liver$Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    4.00   33.00   45.00   44.78   58.00   90.00
```

```
sd(liver$Age)
```

```
## [1] 16.22179
```

```
var(liver$Age)
```

```
## [1] 263.1463
```

```
getMode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
v <- c(liver$Age)
mode <- getMode(v)
print(mode)
```

```
## [1] 60
```

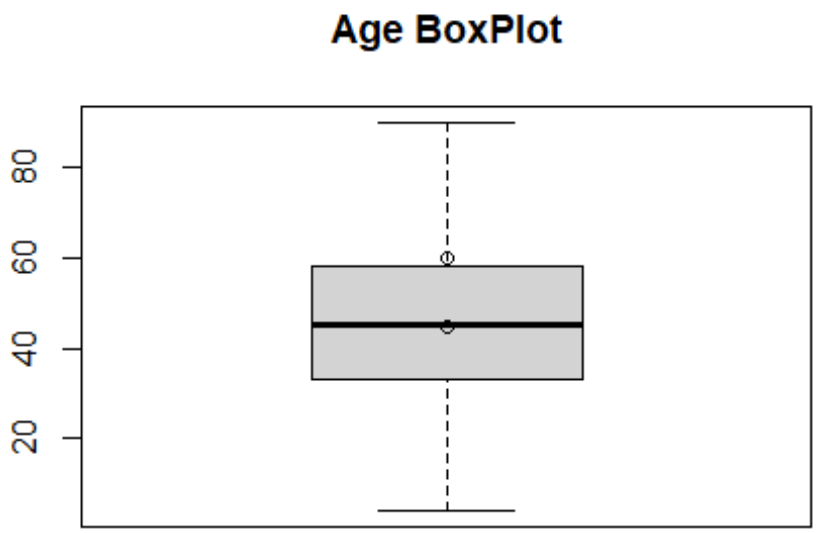
From the above summary we can observe that the Mean (44.78), Median (45.00) and Mode (60) are not equal so the distribution is asymmetrical. Here Mode > Median > Mean so the distribution could be skewed to the left. Now we also confirm this as follows:

```
library(moments)
skewness(liver$Age)
```

```
## [1] -0.03350366
```

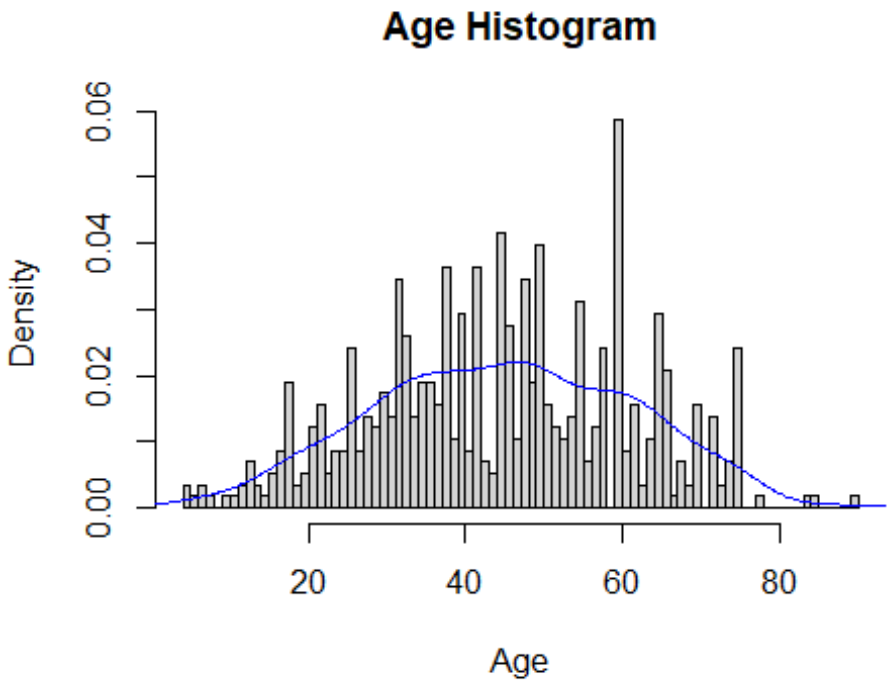
Hence, the skewness is the negative so the distribution is negatively skewed was rightly observed. Also I will plot the Box plot to comparing the distribution of data across dataset as follows:

```
boxplot(liver$Age, main = "Age BoxPlot")
points(mean(liver$Age))
points(mode)
```



Box plots are graphical displays of the five number summary, So they describe the Minimum (lowest mean relative) lower whisker, Quartile 1 (It separates the lowest 25% observation to the rest of the observations), Median (Which divides the lower 50% observation from the upper 50% observations), Quartile 3 (It divides the upper 25% observation to the rest of the observations) and Maximum. In the above Box plot diagram, there are no outliers and the upper whisker shows the maximum. Now, I will plot a histogram a graphical display of data using bars of different heights to measures distribution of continuous data as follows:

```
hist(liver$Age, prob = TRUE, breaks = 72, xlab = "Age", main = "Age
Histogram")
lines(density(liver$Age), col='blue')
```



From the above histogram, we can observe that the diagram has many peaks.Using the density lines, I have approximated the histogram graph. Density provides information about the type of distribution under consideration and allows us to determine whether the attribute is distributed normally or not. From the graph above we can see how the `Age` variable does not seem to fit to a Normal distribution, there is a peak of the frequency distributions around many values and also a left skewed. We have already observed a negative value of `0.03350366`, so we will affirm that the distribution is left skewed. To check whether the distribution is `Platykurtic` or not, we will check this by following R method as follows:

```
kurtosis(liver$Age)
```

```
## [1] 2.429595
```

The distribution is also `Platykurtic`, since the value is less than 3.

Now I will plot the Boxplot of the Age variable with the Liver_Disease variable to check the Liver_Disease across the Age as follows:

```
library(ggplot2)
ggplot(liver, aes(x = Liver_Disease, y = liver$Age, fill = Liver_Disease)) +
  geom_boxplot() +
  ylab("Age") +
  ggtitle("Boxplot of the Age across the Liver_Disease")
```

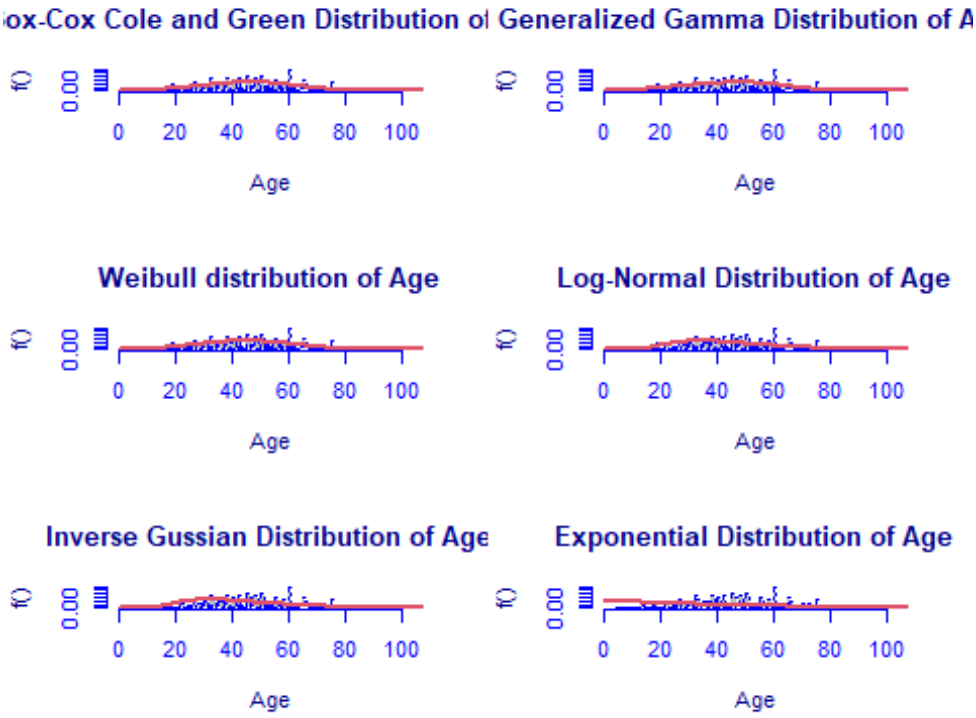
From the graphs we see what the data looks like. The mean age is about 44 which is close to the median of 45. The oldest person in the data is 90 whilst the youngest is 4. The age variable seems to have a bell shaped curve and there seems to be a difference in the ages for the two different liver disease groups. Liver_Disease = 0 has a higher mean than Liver_Disease = 1.

Age Fit for the Data

Now we will try to fit different models to Age distribution evaluating the Akaike Information Criterion (AIC) and the Bayesian Information criterion (BIC), The lower are the indexes, the better is the fitting. I will evaluate both the single parameter distributions and mixture distributions as follows:

```
library(MASS)

## Warning: package 'MASS' was built under R version 4.0.4

library(gamlss)

## Warning: package 'gamlss' was built under R version 4.0.5

## Loading required package: splines

## Loading required package: gamlss.data

## Warning: package 'gamlss.data' was built under R version 4.0.4
```

```
##
## Attaching package: 'gamlss.data'

## The following object is masked from 'package:datasets':
##
##     sleep

## Loading required package: gamlss.dist

## Warning: package 'gamlss.dist' was built under R version 4.0.5

## Loading required package: nlme

## Loading required package: parallel

##  **********   GAMLSS Version 5.3-4  **********

## For more on GAMLSS look at https://www.gamlss.com/

## Type gamlssNews() to see new features/changes/bug fixes.

library(splines)


par(mfrow=c(3,2))

age.BCCG <- histDist(liver$Age, family=BCCG, nbins = 72, xlab = "Age",
main="Box-Cox Cole and Green Distribution of Age")

age.GG <- histDist(liver$Age, family=GG, nbins = 72, xlab = "Age",
main="Generalized Gamma Distribution of Age")

age.WEI <- histDist(liver$Age, family=WEI, nbins = 72, xlab = "Age",
main="Weibull distribution of Age")

age.LOGNO <- histDist(liver$Age, family=LOGNO, nbins = 72, xlab = "Age",
main="Log-Normal Distribution of Age")

age.IG <- histDist(liver$Age, family=IG, nbins=72, xlab = "Age", main =
"Inverse Gussian Distribution of Age")

age.EXP<- histDist(liver$Age, family=EXP, nbins=72, xlab = "Age", main =
"Exponential Distribution of Age")
```




```
library(Matrix)
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.0.5
```

```
## Loaded glmnet 4.1-1
```

```
df <- data.frame(Rownames = c("Box-Cox Cole and Green","Generalized Gamma",
"Weibull",
                              "Log-Normal", "Inverse Gussian", "Exponential"),
                 AIC = c(AIC(age.BCCG), AIC(age.GG), AIC(age.WEI),
AIC(age.LOGNO),
                         AIC(age.IG), AIC(age.EXP)),
                 BIC = c(age.BCCG$sbc, age.GG$sbc, age.WEI$sbc,
age.LOGNO$sbc,
                         age.IG$sbc, age.EXP$sbc),
                 df = c(age.BCCG$df.fit, age.GG$df.fit, age.WEI$df.fit,
                        age.LOGNO$df.fit, age.IG$df.fit, age.EXP$df.fit),
                 LogLike = c(logLik(age.BCCG), logLik(age.GG),
logLik(age.WEI),
                             logLik(age.LOGNO), logLik(age.IG),
logLik(age.EXP)))
df
```

```
##                 Rownames      AIC      BIC df   LogLike
## 1 Box-Cox Cole and Green 4870.080 4883.164  3 -2432.040
## 2      Generalized Gamma 4862.331 4875.415  3 -2428.165
```

```
## 3               Weibull 4867.857 4876.579  2 -2431.928
## 4            Log-Normal 5021.650 5030.373  2 -2508.825
## 5        Inverse Gussian 5068.919 5077.642  2 -2532.459
## 6           Exponential 5562.502 5566.863  1 -2780.251
```

As we can see, the model with the highest log likelihood (-2428.165) and the lowest AIC (4862.331) and BIC (4875.415) is the Generalized Gamma Distribution. Therefore, based on the greatest likelihood technique, our data fits better in the Generalized Gamma Distribution. Now, I will also compare two models by means of the Likelihood ratio test as we performed above as follows:

```
library(zoo)

## Warning: package 'zoo' was built under R version 4.0.5

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

library(lmtest)

## Warning: package 'lmtest' was built under R version 4.0.5

lrtest(age.GG, age.EXP)

## Likelihood ratio test
##
## Model 1: gamlssML(formula = liver$Age, family = "GG")
## Model 2: gamlssML(formula = liver$Age, family = "EXP")
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   3 -2428.2
## 2   1 -2780.2 -2 704.17  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Under the null hypothesis, we compare the Generalized Gamma distribution to the Exponential distribution under the alternative hypothesis. At any level of significance, we can reject the null hypothesis. Because our distributions fit better in the Generalized Gamma model, we'll use it.

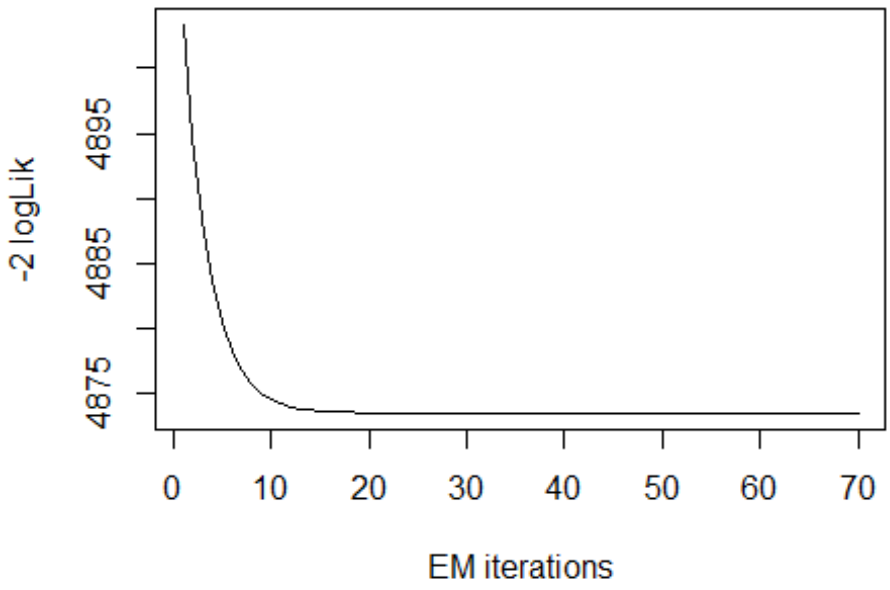
Distributions Mixture

Now, I will also apply the fitting criteria for the distributions with Gamma mixture as follows:

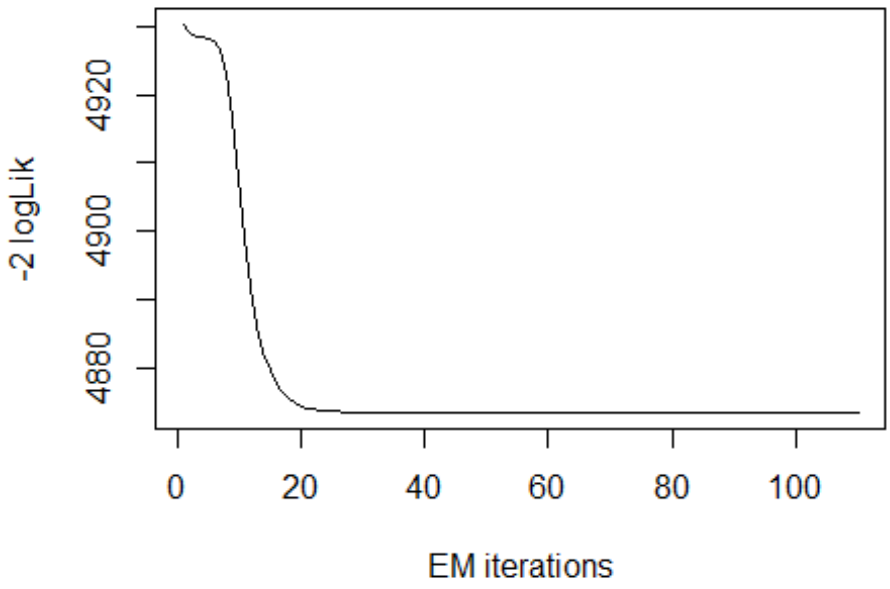
```
library(gamlss.mx)

## Warning: package 'gamlss.mx' was built under R version 4.0.5
```

```
## Loading required package: nnet
```

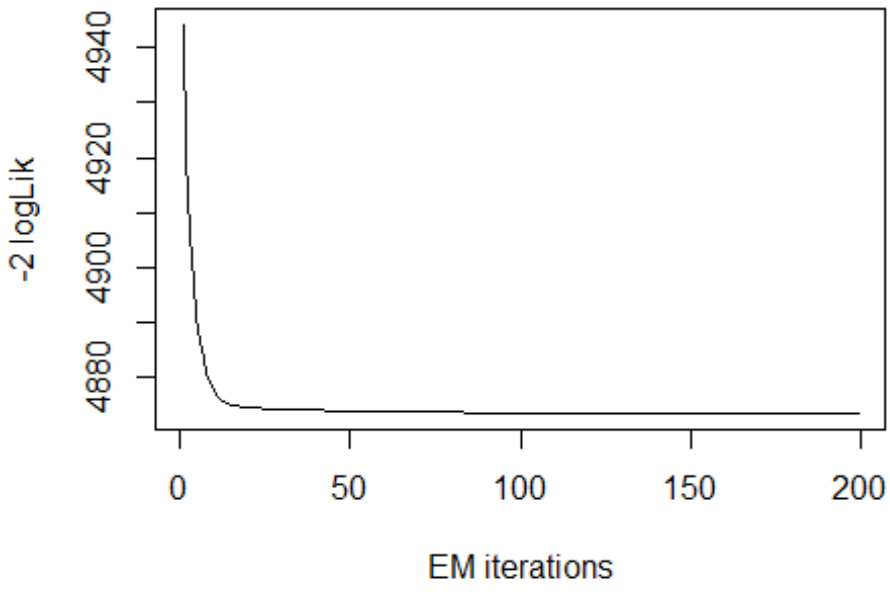
```
library(nnet)
mix.gam <- gamlssMXfits(n = 5, liver$Age~1, family = GA, K = 2, data = liver)
```



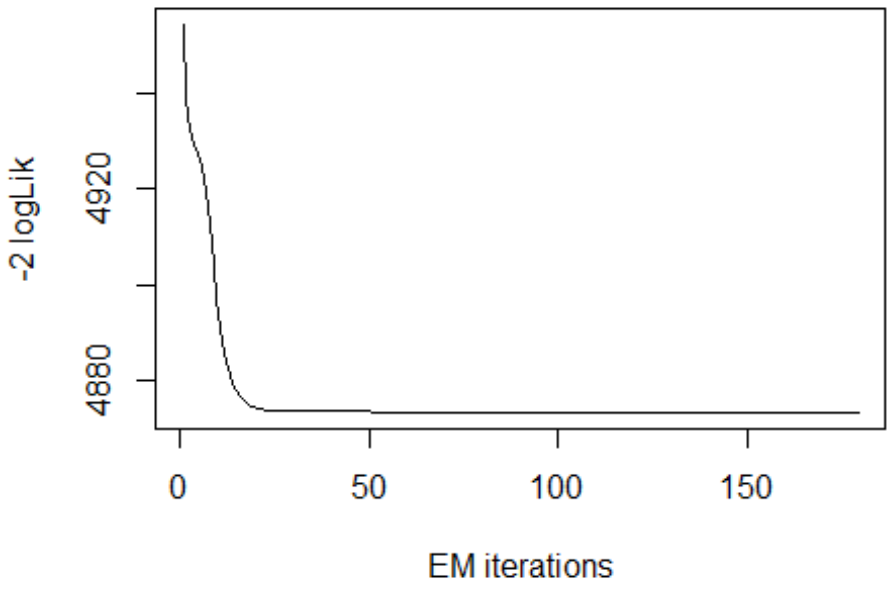

```
## model= 1
```



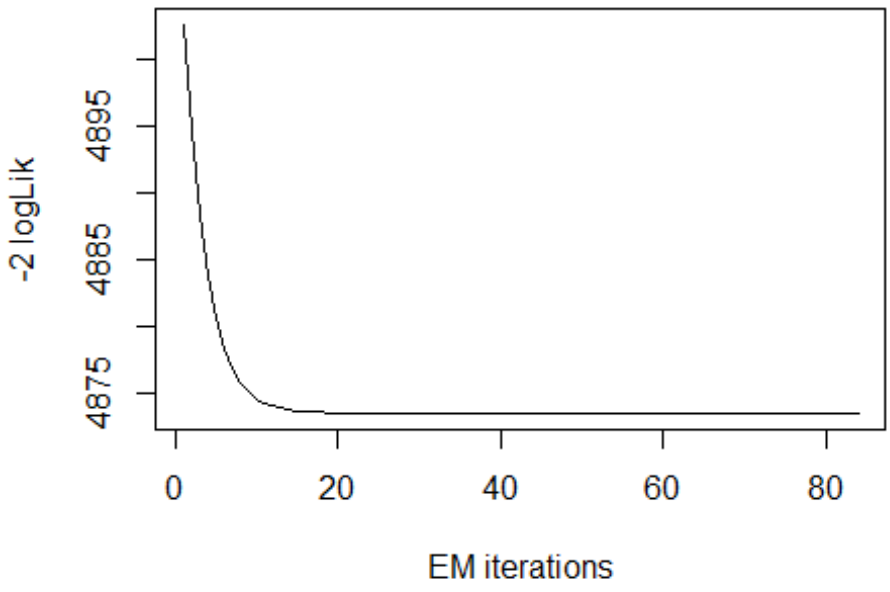

```
## model= 2
```




```
## model= 3
```




```
## model= 4
```




```
## model= 5
```

```r
print(mix.gam)
```

```
##
## Mixing Family:  c("GA", "GA")
##
## Fitting method: EM algorithm
##
## Call:  gamlssMX(formula = liver$Age ~ 1, family = GA, K = 2,
##     data = liver)
##
## Mu Coefficients for model: 1
## (Intercept)
##       3.957
## Sigma Coefficients for model: 1
## (Intercept)
##       -1.42
## Mu Coefficients for model: 2
## (Intercept)
##       3.516
## Sigma Coefficients for model: 2
## (Intercept)
##     -0.7862
##
## Estimated probabilities: 0.59742 0.40258
##
## Degrees of Freedom for the fit: 5 Residual Deg. of Freedom   574
## Global Deviance:     4873.44
##             AIC:     4883.44
##             SBC:     4905.25
```

We can observe that the AIC value of the mixture of Gamma has improved, since it is higher than that of the single Gamma distribution. The current `AIC` value is `4883.44`, whereas the previous value was `4862.331` and the current value of `BIC` value is `4905.25`, whereas the previous value was `4875.415`.

```r
logLik(mix.gam)
```

```
## 'log Lik.' -2436.722 (df=5)
```

```r
mix.gam$prob
```

```
## [1] 0.59742 0.40258
```

```r
fitted(mix.gam, "mu")[1]
```

```
## [1] 44.78439
```

```r
fitted(mix.gam, "sigma")[2]
```

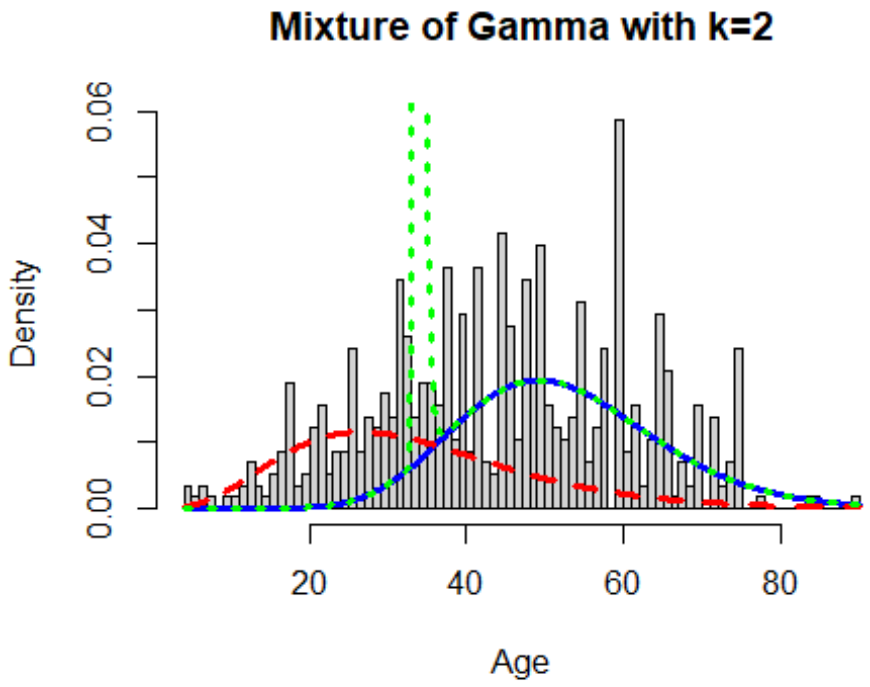
```
## [1] 44.78439
```

```r
hist(liver$Age, breaks = 72, xlab = "Age", main="Mixture of Gamma with k=2",
freq = FALSE)

mu.hat1 <- exp(mix.gam[["models"]][[1]][["mu.coefficients"]])

sigma.hat1 <- exp(mix.gam[["models"]][[1]][["sigma.coefficients"]])

mu.hat2 <- exp(mix.gam[["models"]][[2]][["mu.coefficients"]])

sigma.hat2 <- exp(mix.gam[["models"]][[2]][["sigma.coefficients"]])

hist(liver$Age, breaks = 72, freq = FALSE, plot = FALSE)
```

```
## Warning in hist.default(liver$Age, breaks = 72, freq = FALSE, plot =
FALSE):
## argument 'freq' is not made use of

## $breaks
##  [1]  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
27 28
## [26] 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
52 53
## [51] 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76
77 78
## [76] 79 80 81 82 83 84 85 86 87 88 89 90
##
## $counts
##  [1]  2  1  2  1  0  1  1  2  4  2  1  3  5 11  2  3  7  9  3  5  5 14  5
8  7
## [26] 10  8 20 15  8 11 11  9 21  6 17  5 21  4  3 24 16  6 20 11 23  9  7
6  8
## [51] 18  4  7 14  0 34  5  9  2  6 17 12  1  4  2  9  0  8  2  4 14  0  0
1  0
## [76]  0  0  0  0  1  1  0  0  0  0  1
##
## $density
##  [1] 0.003454231 0.001727116 0.003454231 0.001727116 0.000000000
0.001727116
##  [7] 0.001727116 0.003454231 0.006908463 0.003454231 0.001727116
0.005181347
## [13] 0.008635579 0.018998273 0.003454231 0.005181347 0.012089810
0.015544041
## [19] 0.005181347 0.008635579 0.008635579 0.024179620 0.008635579
0.013816926
## [25] 0.012089810 0.017271157 0.013816926 0.034542314 0.025906736
0.013816926
## [31] 0.018998273 0.018998273 0.015544041 0.036269430 0.010362694
0.029360967
## [37] 0.008635579 0.036269430 0.006908463 0.005181347 0.041450777
0.027633851
```

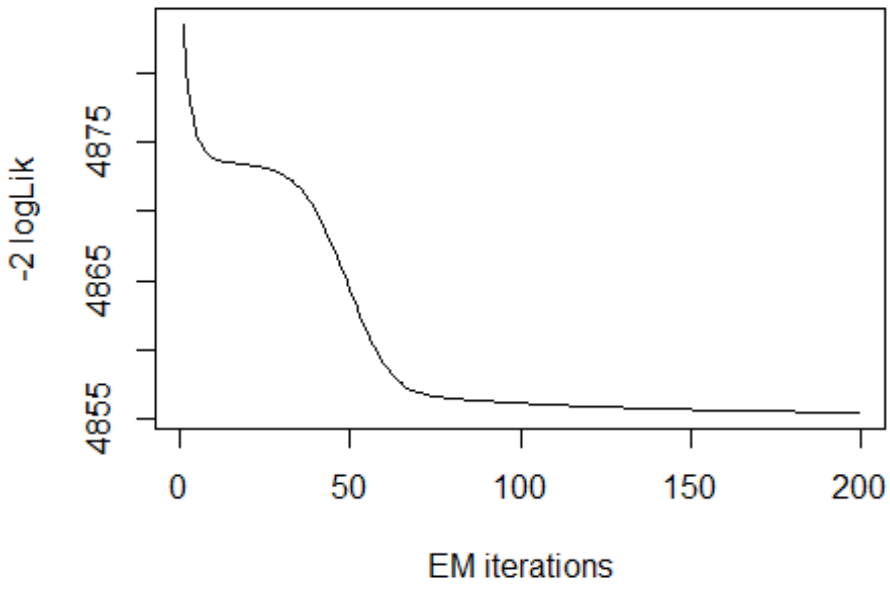
```
## [43] 0.010362694 0.034542314 0.018998273 0.039723661 0.015544041
0.012089810
## [49] 0.010362694 0.013816926 0.031088083 0.006908463 0.012089810
0.024179620
## [55] 0.000000000 0.058721934 0.008635579 0.015544041 0.003454231
0.010362694
## [61] 0.029360967 0.020725389 0.001727116 0.006908463 0.003454231
0.015544041
## [67] 0.000000000 0.013816926 0.003454231 0.006908463 0.024179620
0.000000000
## [73] 0.000000000 0.001727116 0.000000000 0.000000000 0.000000000
0.000000000
## [79] 0.000000000 0.001727116 0.001727116 0.000000000 0.000000000
0.000000000
## [85] 0.000000000 0.001727116
##
## $mids
##  [1]  4.5  5.5  6.5  7.5  8.5  9.5 10.5 11.5 12.5 13.5 14.5 15.5 16.5 17.5
18.5
## [16] 19.5 20.5 21.5 22.5 23.5 24.5 25.5 26.5 27.5 28.5 29.5 30.5 31.5 32.5
33.5
## [31] 34.5 35.5 36.5 37.5 38.5 39.5 40.5 41.5 42.5 43.5 44.5 45.5 46.5 47.5
48.5
## [46] 49.5 50.5 51.5 52.5 53.5 54.5 55.5 56.5 57.5 58.5 59.5 60.5 61.5 62.5
63.5
## [61] 64.5 65.5 66.5 67.5 68.5 69.5 70.5 71.5 72.5 73.5 74.5 75.5 76.5 77.5
78.5
## [76] 79.5 80.5 81.5 82.5 83.5 84.5 85.5 86.5 87.5 88.5 89.5
##
## $xname
## [1] "liver$Age"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"

lines(seq(min(liver$Age), max(liver$Age), length = length(liver$Age)),
      mix.gam[["prob"]][1]*dGA(seq(min(liver$Age), max(liver$Age),
      length = length(liver$Age)), mu = mu.hat1, sigma = sigma.hat1),
      lty=1, lwd=3, col="blue")

lines(seq(min(liver$Age), max(liver$Age), length = length(liver$Age)),
      mix.gam[["prob"]][2]*dGA(seq(min(liver$Age), max(liver$Age),
      length = length(liver$Age)), mu = mu.hat2, sigma = sigma.hat2),
      lty = 2, lwd = 3, col = "red")

lines(seq(min(liver$Age), max(liver$Age), length = length(liver$Age)),
      mix.gam[["prob"]][1]*dGA(seq(min(liver$Age), max(liver$Age),
```

```
      length = length(liver$Age)), mu = mu.hat1, sigma = sigma.hat1)+
      mix.gam[["prob"]][2]*dRG(seq(min(liver$Age), max(liver$Age),
      length = length(liver$Age)), mu = mu.hat2, sigma = sigma.hat2),
      lty = 3, lwd = 3, col = "green")
```

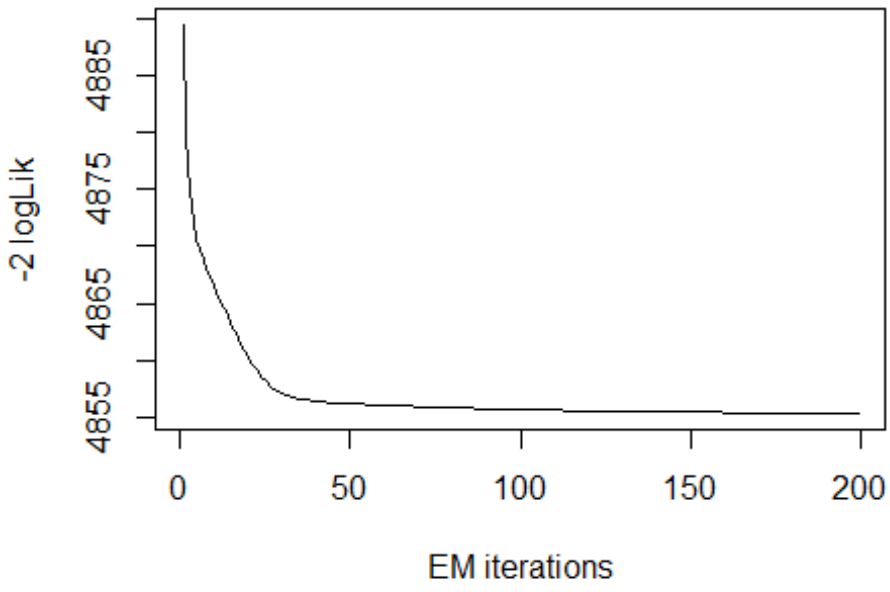


Now, I will again apply the fitting criteria for the distributions with Gamma mixture with the k = 3 as follows:

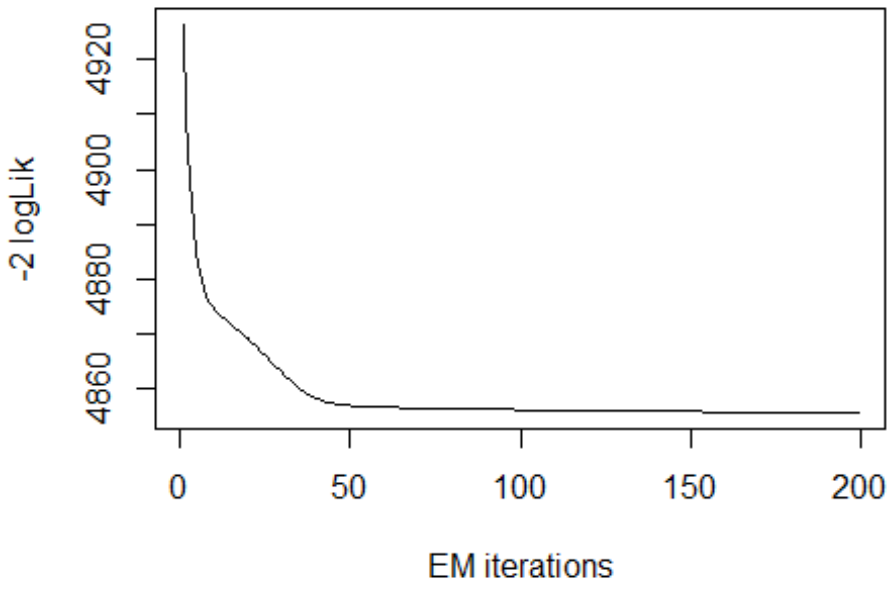
```
mix.gam.3 <- gamlssMXfits(n = 5, liver$Age~1, family = GA, K = 3, data =
liver)
```



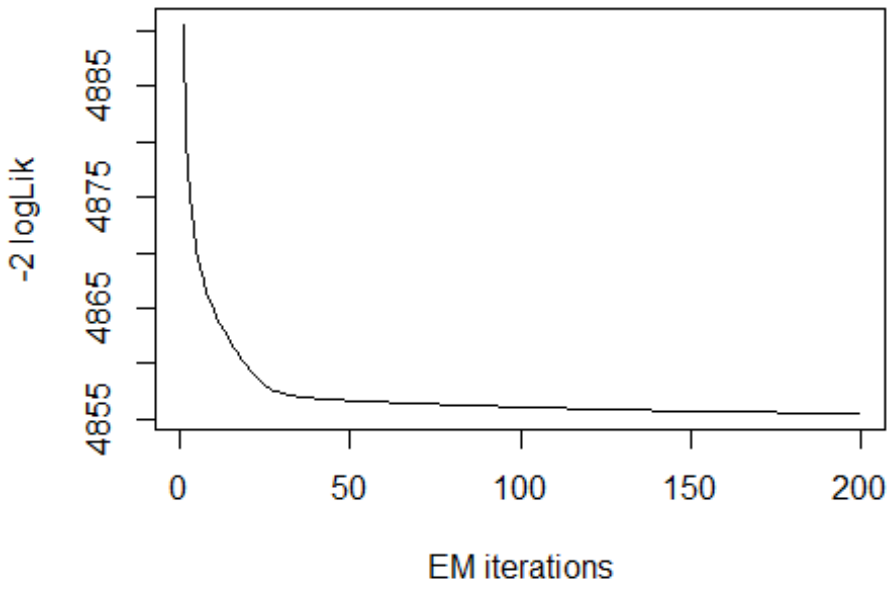

```
## model= 1
```



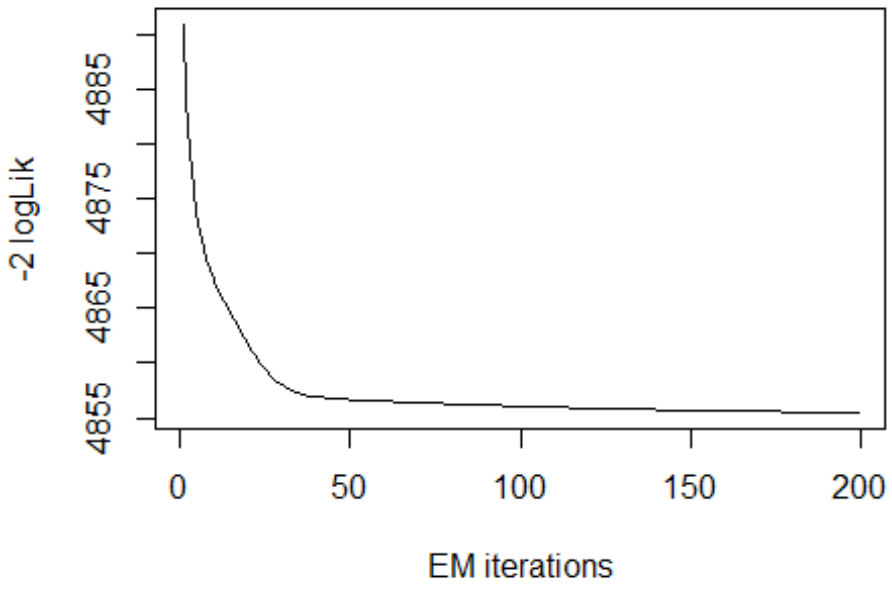

```
## model= 2
```




```
## model= 3
```




```
## model= 4
```




```
## model= 5

print(mix.gam.3)

##
## Mixing Family:  c("GA", "GA", "GA")
##
## Fitting method: EM algorithm
##
## Call:  gamlssMX(formula = liver$Age ~ 1, family = GA, K = 3,
##     data = liver)
##
## Mu Coefficients for model: 1
## (Intercept)
##       4.135
## Sigma Coefficients for model: 1
## (Intercept)
##      -2.036
## Mu Coefficients for model: 2
## (Intercept)
##       3.773
## Sigma Coefficients for model: 2
## (Intercept)
##      -1.297
## Mu Coefficients for model: 3
## (Intercept)
##        3.35
```

```
## Sigma Coefficients for model: 3
## (Intercept)
##     -0.6822
##
## Estimated probabilities: 0.2297243 0.5638963 0.2063795
##
## Degrees of Freedom for the fit: 8 Residual Deg. of Freedom   571
## Global Deviance:     4855.36
##             AIC:     4871.36
##             SBC:     4906.25
```

```
logLik(mix.gam.3)
```

```
## 'log Lik.' -2427.682 (df=8)
```

```
mix.gam.3$prob
```

```
## [1] 0.2297243 0.5638963 0.2063795
```

```
fitted(mix.gam.3, "mu")[1]
```

```
## [1] 44.77739
```

```
fitted(mix.gam.3, "sigma")[2]
```

```
## [1] 44.77739
```

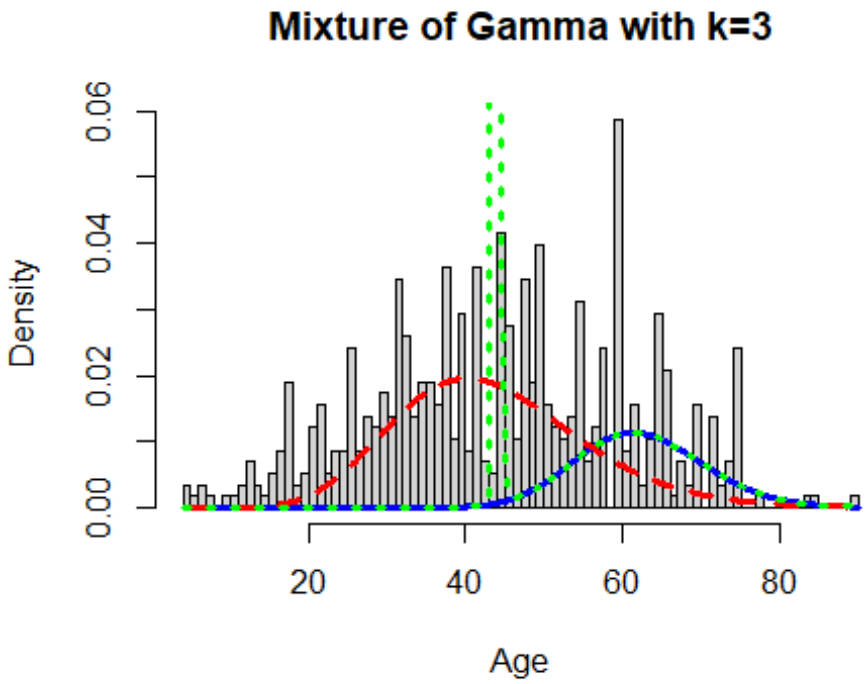
```
hist(liver$Age, breaks = 72, xlab = "Age", main="Mixture of Gamma with k=3",
freq = FALSE)

mu.hat1 <- exp(mix.gam.3[["models"]][[1]][["mu.coefficients"]])

sigma.hat1 <- exp(mix.gam.3[["models"]][[1]][["sigma.coefficients"]])

mu.hat2 <- exp(mix.gam.3[["models"]][[2]][["mu.coefficients"]])

sigma.hat2 <- exp(mix.gam.3[["models"]][[2]][["sigma.coefficients"]])

hist(liver$Age, breaks = 72, freq = FALSE, plot = FALSE)
```

```
## Warning in hist.default(liver$Age, breaks = 72, freq = FALSE, plot =
FALSE):
## argument 'freq' is not made use of

## $breaks
##  [1]  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
27 28
## [26] 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
52 53
## [51] 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76
77 78
## [76] 79 80 81 82 83 84 85 86 87 88 89 90
```

```
##
## $counts
##  [1]  2  1  2  1  0  1  1  2  4  2  1  3  5 11  2  3  7  9  3  5  5 14  5
8  7
## [26] 10  8 20 15  8 11 11  9 21  6 17  5 21  4  3 24 16  6 20 11 23  9  7
6  8
## [51] 18  4  7 14  0 34  5  9  2  6 17 12  1  4  2  9  0  8  2  4 14  0  0
1  0
## [76]  0  0  0  0  1  1  0  0  0  0  1
##
## $density
##  [1] 0.003454231 0.001727116 0.003454231 0.001727116 0.000000000
0.001727116
##  [7] 0.001727116 0.003454231 0.006908463 0.003454231 0.001727116
0.005181347
## [13] 0.008635579 0.018998273 0.003454231 0.005181347 0.012089810
0.015544041
## [19] 0.005181347 0.008635579 0.008635579 0.024179620 0.008635579
0.013816926
## [25] 0.012089810 0.017271157 0.013816926 0.034542314 0.025906736
0.013816926
## [31] 0.018998273 0.018998273 0.015544041 0.036269430 0.010362694
0.029360967
## [37] 0.008635579 0.036269430 0.006908463 0.005181347 0.041450777
0.027633851
## [43] 0.010362694 0.034542314 0.018998273 0.039723661 0.015544041
0.012089810
## [49] 0.010362694 0.013816926 0.031088083 0.006908463 0.012089810
0.024179620
## [55] 0.000000000 0.058721934 0.008635579 0.015544041 0.003454231
0.010362694
## [61] 0.029360967 0.020725389 0.001727116 0.006908463 0.003454231
0.015544041
## [67] 0.000000000 0.013816926 0.003454231 0.006908463 0.024179620
0.000000000
## [73] 0.000000000 0.001727116 0.000000000 0.000000000 0.000000000
0.000000000
## [79] 0.000000000 0.001727116 0.001727116 0.000000000 0.000000000
0.000000000
## [85] 0.000000000 0.001727116
##
## $mids
##  [1]  4.5  5.5  6.5  7.5  8.5  9.5 10.5 11.5 12.5 13.5 14.5 15.5 16.5 17.5
18.5
## [16] 19.5 20.5 21.5 22.5 23.5 24.5 25.5 26.5 27.5 28.5 29.5 30.5 31.5 32.5
33.5
## [31] 34.5 35.5 36.5 37.5 38.5 39.5 40.5 41.5 42.5 43.5 44.5 45.5 46.5 47.5
48.5
## [46] 49.5 50.5 51.5 52.5 53.5 54.5 55.5 56.5 57.5 58.5 59.5 60.5 61.5 62.5
63.5
```

```
## [61] 64.5 65.5 66.5 67.5 68.5 69.5 70.5 71.5 72.5 73.5 74.5 75.5 76.5 77.5
78.5
## [76] 79.5 80.5 81.5 82.5 83.5 84.5 85.5 86.5 87.5 88.5 89.5
##
## $xname
## [1] "liver$Age"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```
lines(seq(min(liver$Age), max(liver$Age), length = length(liver$Age)),
      mix.gam.3[["prob"]][1]*dGA(seq(min(liver$Age), max(liver$Age),
      length = length(liver$Age)), mu = mu.hat1, sigma = sigma.hat1),
      lty=1, lwd=3, col="blue")

lines(seq(min(liver$Age), max(liver$Age), length = length(liver$Age)),
      mix.gam.3[["prob"]][2]*dGA(seq(min(liver$Age), max(liver$Age),
      length = length(liver$Age)), mu = mu.hat2, sigma = sigma.hat2),
      lty = 2, lwd = 3, col = "red")

lines(seq(min(liver$Age), max(liver$Age), length = length(liver$Age)),
      mix.gam.3[["prob"]][1]*dGA(seq(min(liver$Age), max(liver$Age),
      length = length(liver$Age)), mu = mu.hat1, sigma = sigma.hat1)+
      mix.gam.3[["prob"]][2]*dRG(seq(min(liver$Age), max(liver$Age),
      length = length(liver$Age)), mu = mu.hat2, sigma = sigma.hat2),
      lty = 3, lwd = 3, col = "green")
```




```
mix.gm.tb <- data.frame(row.names=c('Gamma Mixture, K=3','Gamma Mixture,
K=2', "Generalized Gamma"),
                        AIC=c(mix.gam.3$aic, mix.gam$aic, AIC(age.GG)),
                        BIC=c(mix.gam.3$sbc, mix.gam$sbc, age.GG$sbc))
mix.gm.tb
```

```
##                           AIC      BIC
## Gamma Mixture, K=3 4871.363 4906.254
## Gamma Mixture, K=2 4883.443 4905.250
## Generalized Gamma  4862.331 4875.415
```

We can observe that the AIC value of the mixture of Gamma with k=3 and k=2 has increased, since values are higher than that of the single Gamma distribution. The previous AIC value is 4862, whereas the current value which is higher is 4871 and the previous BIC value was 4875, whereas the current value which is higher is 4906.

Gender

Lets analyze the features of Gender variable as follows:

```
length(liver$Gender)
```

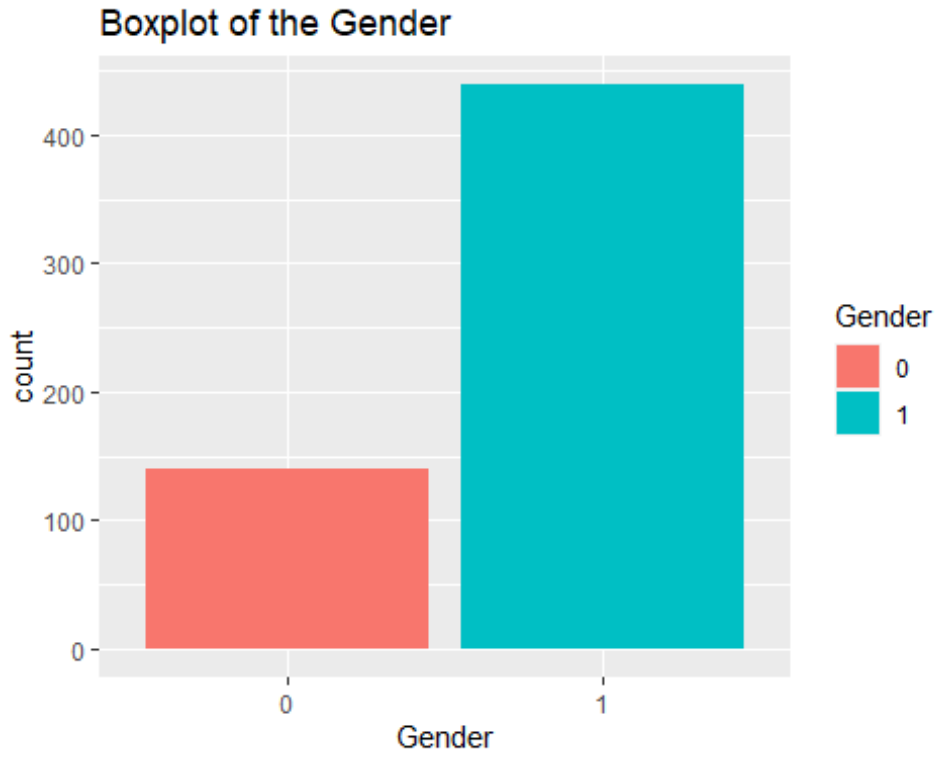
```
## [1] 579
```

```
table(liver$Gender)
```

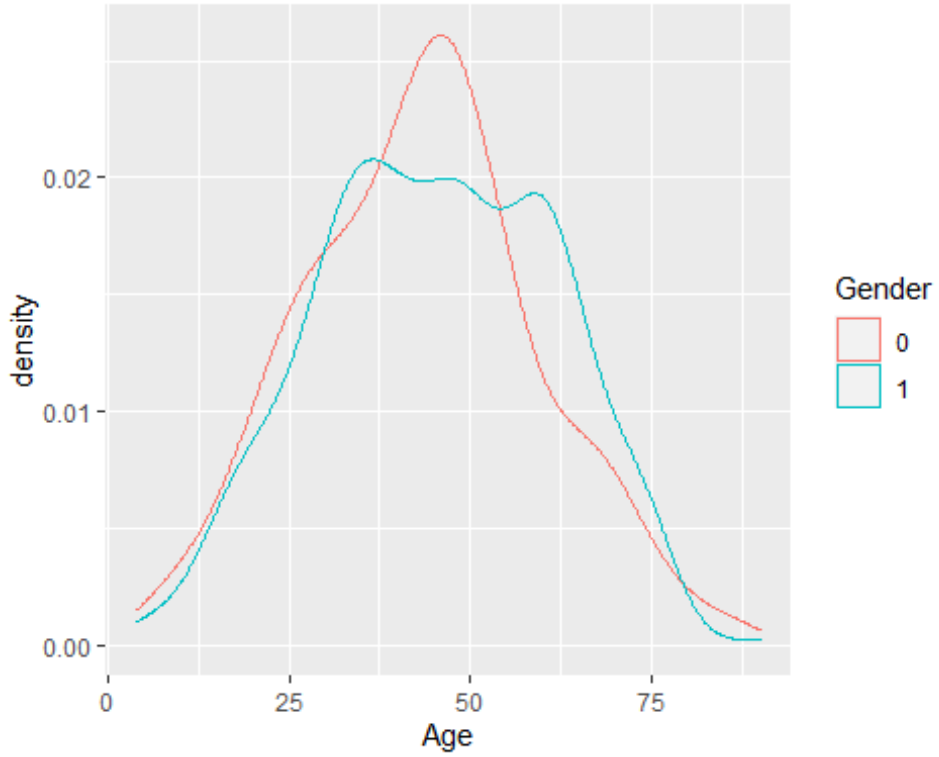
```
##
##   0   1
## 140 439
```

The table() function in R returns the absolute frequencies for each patient-specified value in the data set.Here we can see the `Gender` variable is a categorical variable that can take two values `0` for `Female` and `1` for `Male`.

```
library(ggplot2)
ggplot(liver, aes(x = Gender, fill = Gender)) +
geom_bar() +
  ggtitle("Boxplot of the Gender")
```

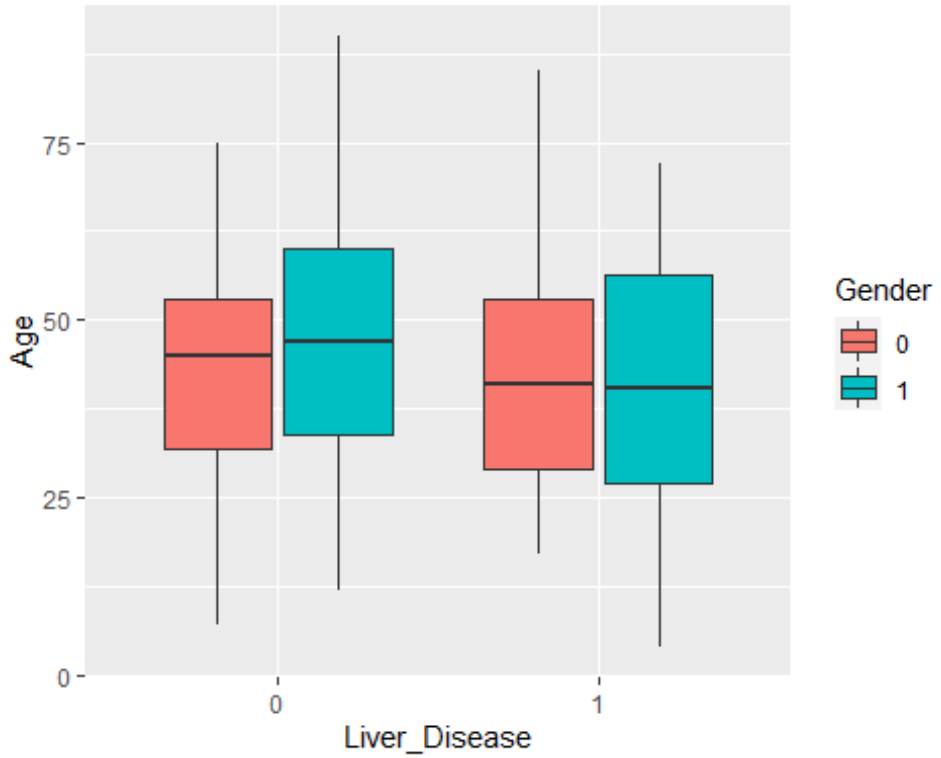



```
ggplot(liver, aes(x=Age, color=Gender)) +
  geom_density()
```




```r
liver$Gender[liver$Gender == "Female"] <- 0
liver$Gender[liver$Gender == "Male"] <- 1


ggplot(liver, aes(x = Gender, y = Age, fill = Gender)) +
  geom_boxplot() +
  ylab("Age") +
  ggtitle("Boxplot of the Gender across the Age")
```


```
ggplot(liver, aes(Liver_Disease, Age)) +
  geom_boxplot(aes(fill = Gender))
```



There are more males than females in the entire dataset. The mean age is higher for males than females in the data, both genders have a similar density shape when you look at the age. There are a few graphical ways of looking at this just to make sure we are doing the work right. When looking at the breakdown of the Age, Gender and Liver_Disease we see some interesting things. When we look at the females, the mean age is higher for the Liver_Disease = 0 and the same is true for Liver_Disease = 1. Within the Liver_Disease = 1 group the mean ages are pretty close but in the Liver_Disease = 0 ,the males have a higher mean age. We can also seen that the barplot shows our data set contains data belonging to 140 Females and 439 Males. So from both plot and table() we can see that our data is not balanced and the study includes more Males than Females.

Total Bilirubin

Lets explore the `Total_Bilirubin` variable:

```
head(liver$Total_Bilirubin)

## [1]  0.7 10.9  7.3  1.0  3.9  1.8

length(liver$Total_Bilirubin)

## [1] 579

table(liver$Total_Bilirubin)

##
##  0.4  0.5  0.6  0.7  0.8  0.9    1  1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8
1.9
##    1    5   45   77   90   56   28   19    8   11   13    5    8   11   14
8
##    2  2.1  2.2  2.3  2.4  2.5  2.6  2.7  2.8  2.9    3  3.1  3.2  3.3  3.4
3.5
##    8    4    8    4    5    2    5    9    4    6    2    2    3    3    1
3
##  3.6  3.7  3.8  3.9    4  4.1  4.2  4.4  4.5  4.7  4.9    5  5.2  5.3  5.5
5.7
##    2    3    1    4    3    2    2    1    2    1    1    2    1    2    1
1
##  5.8  5.9  6.2  6.3  6.6  6.7  6.8  7.1  7.3  7.4  7.5  7.7  7.9    8  8.2
8.6
##    5    1    1    1    1    2    4    2    3    1    1    1    1    1    1
1
##  8.7  8.9  9.4 10.2 10.6 10.9   11 11.1 11.3 11.5 12.1 12.7 14.1 14.2 14.5
14.8
##    1    3    1    1    1    2    1    1    1    1    1    2    1    1    1
1
##   15 15.2 15.6 15.8 15.9 16.4 16.6 16.7 17.3 17.7   18 18.4 18.5 19.6 19.8
20
##    1    1    1    1    1    1    1    1    1    1    1    2    1    1    1
1
```

```
## 20.2 22.5 22.6 22.7 22.8   23 23.2 23.3   25 26.3 27.2 27.7 30.5 30.8 32.6
42.8
##    1    1    1    1    1    1    1    1    1    1    1    1    2    1    1
1
##   75
##    1
```

The table() function in R returns the absolute frequencies for each patient-specified value in the data set.

```
unique(liver$Total_Bilirubin)
```

```
##   [1]  0.7 10.9  7.3  1.0  3.9  1.8  0.9  0.6  2.7  1.1  1.6  2.2  2.9
6.8  1.9
##  [16]  4.1  6.2  4.0  2.6  1.3 14.2  1.4  2.4 18.4  3.1  8.9  0.8  2.8
2.0  5.7
##  [31]  8.6  5.8  5.2  3.8  6.6  0.5  5.3  3.2  1.2 12.7 15.9 18.0 23.0
22.7  1.7
##  [46]  3.0 11.3  4.7  4.2  3.5  5.9  8.7 11.0 11.5  4.5 75.0 22.8 14.1
14.8 10.6
##  [61]  8.0  1.5  2.1  6.3  2.3 27.2  2.5  3.6 30.5 16.4 14.5 18.5 23.2
3.7  3.3
##  [76]  7.1  6.7 22.6  7.5  5.0  4.9  8.2  0.4  7.4 23.3  7.9  3.4 19.8
32.6 17.7
##  [91] 20.0 26.3  4.4  9.4 30.8 19.6 15.8  5.5 20.2 27.7 11.1 10.2 42.8
15.2 16.6
## [106] 17.3 22.5 16.7  7.7 15.6 12.1 25.0 15.0
```

```
length(unique(liver$Total_Bilirubin))
```

```
## [1] 113
```

```
min(liver$Total_Bilirubin)
```

```
## [1] 0.4
```

```
max(liver$Total_Bilirubin)
```

```
## [1] 75
```

Here the total observation of Age is 579 and is a continuous variable that range lies between 0.4-75. Now, we will see the summary of the Total_Bilirubin variable, for further analysis as follows:

```
summary(liver$Total_Bilirubin)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.400   0.800   1.000   3.315   2.600  75.000
```

```
sd(liver$Total_Bilirubin)
```

```
## [1] 6.227716
```

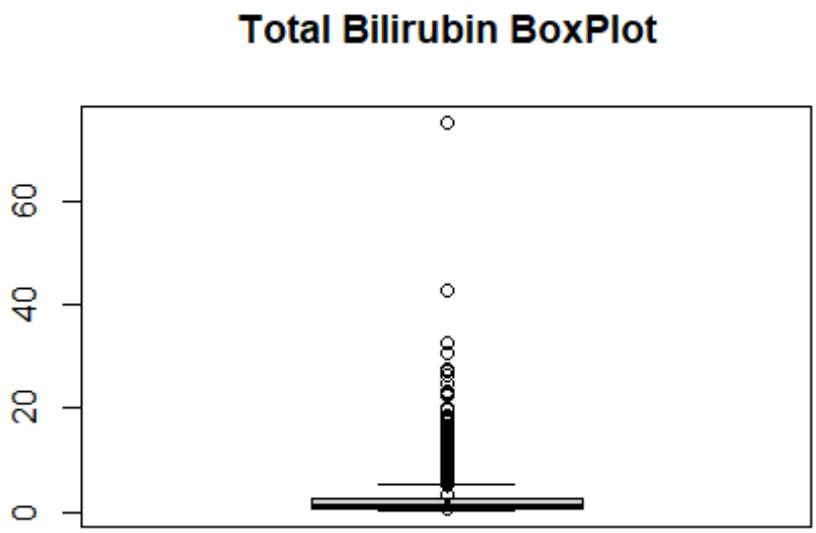
```r
var(liver$Total_Bilirubin)

## [1] 38.78445

getMode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
v <- c(liver$Total_Bilirubin)
mode <- getMode(v)
print(mode)

## [1] 0.8
```

From the above summary we can observe that the Mean (3.315), Median (1.000) and Mode (0.8) are not equal so the distribution is asymmetrical. Here Mean > Median > Mode so the distribution could be positive and skewed to the right. Now we also confirm this as follows:

```r
library(moments)
skewness(liver$Total_Bilirubin)

## [1] 4.878088
```

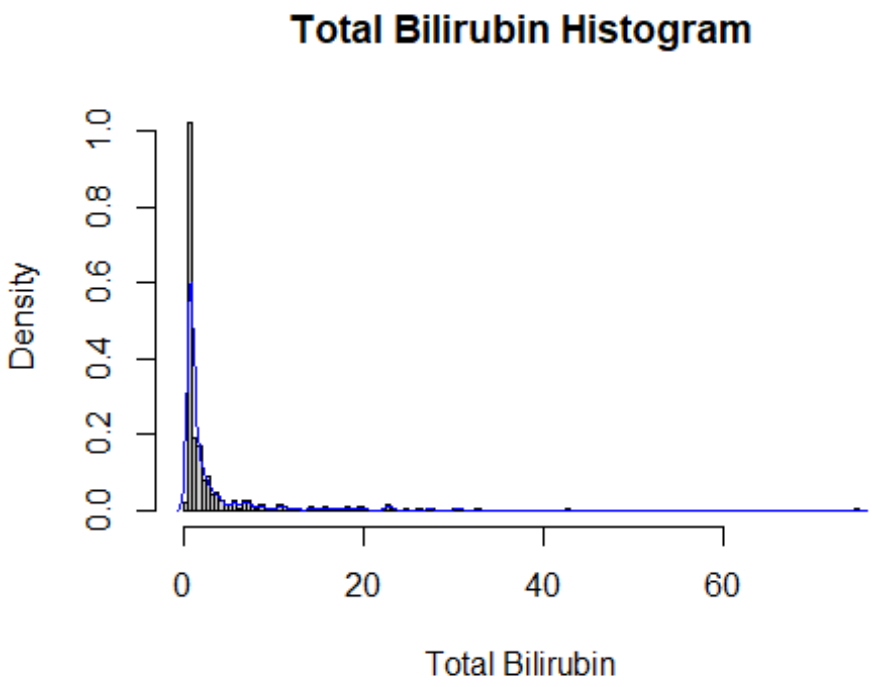
Hence, the skewness is the negative so the distribution is positively skewed was rightly observed. Also I will plot the Box plot to comparing the distribution of data across dataset as follows:

```r
boxplot(liver$Total_Bilirubin, main = "Total Bilirubin BoxPlot")
points(mean(liver$Total_Bilirubin))
points(mode)
```



Box plots are graphical displays of the five number summary, So they describe the Minimum (lowest mean relative) lower whisker, Quartile 1 (It separates the lowest 25% observation to the rest of the observations), Median (Which divides the lower 50% observation from the upper 50% observations), Quartile 3 (It divides the upper 25% observation to the rest of the observations) and Maximum. In the above Box plot diagram, there are many outliers and the upper whisker shows the highest mean relative. Now, I will plot a histogram a graphical display of data using bars of different heights to measures distribution of continuous data as follows:

```
hist(liver$Total_Bilirubin, prob = TRUE, breaks = 113, xlab = "Total
Bilirubin", main = "Total Bilirubin Histogram")
lines(density(liver$Total_Bilirubin), col='blue')
```

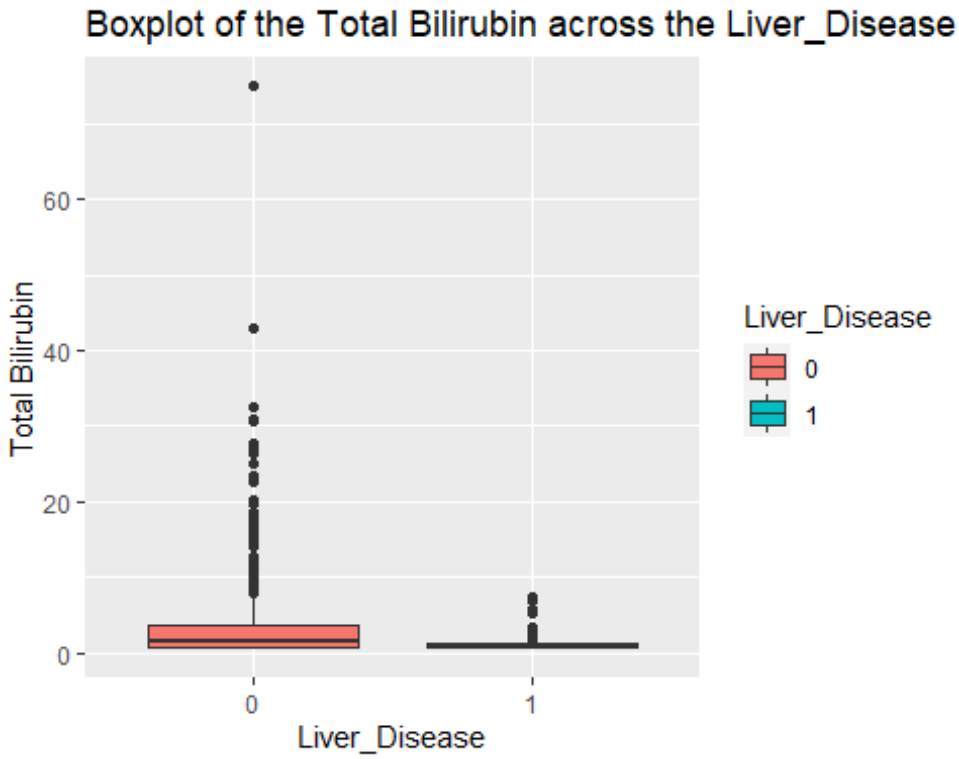


From the above histogram, we can observe that the diagram has many peaks.Using the density lines, I have approximated the histogram graph. Density provides information about the type of distribution under consideration and allows us to determine whether the attribute is distributed normally or not. From the graph above we can see how the `Total_Bilirubin` variable does not seem to fit to a Normal distribution, there is a peak of the frequency distributions around many values and also a right skewed. We have already observed a positive value of `4.878088`, so we will affirm that the distribution is right skewed. To check whether the distribution is `Platykurtic` or not, we will check this by following R method as follows:

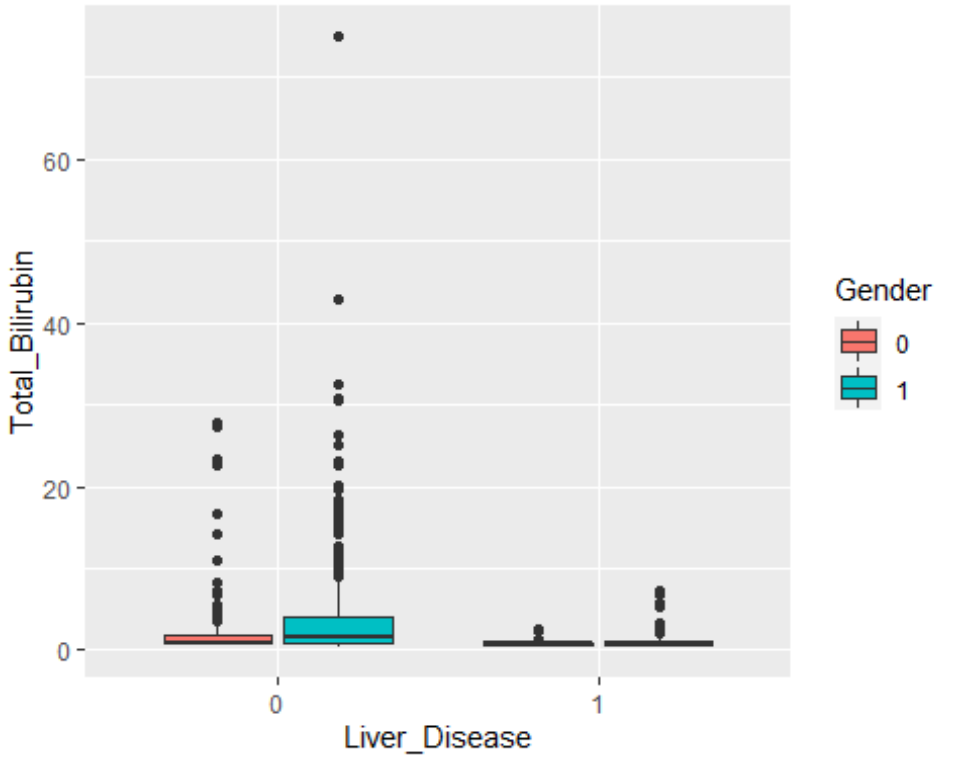
```
kurtosis(liver$Total_Bilirubin)

## [1] 39.59283
```

The distribution is also `leptokurtic`, since the value is greater than 3.

```
ggplot(liver, aes(x = Liver_Disease, y = Total_Bilirubin, fill =
Liver_Disease)) +
  geom_boxplot() +
  ylab("Total Bilirubin") +
  ggtitle("Boxplot of the Total Bilirubin across the Liver_Disease")
```




```
ggplot(liver, aes(Liver_Disease, Total_Bilirubin)) +
  geom_boxplot(aes(fill = Gender))
```

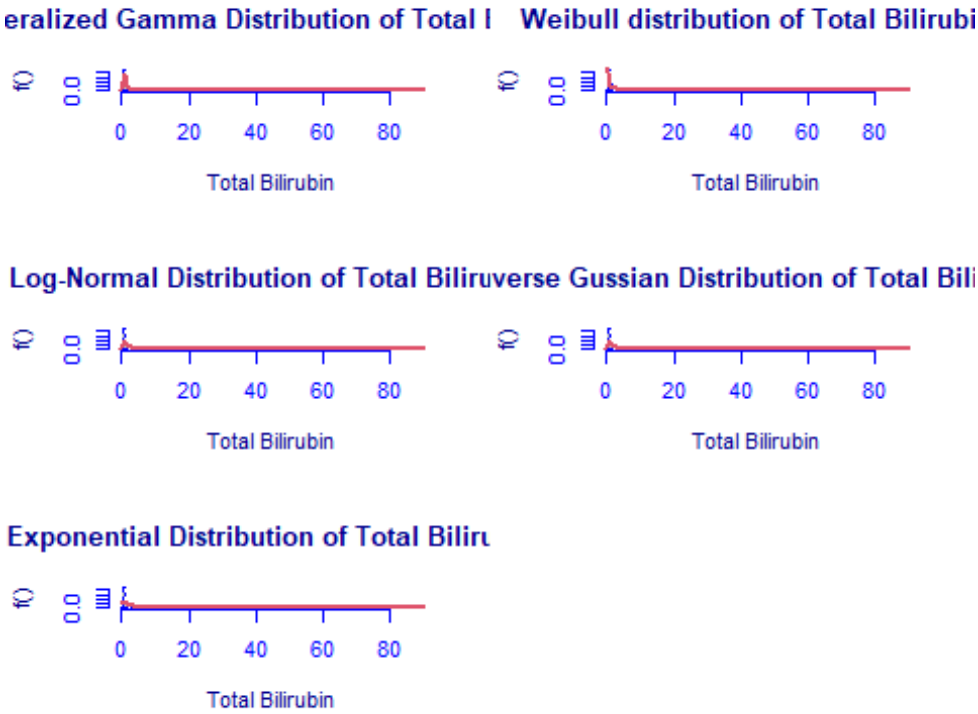


There is a wide range for the Total Bilirubin. The data is skewed, very skewed, we have a maximum value of 75 with the mean of 3.299, median of 1. There is a difference between the mean Total Bilirubin for the Liver_Disease and this also seems to be true when you break it down further by gender.

Total Bilirubin Fit for the Data

Now we will try to fit different models to Total_Bilirubin distribution evaluating the Akaike Information Criterion (AIC) and the Bayesian Information criterion (BIC), The lower are the indexes, the better is the fitting. I will evaluate both the single parameter distributions and mixture distributions as follows:

```
library(MASS)
library(gamlss)

par(mfrow=c(3,2))

tb.GG <- histDist(liver$Total_Bilirubin, family=GG, nbins = 113, xlab =
"Total Bilirubin", main="Generalized Gamma Distribution of Total Bilirubin")

tb.WEI <- histDist(liver$Total_Bilirubin, family=WEI, nbins = 113, xlab =
"Total Bilirubin", main="Weibull distribution of Total Bilirubin")

tb.LOGNO <- histDist(liver$Total_Bilirubin, family=LOGNO, nbins = 113, xlab =
"Total Bilirubin", main="Log-Normal Distribution of Total Bilirubin")

tb.IG <- histDist(liver$Total_Bilirubin, family=IG, nbins=113, xlab = "Total
Bilirubin", main = "Inverse Gussian Distribution of Total Bilirubin")

tb.EXP<- histDist(liver$Total_Bilirubin, family=EXP, nbins=113, xlab = "Total
Bilirubin", main = "Exponential Distribution of Total Bilirubin")
```




```
library(Matrix)
library(glmnet)
df <- data.frame(Rownames = c("Generalized Gamma", "Weibull", "Log-Normal",
                              "Inverse Gussian", "Exponential"),
                 AIC = c(AIC(tb.GG), AIC(tb.WEI), AIC(tb.LOGNO),
AIC(tb.IG),
                         AIC(tb.EXP)),
                 BIC = c(tb.GG$sbc, tb.WEI$sbc, tb.LOGNO$sbc, tb.IG$sbc,
                         tb.EXP$sbc),
                 df = c(tb.GG$df.fit, tb.WEI$df.fit, tb.LOGNO$df.fit,
                        tb.IG$df.fit, tb.EXP$df.fit),
                 LogLike = c(logLik(tb.GG), logLik(tb.WEI),
logLik(tb.LOGNO),
                             logLik(tb.IG), logLik(tb.EXP)))
df
```

```
##            Rownames      AIC      BIC df    LogLike
## 1 Generalized Gamma 1833.703 1846.787  3  -913.8516
## 2           Weibull 2481.176 2489.898  2 -1238.5878
## 3        Log-Normal 2211.160 2219.883  2 -1103.5801
## 4   Inverse Gussian 2143.455 2152.178  2 -1069.7275
## 5       Exponential 2547.944 2552.305  1 -1272.9718
```

As we can see, the model with the highest log likelihood (-913.8516) and the lowest AIC (1833.703) and BIC (1846.787) is the Generalized Gamma Distribution. Therefore, based on the greatest likelihood technique, our data fits better in the Generalized Gamma

Distribution. Now, I will also compare two models by means of the Likelihood ratio test as we performed above as follows:

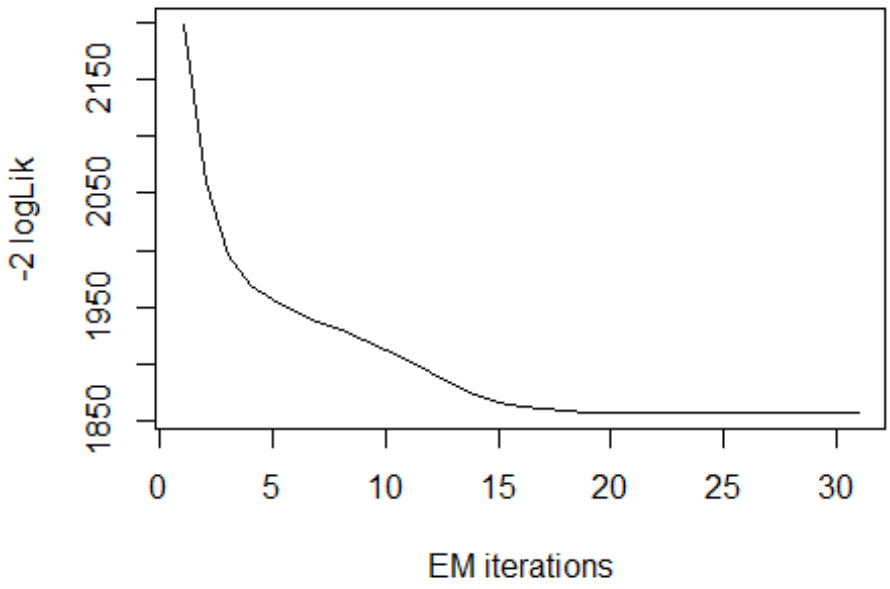
```
library(zoo)
library(lmtest)
lrtest(tb.GG, tb.EXP)

## Likelihood ratio test
##
## Model 1: gamlssML(formula = liver$Total_Bilirubin, family = "GG")
## Model 2: gamlssML(formula = liver$Total_Bilirubin, family = "EXP")
##   #Df   LogLik Df  Chisq Pr(>Chisq)
## 1   3  -913.85
## 2   1 -1272.97 -2 718.24  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Under the null hypothesis, we compare the Generalized Gamma distribution to the Exponential distribution under the alternative hypothesis. At any level of significance, we can reject the null hypothesis. Because our distributions fit better in the Generalized Gamma model, we'll use it.

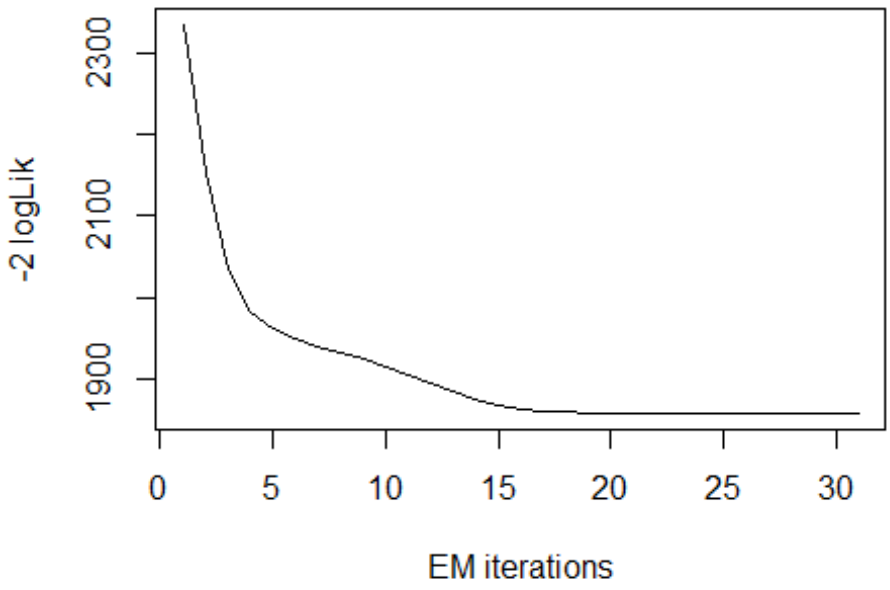
Distributions Mixture

Now, I will also apply the fitting criteria for the distributions with Gamma mixture as follows:

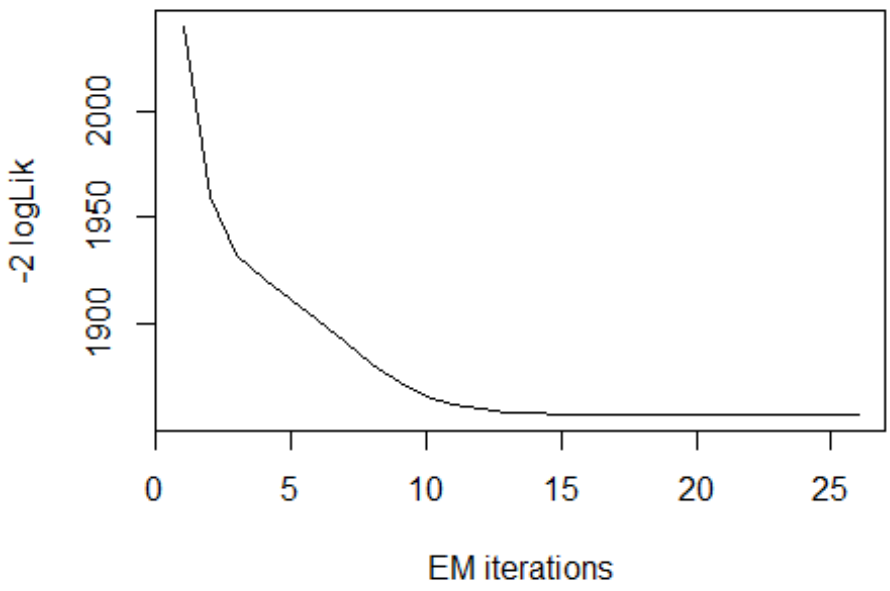
```
library(gamlss.mx)
mix.gam <- gamlssMXfits(n = 5, liver$Total_Bilirubin~1, family = GA, K = 2,
data = liver)
```



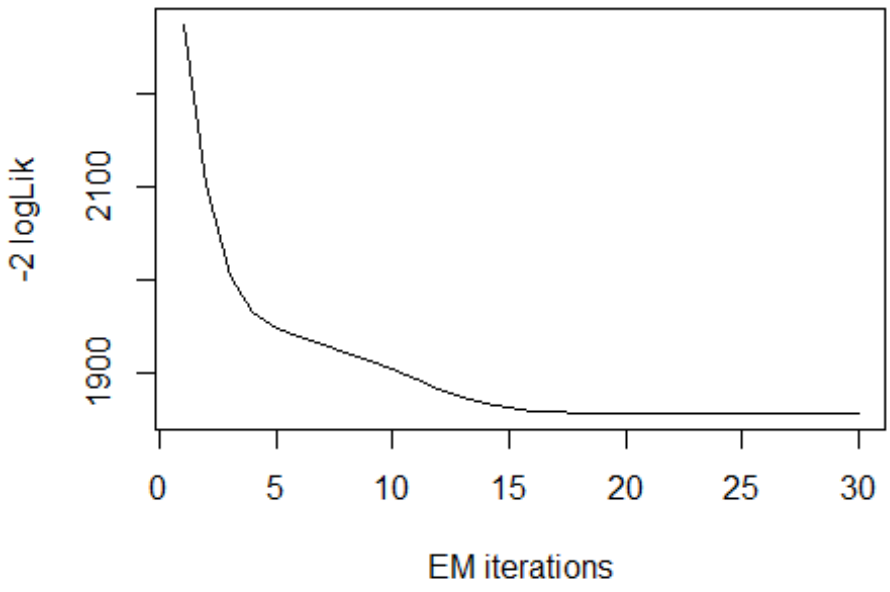

```
## model= 1
```



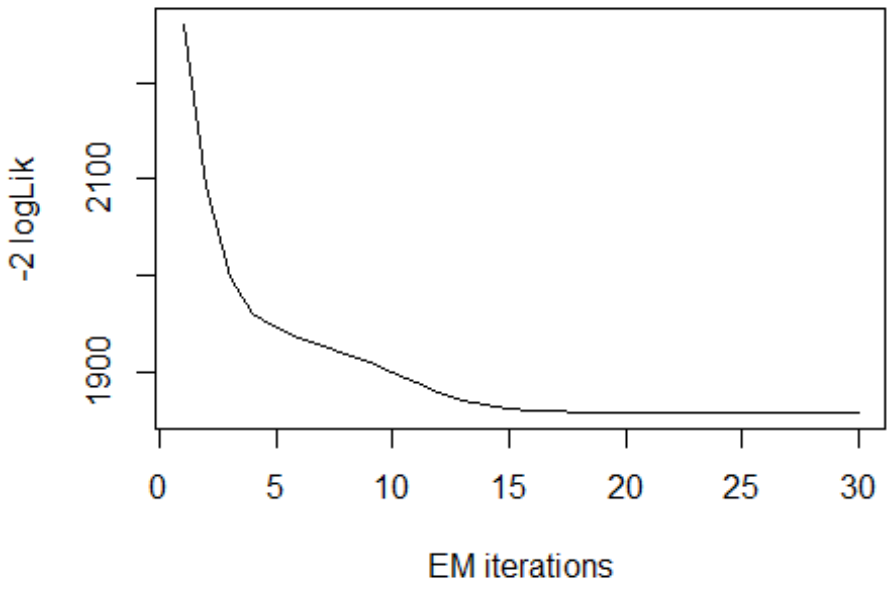

```
## model= 2
```




```
## model= 3
```




```
## model= 4
```




```
## model= 5

print(mix.gam)

##
## Mixing Family:  c("GA", "GA")
##
## Fitting method: EM algorithm
##
## Call:  gamlssMX(formula = liver$Total_Bilirubin ~ 1, family = GA,
##     K = 2, data = liver)
##
## Mu Coefficients for model: 1
## (Intercept)
##       1.795
## Sigma Coefficients for model: 1
## (Intercept)
##   -0.005117
## Mu Coefficients for model: 2
## (Intercept)
##     -0.2179
## Sigma Coefficients for model: 2
## (Intercept)
##      -1.693
##
## Estimated probabilities: 0.4813882 0.5186118
##
```

```
## Degrees of Freedom for the fit: 5 Residual Deg. of Freedom   574
## Global Deviance:     1857.52
##             AIC:     1867.52
##             SBC:     1889.32
```

We can observe that the AIC value of the mixture of Gamma has not improved, since it is higher than that of the single Gamma distribution. The current `AIC` value is `1867.52`, whereas the previous value was `1833.703` and the current `BIC` value is `1889.32`, whereas the previous value was `1846.787`.

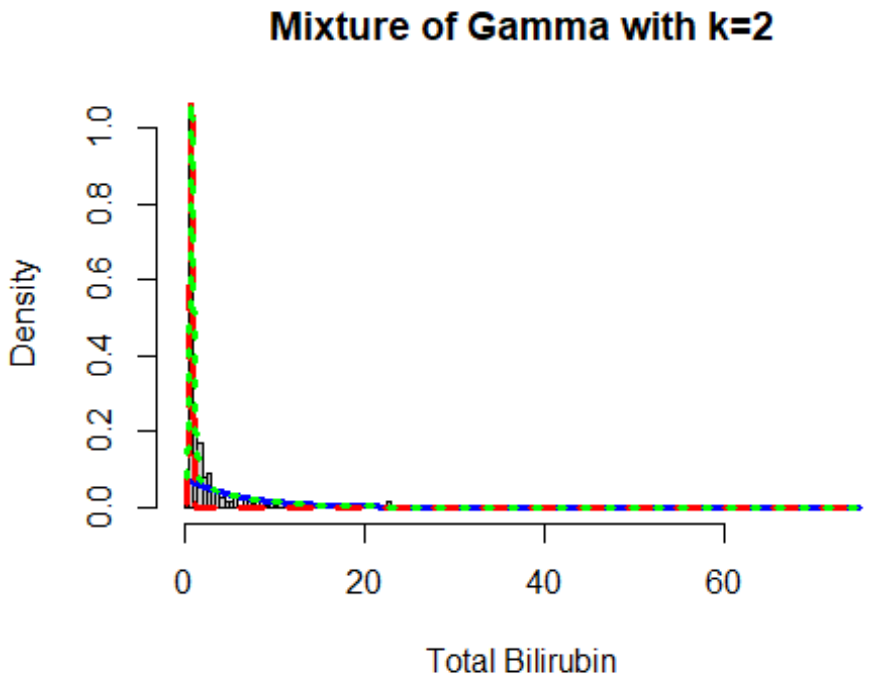
```
logLik(mix.gam)

## 'log Lik.' -928.7576 (df=5)

mix.gam$prob

## [1] 0.4813882 0.5186118

fitted(mix.gam, "mu")[1]

## [1] 3.315516

fitted(mix.gam, "sigma")[2]

## [1] 3.315516

hist(liver$Total_Bilirubin, breaks = 113, xlab = "Total Bilirubin",
main="Mixture of Gamma with k=2", freq = FALSE)

mu.hat1 <- exp(mix.gam[["models"]][[1]][["mu.coefficients"]])

sigma.hat1 <- exp(mix.gam[["models"]][[1]][["sigma.coefficients"]])

mu.hat2 <- exp(mix.gam[["models"]][[2]][["mu.coefficients"]])

sigma.hat2 <- exp(mix.gam[["models"]][[2]][["sigma.coefficients"]])

hist(liver$Total_Bilirubin, breaks = 113, freq = FALSE, plot = FALSE)

## Warning in hist.default(liver$Total_Bilirubin, breaks = 113, freq = FALSE,
:
## argument 'freq' is not made use of

## $breaks
##   [1]  0.0  0.5  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0
6.5  7.0
##  [16]  7.5  8.0  8.5  9.0  9.5 10.0 10.5 11.0 11.5 12.0 12.5 13.0 13.5
14.0 14.5
##  [31] 15.0 15.5 16.0 16.5 17.0 17.5 18.0 18.5 19.0 19.5 20.0 20.5 21.0
21.5 22.0
##  [46] 22.5 23.0 23.5 24.0 24.5 25.0 25.5 26.0 26.5 27.0 27.5 28.0 28.5
29.0 29.5
```

```
##  [61] 30.0 30.5 31.0 31.5 32.0 32.5 33.0 33.5 34.0 34.5 35.0 35.5 36.0
36.5 37.0
##  [76] 37.5 38.0 38.5 39.0 39.5 40.0 40.5 41.0 41.5 42.0 42.5 43.0 43.5
44.0 44.5
##  [91] 45.0 45.5 46.0 46.5 47.0 47.5 48.0 48.5 49.0 49.5 50.0 50.5 51.0
51.5 52.0
## [106] 52.5 53.0 53.5 54.0 54.5 55.0 55.5 56.0 56.5 57.0 57.5 58.0 58.5
59.0 59.5
## [121] 60.0 60.5 61.0 61.5 62.0 62.5 63.0 63.5 64.0 64.5 65.0 65.5 66.0
66.5 67.0
## [136] 67.5 68.0 68.5 69.0 69.5 70.0 70.5 71.0 71.5 72.0 72.5 73.0 73.5
74.0 74.5
## [151] 75.0
##
## $counts
##   [1]   6 296  56  49  23  26  12  13   7   4   4   7   2   7   7   3   1
5
##  [19]   1   0   1   4   3   0   1   2   0   0   3   2   1   3   1   2   1
2
##  [37]   3   0   0   3   1   0   0   0   1   4   2   0   0   1   0   0   1
0
##  [55]   1   1   0   0   0   0   2   1   0   0   0   1   0   0   0   0   0
0
##  [73]   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0
0
##  [91]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [109]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [127]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [145]   0   0   0   0   0   1
##
## $density
##   [1] 0.020725389 1.022452504 0.193436960 0.169257340 0.079447323
0.089810017
##   [7] 0.041450777 0.044905009 0.024179620 0.013816926 0.013816926
0.024179620
##  [13] 0.006908463 0.024179620 0.024179620 0.010362694 0.003454231
0.017271157
##  [19] 0.003454231 0.000000000 0.003454231 0.013816926 0.010362694
0.000000000
##  [25] 0.003454231 0.006908463 0.000000000 0.000000000 0.010362694
0.006908463
##  [31] 0.003454231 0.010362694 0.003454231 0.006908463 0.003454231
0.006908463
##  [37] 0.010362694 0.000000000 0.000000000 0.010362694 0.003454231
0.000000000
##  [43] 0.000000000 0.000000000 0.003454231 0.013816926 0.006908463
0.000000000
```

```
##  [49] 0.000000000 0.003454231 0.000000000 0.000000000 0.003454231
0.000000000
##  [55] 0.003454231 0.003454231 0.000000000 0.000000000 0.000000000
0.000000000
##  [61] 0.006908463 0.003454231 0.000000000 0.000000000 0.000000000
0.003454231
##  [67] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
##  [73] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
##  [79] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
##  [85] 0.000000000 0.003454231 0.000000000 0.000000000 0.000000000
0.000000000
##  [91] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
##  [97] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [103] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [109] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [115] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [121] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [127] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [133] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [139] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [145] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.003454231
##
## $mids
##   [1]  0.25  0.75  1.25  1.75  2.25  2.75  3.25  3.75  4.25  4.75  5.25
5.75
##  [13]  6.25  6.75  7.25  7.75  8.25  8.75  9.25  9.75 10.25 10.75 11.25
11.75
##  [25] 12.25 12.75 13.25 13.75 14.25 14.75 15.25 15.75 16.25 16.75 17.25
17.75
##  [37] 18.25 18.75 19.25 19.75 20.25 20.75 21.25 21.75 22.25 22.75 23.25
23.75
##  [49] 24.25 24.75 25.25 25.75 26.25 26.75 27.25 27.75 28.25 28.75 29.25
29.75
##  [61] 30.25 30.75 31.25 31.75 32.25 32.75 33.25 33.75 34.25 34.75 35.25
35.75
##  [73] 36.25 36.75 37.25 37.75 38.25 38.75 39.25 39.75 40.25 40.75 41.25
41.75
```

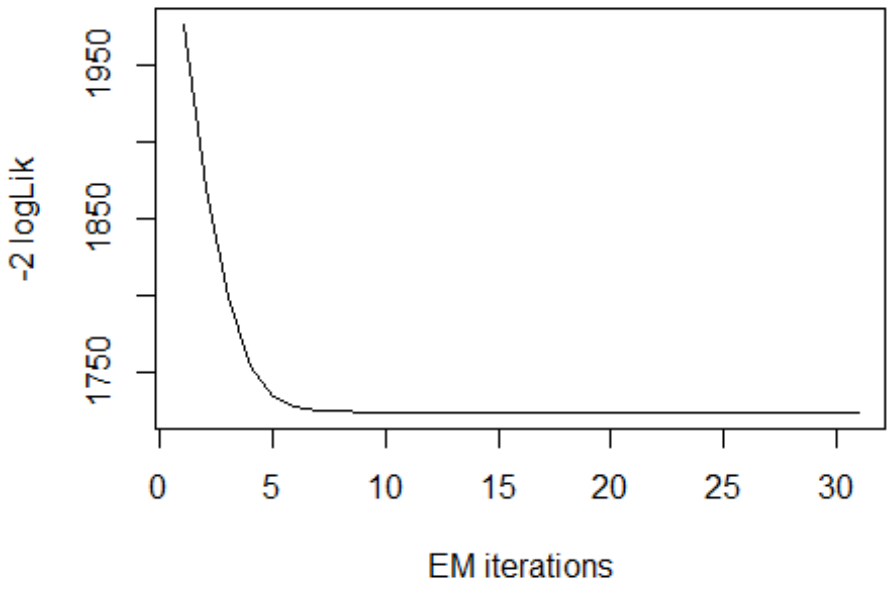
```
##  [85] 42.25 42.75 43.25 43.75 44.25 44.75 45.25 45.75 46.25 46.75 47.25
47.75
##  [97] 48.25 48.75 49.25 49.75 50.25 50.75 51.25 51.75 52.25 52.75 53.25
53.75
## [109] 54.25 54.75 55.25 55.75 56.25 56.75 57.25 57.75 58.25 58.75 59.25
59.75
## [121] 60.25 60.75 61.25 61.75 62.25 62.75 63.25 63.75 64.25 64.75 65.25
65.75
## [133] 66.25 66.75 67.25 67.75 68.25 68.75 69.25 69.75 70.25 70.75 71.25
71.75
## [145] 72.25 72.75 73.25 73.75 74.25 74.75
##
## $xname
## [1] "liver$Total_Bilirubin"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```
lines(seq(min(liver$Total_Bilirubin), max(liver$Total_Bilirubin),
      length = length(liver$Total_Bilirubin)),
      mix.gam[["prob"]][1]*dGA(seq(min(liver$Total_Bilirubin),
      max(liver$Total_Bilirubin), length = length(liver$Total_Bilirubin)),
      mu = mu.hat1, sigma = sigma.hat1), lty=1, lwd=3, col="blue")

lines(seq(min(liver$Total_Bilirubin), max(liver$Total_Bilirubin),
      length = length(liver$Total_Bilirubin)),
      mix.gam[["prob"]][2]*dGA(seq(min(liver$Total_Bilirubin),
      max(liver$Total_Bilirubin), length = length(liver$Total_Bilirubin)),
      mu = mu.hat2, sigma = sigma.hat2), lty = 2, lwd = 3, col = "red")

lines(seq(min(liver$Total_Bilirubin), max(liver$Total_Bilirubin),
      length = length(liver$Total_Bilirubin)),
      mix.gam[["prob"]][1]*dGA(seq(min(liver$Total_Bilirubin),
      max(liver$Total_Bilirubin), length = length(liver$Total_Bilirubin)),
      mu = mu.hat1, sigma = sigma.hat1)+
      mix.gam[["prob"]][2]*dRG(seq(min(liver$Total_Bilirubin),
      max(liver$Total_Bilirubin), length = length(liver$Total_Bilirubin)),
      mu = mu.hat2, sigma = sigma.hat2), lty = 3, lwd = 3, col = "green")
```

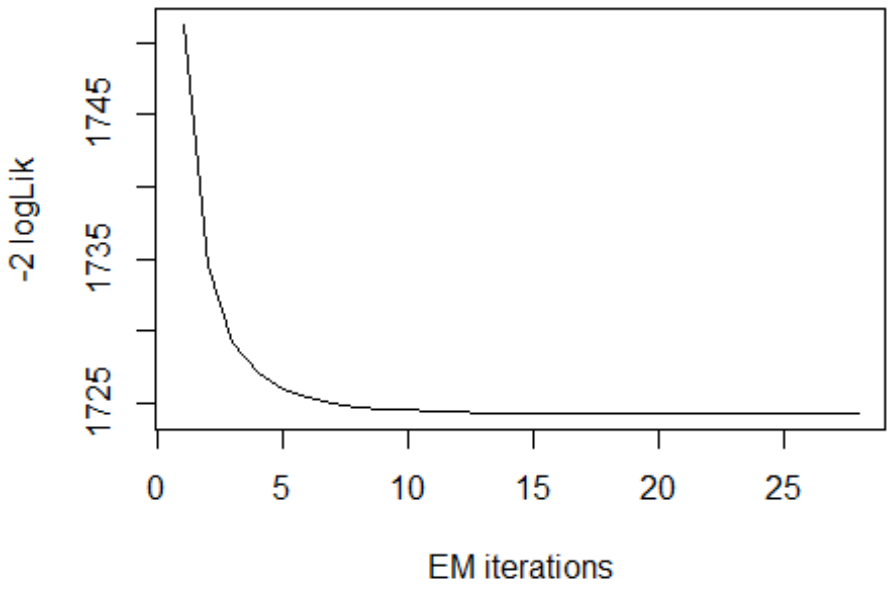


Now, I will again apply the fitting criteria for the distributions with Gamma mixture with the k = 3 as follows:

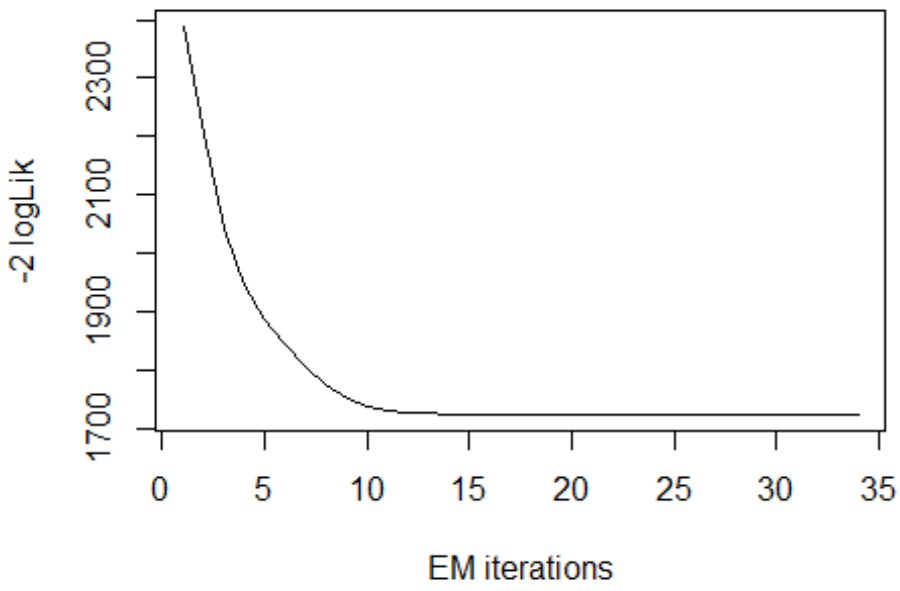
```
mix.gam.3 <- gamlssMXfits(n = 5, liver$Total_Bilirubin~1, family = GA, K = 3,
                          data = liver)
```



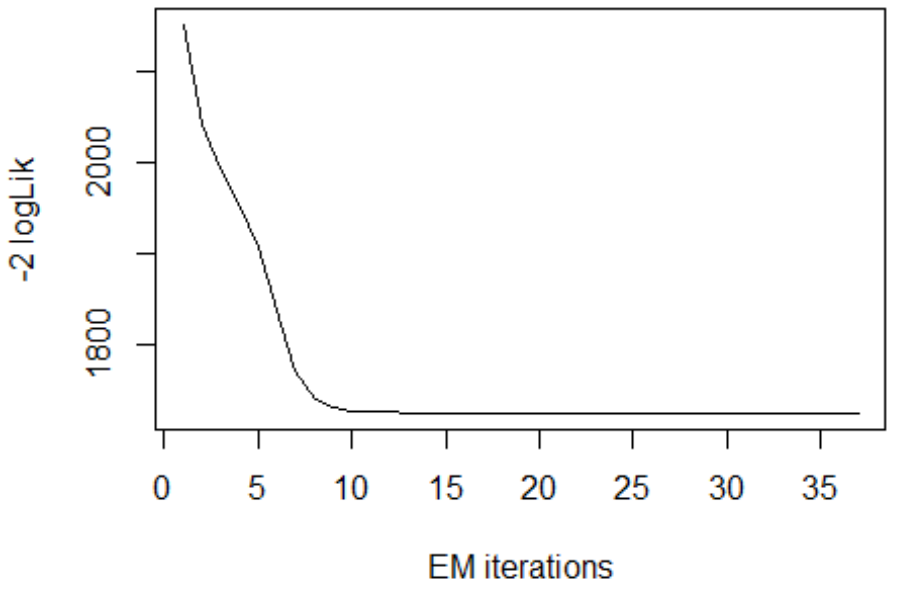

```
## model= 1
```



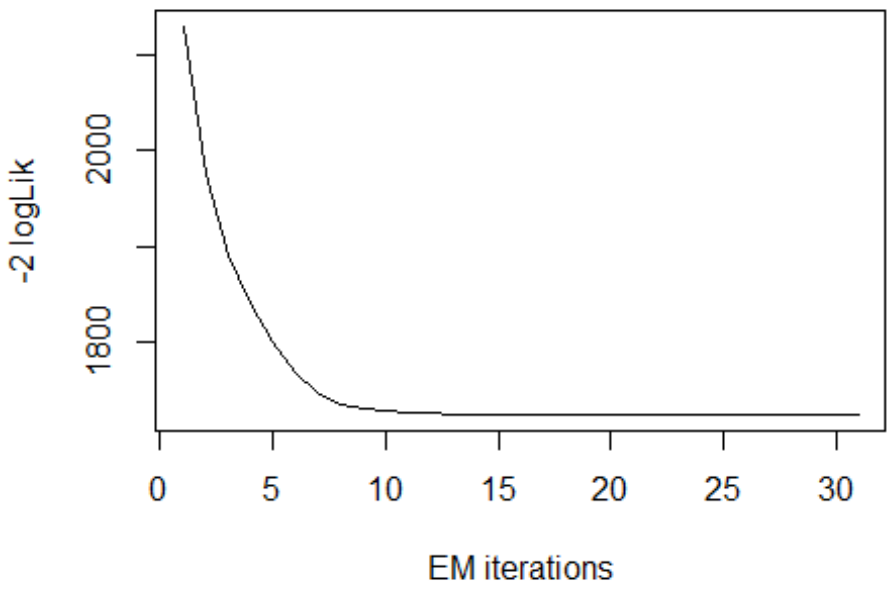

```
## model= 2
```




```
## model= 3
```




```
## model= 4
```




```
## model= 5

print(mix.gam.3)

##
## Mixing Family:  c("GA", "GA", "GA")
##
## Fitting method: EM algorithm
##
## Call:  gamlssMX(formula = liver$Total_Bilirubin ~ 1, family = GA,
##     K = 3, data = liver)
##
## Mu Coefficients for model: 1
## (Intercept)
##     -0.2409
## Sigma Coefficients for model: 1
## (Intercept)
##      -1.765
## Mu Coefficients for model: 2
## (Intercept)
##      0.6743
## Sigma Coefficients for model: 2
## (Intercept)
##     -0.7985
## Mu Coefficients for model: 3
## (Intercept)
##       2.433
```

```
## Sigma Coefficients for model: 3
## (Intercept)
##     -0.2131
##
## Estimated probabilities: 0.4978339 0.2966577 0.2055084
##
## Degrees of Freedom for the fit: 8 Residual Deg. of Freedom   571
## Global Deviance:     1724.3
##             AIC:     1740.3
##             SBC:     1775.19
```

```
logLik(mix.gam.3)
```

```
## 'log Lik.' -862.1487 (df=8)
```

```
mix.gam.3$prob
```

```
## [1] 0.4978339 0.2966577 0.2055084
```

```
fitted(mix.gam.3, "mu")[1]
```

```
## [1] 3.315623
```

```
fitted(mix.gam.3, "sigma")[2]
```

```
## [1] 3.315623
```

```
hist(liver$Total_Bilirubin, breaks = 113, xlab = "Total Bilirubin",
main="Mixture of Gamma with k=3", freq = FALSE)
```

```
mu.hat1 <- exp(mix.gam.3[["models"]][[1]][["mu.coefficients"]])
```

```
sigma.hat1 <- exp(mix.gam.3[["models"]][[1]][["sigma.coefficients"]])
```

```
mu.hat2 <- exp(mix.gam.3[["models"]][[2]][["mu.coefficients"]])
```

```
sigma.hat2 <- exp(mix.gam.3[["models"]][[2]][["sigma.coefficients"]])
```

```
hist(liver$Total_Bilirubin, breaks = 113, freq = FALSE, plot = FALSE)
```

```
## Warning in hist.default(liver$Total_Bilirubin, breaks = 113, freq = FALSE,
:
## argument 'freq' is not made use of
```

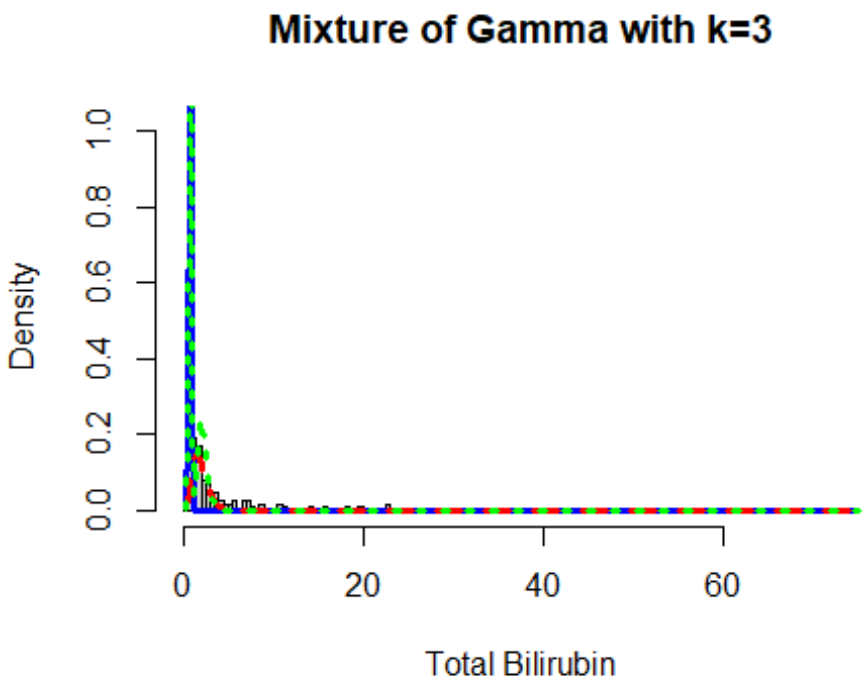
```
## $breaks
##   [1]  0.0  0.5  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0
6.5  7.0
##  [16]  7.5  8.0  8.5  9.0  9.5 10.0 10.5 11.0 11.5 12.0 12.5 13.0 13.5
14.0 14.5
##  [31] 15.0 15.5 16.0 16.5 17.0 17.5 18.0 18.5 19.0 19.5 20.0 20.5 21.0
21.5 22.0
##  [46] 22.5 23.0 23.5 24.0 24.5 25.0 25.5 26.0 26.5 27.0 27.5 28.0 28.5
```

```
29.0 29.5
##  [61] 30.0 30.5 31.0 31.5 32.0 32.5 33.0 33.5 34.0 34.5 35.0 35.5 36.0
36.5 37.0
##  [76] 37.5 38.0 38.5 39.0 39.5 40.0 40.5 41.0 41.5 42.0 42.5 43.0 43.5
44.0 44.5
##  [91] 45.0 45.5 46.0 46.5 47.0 47.5 48.0 48.5 49.0 49.5 50.0 50.5 51.0
51.5 52.0
## [106] 52.5 53.0 53.5 54.0 54.5 55.0 55.5 56.0 56.5 57.0 57.5 58.0 58.5
59.0 59.5
## [121] 60.0 60.5 61.0 61.5 62.0 62.5 63.0 63.5 64.0 64.5 65.0 65.5 66.0
66.5 67.0
## [136] 67.5 68.0 68.5 69.0 69.5 70.0 70.5 71.0 71.5 72.0 72.5 73.0 73.5
74.0 74.5
## [151] 75.0
##
## $counts
##   [1]   6 296  56  49  23  26  12  13   7   4   4   7   2   7   7   3   1
5
##  [19]   1   0   1   4   3   0   1   2   0   0   3   2   1   3   1   2   1
2
##  [37]   3   0   0   3   1   0   0   0   1   4   2   0   0   1   0   0   1
0
##  [55]   1   1   0   0   0   0   2   1   0   0   0   1   0   0   0   0   0
0
##  [73]   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0
0
##  [91]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [109]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [127]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [145]   0   0   0   0   0   1
##
## $density
##   [1] 0.020725389 1.022452504 0.193436960 0.169257340 0.079447323
0.089810017
##   [7] 0.041450777 0.044905009 0.024179620 0.013816926 0.013816926
0.024179620
##  [13] 0.006908463 0.024179620 0.024179620 0.010362694 0.003454231
0.017271157
##  [19] 0.003454231 0.000000000 0.003454231 0.013816926 0.010362694
0.000000000
##  [25] 0.003454231 0.006908463 0.000000000 0.000000000 0.010362694
0.006908463
##  [31] 0.003454231 0.010362694 0.003454231 0.006908463 0.003454231
0.006908463
##  [37] 0.010362694 0.000000000 0.000000000 0.010362694 0.003454231
0.000000000
##  [43] 0.000000000 0.000000000 0.003454231 0.013816926 0.006908463
```

```
0.000000000
##  [49] 0.000000000 0.003454231 0.000000000 0.000000000 0.003454231
0.000000000
##  [55] 0.003454231 0.003454231 0.000000000 0.000000000 0.000000000
0.000000000
##  [61] 0.006908463 0.003454231 0.000000000 0.000000000 0.000000000
0.003454231
##  [67] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
##  [73] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
##  [79] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
##  [85] 0.000000000 0.003454231 0.000000000 0.000000000 0.000000000
0.000000000
##  [91] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
##  [97] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [103] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [109] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [115] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [121] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [127] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [133] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [139] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [145] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.003454231
##
## $mids
##   [1]  0.25  0.75  1.25  1.75  2.25  2.75  3.25  3.75  4.25  4.75  5.25
5.75
##  [13]  6.25  6.75  7.25  7.75  8.25  8.75  9.25  9.75 10.25 10.75 11.25
11.75
##  [25] 12.25 12.75 13.25 13.75 14.25 14.75 15.25 15.75 16.25 16.75 17.25
17.75
##  [37] 18.25 18.75 19.25 19.75 20.25 20.75 21.25 21.75 22.25 22.75 23.25
23.75
##  [49] 24.25 24.75 25.25 25.75 26.25 26.75 27.25 27.75 28.25 28.75 29.25
29.75
##  [61] 30.25 30.75 31.25 31.75 32.25 32.75 33.25 33.75 34.25 34.75 35.25
35.75
##  [73] 36.25 36.75 37.25 37.75 38.25 38.75 39.25 39.75 40.25 40.75 41.25
```

```
41.75
##  [85] 42.25 42.75 43.25 43.75 44.25 44.75 45.25 45.75 46.25 46.75 47.25
47.75
##  [97] 48.25 48.75 49.25 49.75 50.25 50.75 51.25 51.75 52.25 52.75 53.25
53.75
## [109] 54.25 54.75 55.25 55.75 56.25 56.75 57.25 57.75 58.25 58.75 59.25
59.75
## [121] 60.25 60.75 61.25 61.75 62.25 62.75 63.25 63.75 64.25 64.75 65.25
65.75
## [133] 66.25 66.75 67.25 67.75 68.25 68.75 69.25 69.75 70.25 70.75 71.25
71.75
## [145] 72.25 72.75 73.25 73.75 74.25 74.75
##
## $xname
## [1] "liver$Total_Bilirubin"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```r
lines(seq(min(liver$Total_Bilirubin), max(liver$Total_Bilirubin),
      length = length(liver$Total_Bilirubin)),
      mix.gam.3[["prob"]][1]*dGA(seq(min(liver$Total_Bilirubin),
      max(liver$Total_Bilirubin), length = length(liver$Total_Bilirubin)),
      mu = mu.hat1, sigma = sigma.hat1), lty=1, lwd=3, col="blue")

lines(seq(min(liver$Total_Bilirubin), max(liver$Total_Bilirubin),
      length = length(liver$Total_Bilirubin)),
      mix.gam.3[["prob"]][2]*dGA(seq(min(liver$Total_Bilirubin),
      max(liver$Total_Bilirubin), length = length(liver$Total_Bilirubin)),
      mu = mu.hat2, sigma = sigma.hat2), lty = 2, lwd = 3, col = "red")

lines(seq(min(liver$Total_Bilirubin), max(liver$Total_Bilirubin),
      length = length(liver$Total_Bilirubin)),
      mix.gam.3[["prob"]][1]*dGA(seq(min(liver$Total_Bilirubin),
      max(liver$Total_Bilirubin), length = length(liver$Total_Bilirubin)),
      mu = mu.hat1, sigma = sigma.hat1)+
      mix.gam.3[["prob"]][2]*dRG(seq(min(liver$Total_Bilirubin),
      max(liver$Total_Bilirubin), length = length(liver$Total_Bilirubin)),
      mu = mu.hat2, sigma = sigma.hat2), lty = 3, lwd = 3, col = "green")
```




```
mix.gm.tb <- data.frame(row.names=c('Gamma Mixture, K=3','Gamma Mixture,
K=2', "Generalized Gamma"),
                        AIC=c(mix.gam.3$aic, mix.gam$aic, AIC(tb.GG)),
                        BIC=c(mix.gam.3$sbc, mix.gam$sbc, tb.GG$sbc))
mix.gm.tb
```

```
##                           AIC      BIC
## Gamma Mixture, K=3 1740.297 1775.188
## Gamma Mixture, K=2 1867.515 1889.322
## Generalized Gamma  1833.703 1846.787
```

We can observe that the AIC value of the mixture of Gamma with k=3 has increased, since k=2 and single Gamma distribution values are higher than that of the current AIC and BIC values. The current AIC value is `1740.297`, whereas the previous value which is higher is `1833.703` and the current BIC value is `1775.188`, whereas the current value which is higher is `1846.787`. Here we can clearly see that our data fits better in the Gamma Mixture Distribution with k=3.

Direct Bilirubin

Lets explore the `Direct Bilirubin` variable:

```
head(liver$Direct_Bilirubin)
```

```
## [1] 0.1 5.5 4.1 0.4 2.0 0.7
```

```
length(liver$Direct_Bilirubin)
```

```
## [1] 579
```

```
table(liver$Direct_Bilirubin)
```

```
##
##  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9    1  1.1  1.2  1.3  1.4  1.5
1.6
##   63  192   50   21   20   15   11   22    7   13    7   10   12    7    6
11
##  1.7  1.8  1.9    2  2.1  2.2  2.3  2.4  2.5  2.6  2.7  2.8  2.9    3  3.2
3.3
##    2    3    2    3    4    2    5    1    3    1    3    1    1    5    6
1
##  3.6  3.7  3.9    4  4.1  4.2  4.3  4.5  4.6  4.9    5  5.1  5.2  5.5  5.6
6
##    4    2    1    3    2    1    2    2    1    1    2    1    1    1    1
1
##  6.1  6.2  6.4    7  7.2  7.6  7.7  7.8  8.2  8.4  8.5  8.8  8.9    9  9.5
10
##    1    1    1    2    1    2    1    1    2    2    2    2    1    2    3
1
## 10.2 10.4 10.8 11.3 11.4 11.7 11.8 12.1 12.6 12.8 13.7 14.1 14.2 17.1 18.3
19.7
##    1    1    1    1    1    1    2    1    2    1    1    1    1    1    1
1
```

The table() function in R returns the absolute frequencies for each patient-specified value in the data set.

```
unique(liver$Direct_Bilirubin)
```

```
##  [1]  0.1  5.5  4.1  0.4  2.0  0.7  0.2  0.3  1.3  0.8  0.5  1.0  3.0  1.9
1.2
## [16]  7.8  0.6  1.1  3.2  1.8  8.8  1.6  4.5  2.8  4.0  2.7  2.4  1.5  2.3
3.6
## [31]  6.2  7.0  8.2 11.3 10.2  2.5  1.4  1.7  5.6  2.2  2.1  4.9  5.0  0.9
12.6
## [46]  7.6  9.0  4.6 11.8 14.2  8.9  6.4  9.5  3.3 11.4  4.3  3.7  2.6  3.9
5.1
## [61] 12.8 10.4 17.1 14.1  8.5 10.0 12.1  2.9  5.2 18.3  7.2 11.7 10.8  6.1
4.2
## [76] 19.7  7.7  8.4  6.0 13.7
```

```
length(unique(liver$Direct_Bilirubin))
```

```
## [1] 80
```

```
min(liver$Direct_Bilirubin)
```

```
## [1] 0.1
```

```
max(liver$Direct_Bilirubin)
```

```
## [1] 19.7
```

Here the total observation of Age is 579 and is a continuous variable that range lies between 0.1-19.7. Now, we will see the summary of the Age variable and also age is a continues variable, for further analysis as follows:

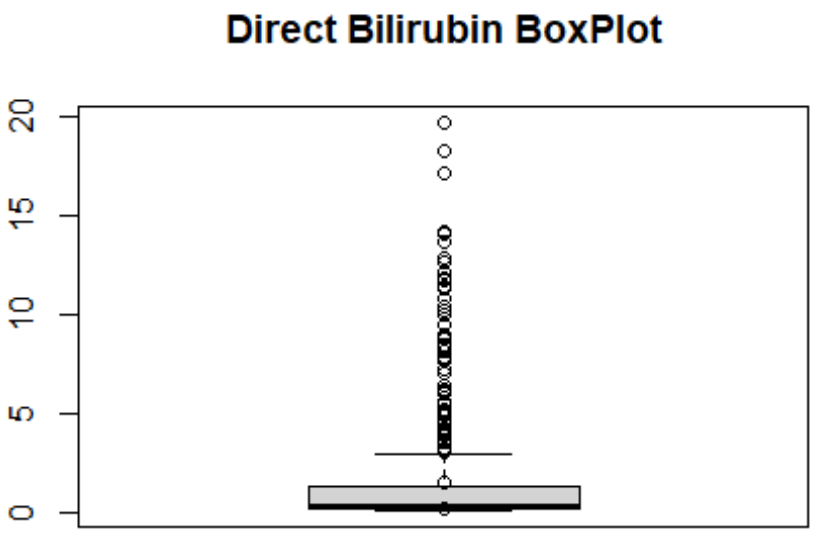
```
summary(liver$Direct_Bilirubin)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.100   0.200   0.300   1.494   1.300  19.700

sd(liver$Direct_Bilirubin)

## [1] 2.816499

var(liver$Direct_Bilirubin)

## [1] 7.932664

getMode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
v <- c(liver$Direct_Bilirubin)
mode <- getMode(v)
print(mode)

## [1] 0.2
```

From the above summary we can observe that the Mean (1.494), Median (0.300) and Mode (0.2) are not equal so the distribution is asymmetrical. Here Mean > Median > Mode so the distribution could be positive and skewed to the right. Now we also confirm this as follows:

```
library(moments)
skewness(liver$Direct_Bilirubin)

## [1] 3.190869
```

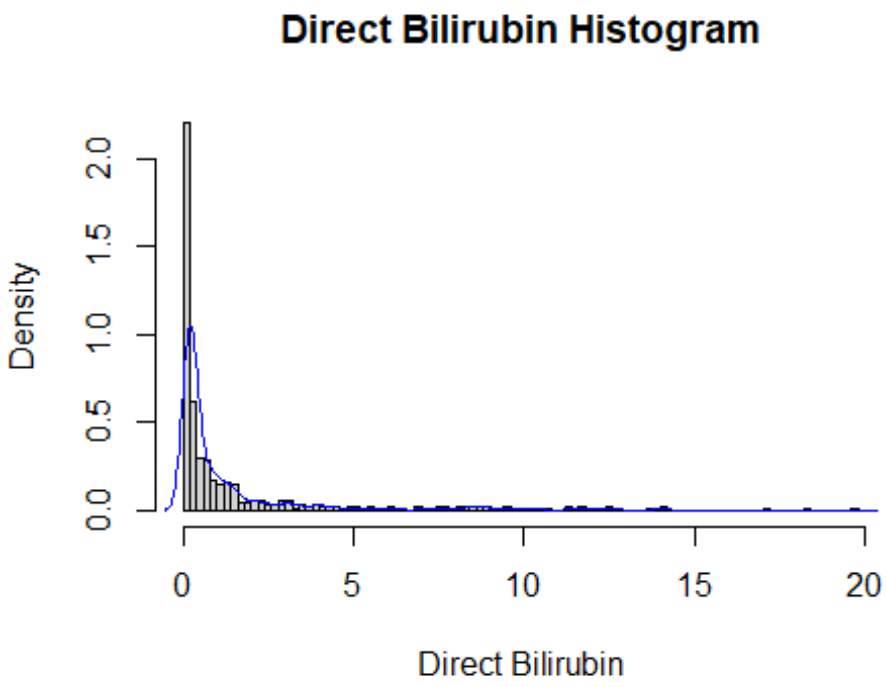
Hence, the skewness is the positive so the distribution is positively skewed was rightly observed. Also I will plot the Box plot to comparing the distribution of data across data set as follows:

```
boxplot(liver$Direct_Bilirubin, main = "Direct Bilirubin BoxPlot")
points(mean(liver$Direct_Bilirubin))
points(mode)
```



Box plots are graphical displays of the five number summary, So they describe the Minimum (lowest mean relative) lower whisker, Quartile 1 (It separates the lowest 25% observation to the rest of the observations), Median (Which divides the lower 50% observation from the upper 50% observations), Quartile 3 (It divides the upper 25% observation to the rest of the observations) and Maximum. In the above Boxplot diagram, there are many outliers and the upper whisker shows the highest mean relative. Now, I will plot a histogram a graphical display of data using bars of different heights to measures distribution of continuous data as follows:

```
hist(liver$Direct_Bilirubin, prob = TRUE, breaks = 80, xlab = "Direct
Bilirubin", main = "Direct Bilirubin Histogram")
lines(density(liver$Direct_Bilirubin), col='blue')
```

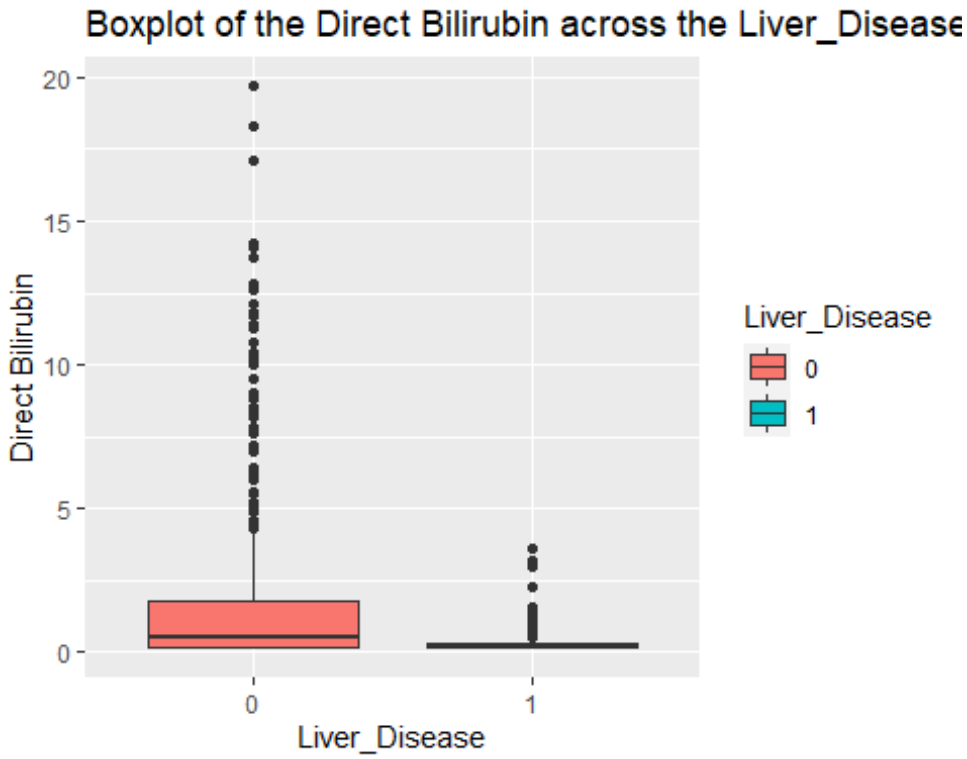


From the above histogram, we can observe that the diagram has many peaks.Using the density lines, I have approximated the histogram graph. Density provides information about the type of distribution under consideration and allows us to determine whether the attribute is distributed normally or not. From the graph above we can see how the `Direct Bilirubin` variable does not seem to fit to a Normal distribution, there is a peak of the frequency distributions around many values and also a left skewed. We have already observed a positive value of `3.190869`, so we will affirm that the distribution is right skewed. To check whether the distribution is `Platykurtic` or not, we will check this by following R method as follows:

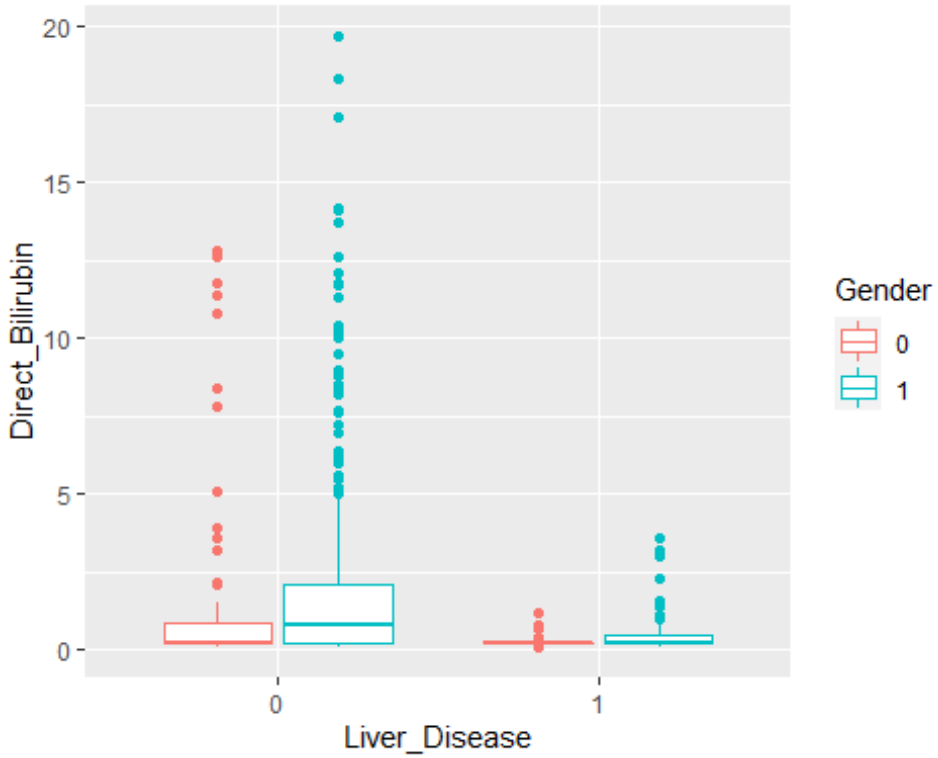
```
kurtosis(liver$Direct_Bilirubin)

## [1] 14.14493
```

The distribution is also `Leptokurtic`, since the value is greater than 3.

```
ggplot(liver, aes(x = Liver_Disease, y = Direct_Bilirubin, fill =
Liver_Disease)) +
  geom_boxplot() +
  ylab("Direct Bilirubin") +
  ggtitle("Boxplot of the Direct Bilirubin across the Liver_Disease")
```




```
ggplot(liver, aes(Liver_Disease, Direct_Bilirubin)) +
  geom_boxplot(aes(color = Gender))
```

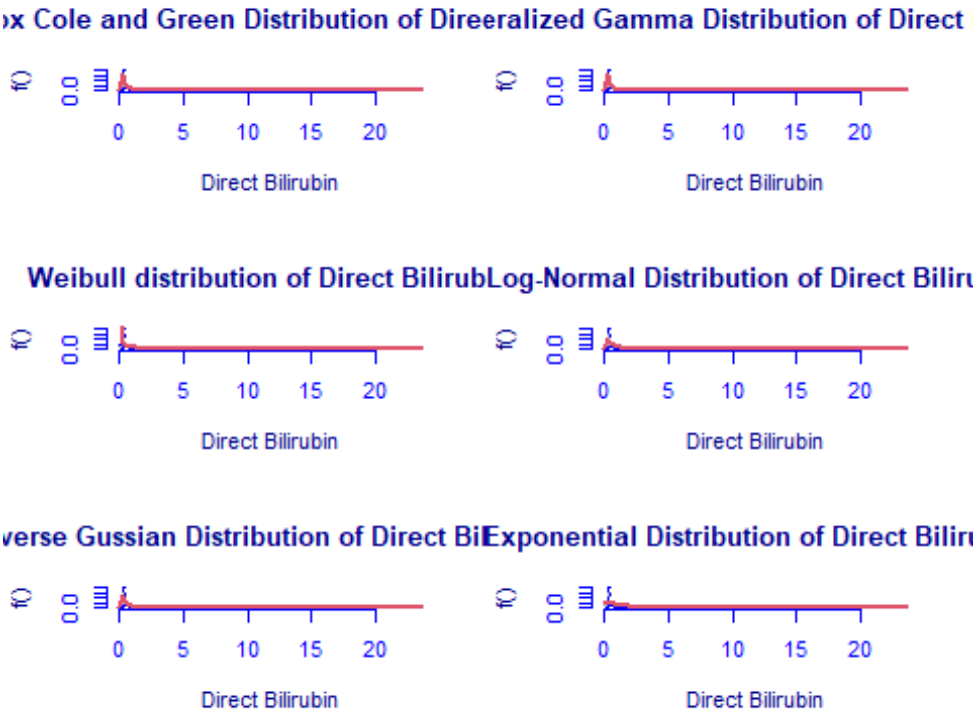


Similar to the Total Bilirubin we have skewed data for the Direct Bilirubin. The mean is 1.486, with a maximum being 19.7 and the median being 0.3. There is a difference in the mean and range of the Direct Bilirubin for the Liver_Disease. There are differences also when you look t it further by gender. The mean Direct Bilirubin is higher for the male group across the Liver_Disease. There is a bigger range for both genders for Direct Bilirubin when the response is 0.

Direct Bilirubin Fit for the Data

Now we will try to fit different models to Age distribution evaluating the Akaike Information Criterion (AIC) and the Bayesian Information criterion (BIC), The lower are the indexes, the better is the fitting. I will evaluate both the single parameter distributions and mixture distributions as follows:

```
library(MASS)
library(gamlss)


par(mfrow=c(3,2))

db.BCCG <- histDist(liver$Direct_Bilirubin, family=BCCG, nbins = 80, xlab =
"Direct Bilirubin", main="Box-Cox Cole and Green Distribution of Direct
Bilirubin")

db.GG <- histDist(liver$Direct_Bilirubin, family=GG, nbins = 80, xlab =
"Direct Bilirubin", main="Generalized Gamma Distribution of Direct
Bilirubin")

db.WEI <- histDist(liver$Direct_Bilirubin, family=WEI, nbins = 80, xlab =
"Direct Bilirubin", main="Weibull distribution of Direct Bilirubin")

db.LOGNO <- histDist(liver$Direct_Bilirubin, family=LOGNO, nbins = 80, xlab =
"Direct Bilirubin", main="Log-Normal Distribution of Direct Bilirubin")

db.IG <- histDist(liver$Direct_Bilirubin, family=IG, nbins = 80, xlab =
"Direct Bilirubin", main = "Inverse Gussian Distribution of Direct
Bilirubin")

db.EXP<- histDist(liver$Direct_Bilirubin, family=EXP, nbins = 80, xlab =
"Direct Bilirubin", main = "Exponential Distribution of Direct Bilirubin")
```




```
library(Matrix)
library(glmnet)
df <- data.frame(Rownames = c("Box-Cox Cole and Green","Generalized Gamma",
"Weibull",
                              "Log-Normal", "Inverse Gussian", "Exponential"),
                 AIC = c(AIC(db.BCCG), AIC(db.GG), AIC(db.WEI),
AIC(db.LOGNO),
                         AIC(db.IG), AIC(db.EXP)),
                 BIC = c(db.BCCG$sbc, db.GG$sbc, db.WEI$sbc,
db.LOGNO$sbc,
                         db.IG$sbc, db.EXP$sbc),
                 df = c(db.BCCG$df.fit, db.GG$df.fit, db.WEI$df.fit,
                        db.LOGNO$df.fit, db.IG$df.fit, db.EXP$df.fit),
                 LogLike = c(logLik(db.BCCG), logLik(db.GG),
logLik(db.WEI),
                             logLik(db.LOGNO), logLik(db.IG),
logLik(db.EXP)))
df
```

```
##                 Rownames      AIC      BIC df   LogLike
## 1 Box-Cox Cole and Green 1078.120 1091.204  3 -536.0602
## 2      Generalized Gamma 1062.880 1075.964  3 -528.4398
## 3                Weibull 1417.420 1426.142  2 -706.7099
## 4             Log-Normal 1227.517 1236.240  2 -611.7586
## 5        Inverse Gussian 1119.812 1128.535  2 -557.9061
## 6            Exponential 1624.986 1629.348  1 -811.4932
```

As we can see, the model with the highest log likelihood (-528.4398) and the lowest AIC (1062.880) and BIC (1075.964) is the Generalized Gamma Distribution. Therefore, based on the greatest likelihood technique, our data fits better in the Generalized Gamma Distribution. Now, I will also compare two models by means of the Likelihood ratio test as we performed above as follows:

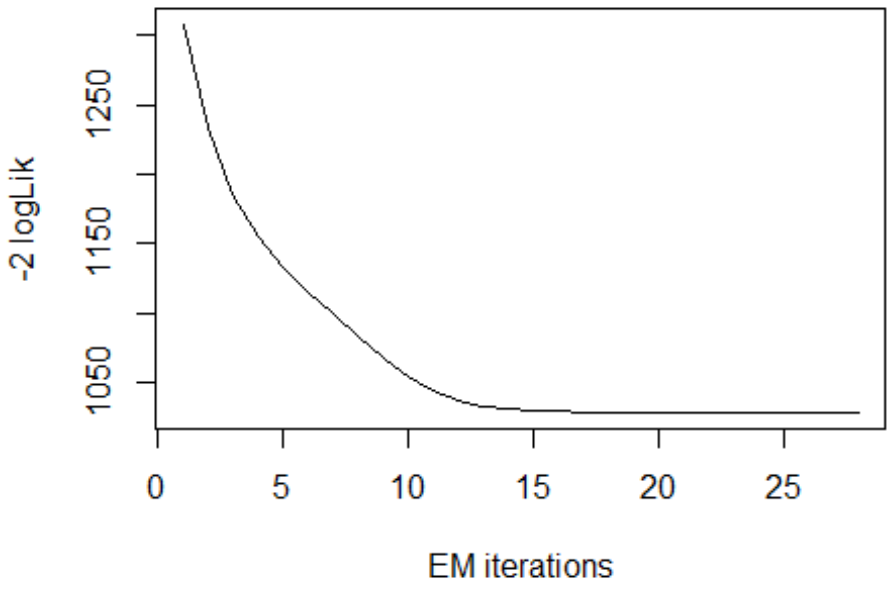
```
library(zoo)
library(lmtest)
lrtest(db.GG, db.EXP)

## Likelihood ratio test
##
## Model 1: gamlssML(formula = liver$Direct_Bilirubin, family = "GG")
## Model 2: gamlssML(formula = liver$Direct_Bilirubin, family = "EXP")
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   3 -528.44
## 2   1 -811.49 -2 566.11  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Under the null hypothesis, we compare the Generalized Gamma distribution to the Exponential distribution under the alternative hypothesis. At any level of significance, we can reject the null hypothesis. Because our distributions fit better in the Generalized Gamma model, we'll use it.

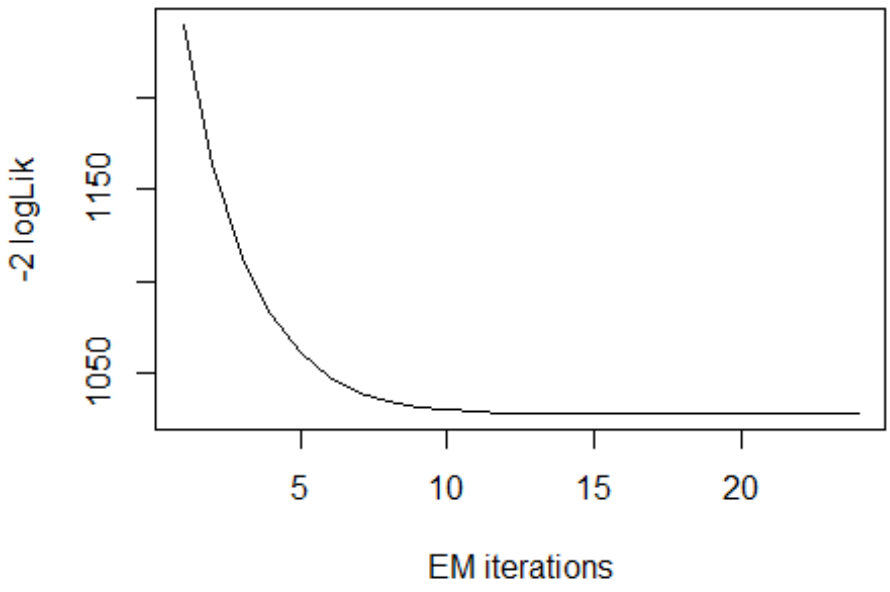
Distributions Mixture

Now, I will also apply the fitting criteria for the distributions with Gamma mixture as follows:

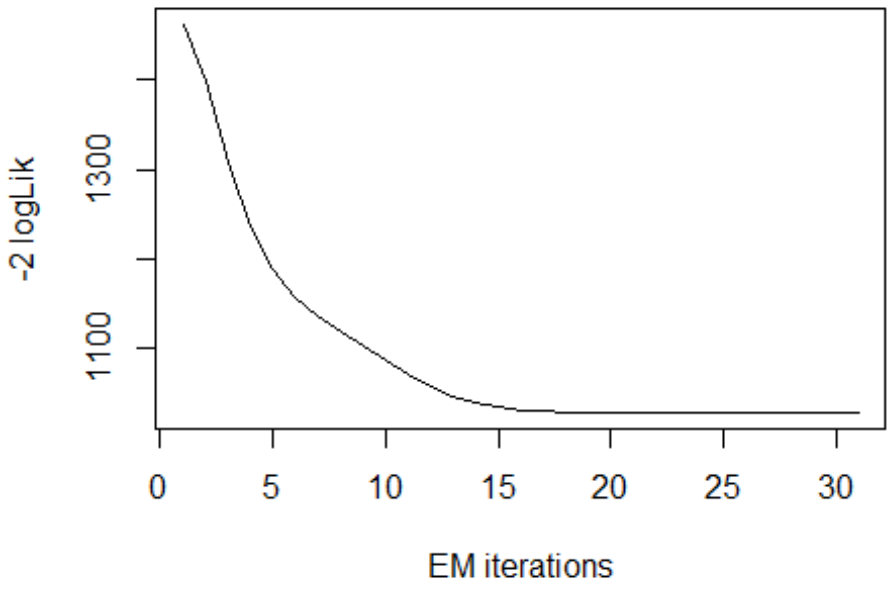
```
library(gamlss.mx)
mix.gam <- gamlssMXfits(n = 5, liver$Direct_Bilirubin~1, family = GA, K = 2,
data = liver)
```



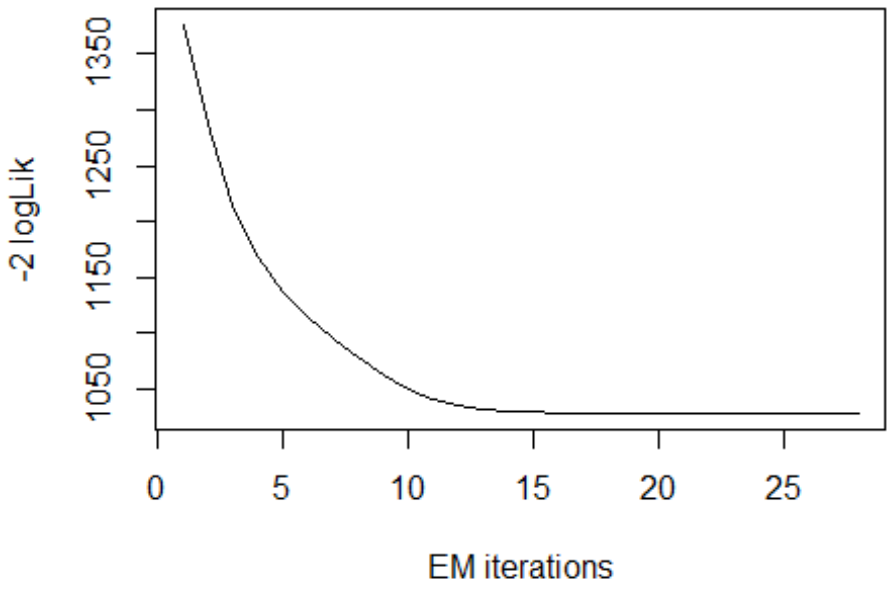

```
## model= 1
```



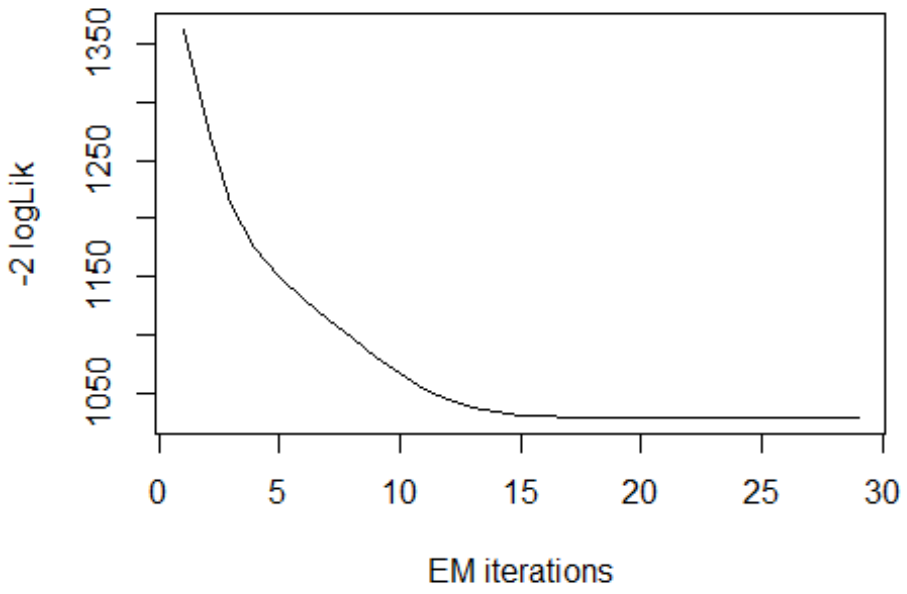

```
## model= 2
```




```
## model= 3
```




```
## model= 4
```




```
## model= 5

print(mix.gam)

##
## Mixing Family:  c("GA", "GA")
##
## Fitting method: EM algorithm
##
## Call:  gamlssMX(formula = liver$Direct_Bilirubin ~ 1, family = GA,
##     K = 2, data = liver)
##
## Mu Coefficients for model: 1
## (Intercept)
##      -1.587
## Sigma Coefficients for model: 1
## (Intercept)
##      -1.058
## Mu Coefficients for model: 2
## (Intercept)
##      0.9989
## Sigma Coefficients for model: 2
## (Intercept)
##     0.08336
##
## Estimated probabilities: 0.4863282 0.5136718
##
```

```
## Degrees of Freedom for the fit: 5 Residual Deg. of Freedom   574
## Global Deviance:     1028.49
##             AIC:     1038.49
##             SBC:     1060.3
```

We can observe that the AIC value of the mixture of Gamma has improved, since it is lower than that of the single Gamma distribution. The current `AIC` value is `1038.49`, whereas the previous value was `1062.880` and the current `BIC` value is `1060.3`, whereas the previous value was `1075.964`.

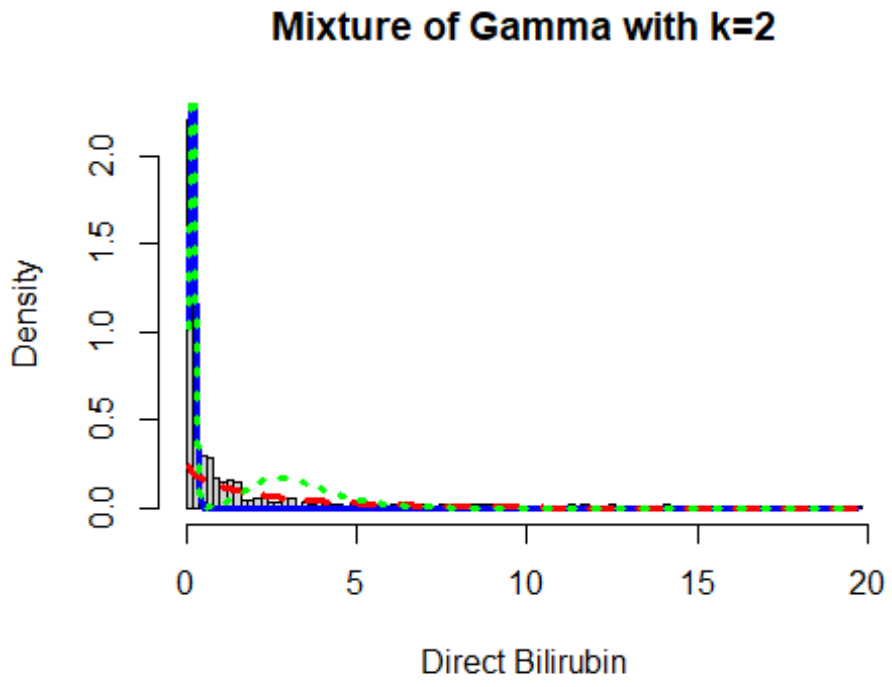
```
logLik(mix.gam)

## 'log Lik.' -514.2449 (df=5)

mix.gam$prob

## [1] 0.4863282 0.5136718

fitted(mix.gam, "mu")[1]

## [1] 1.494229

fitted(mix.gam, "sigma")[2]

## [1] 1.494229

hist(liver$Direct_Bilirubin, breaks = 80, xlab = "Direct Bilirubin",
main="Mixture of Gamma with k=2", freq = FALSE)

mu.hat1 <- exp(mix.gam[["models"]][[1]][["mu.coefficients"]])

sigma.hat1 <- exp(mix.gam[["models"]][[1]][["sigma.coefficients"]])

mu.hat2 <- exp(mix.gam[["models"]][[2]][["mu.coefficients"]])

sigma.hat2 <- exp(mix.gam[["models"]][[2]][["sigma.coefficients"]])

hist(liver$Direct_Bilirubin, breaks = 80, freq = FALSE, plot = FALSE)

## Warning in hist.default(liver$Direct_Bilirubin, breaks = 80, freq = FALSE,
:
## argument 'freq' is not made use of

## $breaks
##   [1]  0.0  0.2  0.4  0.6  0.8  1.0  1.2  1.4  1.6  1.8  2.0  2.2  2.4
2.6  2.8
##  [16]  3.0  3.2  3.4  3.6  3.8  4.0  4.2  4.4  4.6  4.8  5.0  5.2  5.4
5.6  5.8
##  [31]  6.0  6.2  6.4  6.6  6.8  7.0  7.2  7.4  7.6  7.8  8.0  8.2  8.4
8.6  8.8
##  [46]  9.0  9.2  9.4  9.6  9.8 10.0 10.2 10.4 10.6 10.8 11.0 11.2 11.4
11.6 11.8
```

```
##  [61] 12.0 12.2 12.4 12.6 12.8 13.0 13.2 13.4 13.6 13.8 14.0 14.2 14.4
14.6 14.8
##  [76] 15.0 15.2 15.4 15.6 15.8 16.0 16.2 16.4 16.6 16.8 17.0 17.2 17.4
17.6 17.8
##  [91] 18.0 18.2 18.4 18.6 18.8 19.0 19.2 19.4 19.6 19.8
##
## $counts
##  [1] 255  71  35  33  20  17  19  17   5   5   6   6   4   4   6   6   1
4   2
## [20]   4   3   2   3   0   3   2   0   2   0   1   2   1   0   0   2   1
0   2
## [39]   2   0   2   2   2   2   3   0   0   3   0   1   1   1   0   1   0
0   2
## [58]   0   3   0   1   0   2   1   0   0   0   0   1   0   2   0   0   0
0   0
## [77]   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   1   0
0   0
## [96]   0   0   0   1
##
## $density
##  [1] 2.202072539 0.613126079 0.302245250 0.284974093 0.172711572
0.146804836
##  [7] 0.164075993 0.146804836 0.043177893 0.043177893 0.051813472
0.051813472
## [13] 0.034542314 0.034542314 0.051813472 0.051813472 0.008635579
0.034542314
## [19] 0.017271157 0.034542314 0.025906736 0.017271157 0.025906736
0.000000000
## [25] 0.025906736 0.017271157 0.000000000 0.017271157 0.000000000
0.008635579
## [31] 0.017271157 0.008635579 0.000000000 0.000000000 0.017271157
0.008635579
## [37] 0.000000000 0.017271157 0.017271157 0.000000000 0.017271157
0.017271157
## [43] 0.017271157 0.017271157 0.025906736 0.000000000 0.000000000
0.025906736
## [49] 0.000000000 0.008635579 0.008635579 0.008635579 0.000000000
0.008635579
## [55] 0.000000000 0.000000000 0.017271157 0.000000000 0.025906736
0.000000000
## [61] 0.008635579 0.000000000 0.017271157 0.008635579 0.000000000
0.000000000
## [67] 0.000000000 0.000000000 0.008635579 0.000000000 0.017271157
0.000000000
## [73] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [79] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [85] 0.000000000 0.008635579 0.000000000 0.000000000 0.000000000
0.000000000
```

```
## [91] 0.000000000 0.008635579 0.000000000 0.000000000 0.000000000
0.000000000
## [97] 0.000000000 0.000000000 0.008635579
##
## $mids
##  [1]  0.1  0.3  0.5  0.7  0.9  1.1  1.3  1.5  1.7  1.9  2.1  2.3  2.5  2.7
2.9
## [16]  3.1  3.3  3.5  3.7  3.9  4.1  4.3  4.5  4.7  4.9  5.1  5.3  5.5  5.7
5.9
## [31]  6.1  6.3  6.5  6.7  6.9  7.1  7.3  7.5  7.7  7.9  8.1  8.3  8.5  8.7
8.9
## [46]  9.1  9.3  9.5  9.7  9.9 10.1 10.3 10.5 10.7 10.9 11.1 11.3 11.5 11.7
11.9
## [61] 12.1 12.3 12.5 12.7 12.9 13.1 13.3 13.5 13.7 13.9 14.1 14.3 14.5 14.7
14.9
## [76] 15.1 15.3 15.5 15.7 15.9 16.1 16.3 16.5 16.7 16.9 17.1 17.3 17.5 17.7
17.9
## [91] 18.1 18.3 18.5 18.7 18.9 19.1 19.3 19.5 19.7
##
## $xname
## [1] "liver$Direct_Bilirubin"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

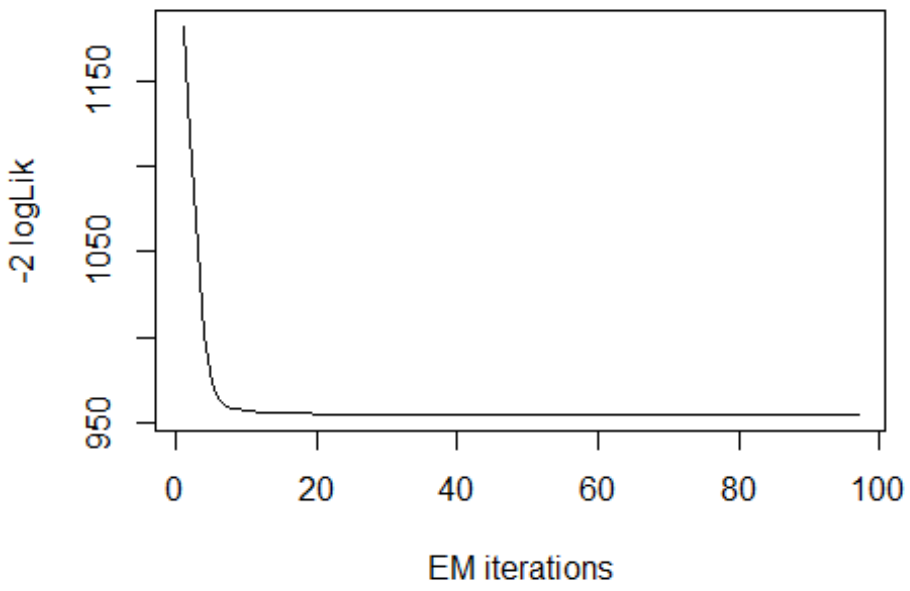
```
lines(seq(min(liver$Direct_Bilirubin), max(liver$Direct_Bilirubin), length =
length(liver$Direct_Bilirubin)),
      mix.gam[["prob"]][1]*dGA(seq(min(liver$Direct_Bilirubin),
max(liver$Direct_Bilirubin),
      length = length(liver$Direct_Bilirubin)), mu = mu.hat1, sigma =
sigma.hat1),
      lty=1, lwd=3, col="blue")

lines(seq(min(liver$Direct_Bilirubin), max(liver$Direct_Bilirubin), length =
length(liver$Direct_Bilirubin)),
      mix.gam[["prob"]][2]*dGA(seq(min(liver$Direct_Bilirubin),
max(liver$Direct_Bilirubin),
      length = length(liver$Direct_Bilirubin)), mu = mu.hat2, sigma =
sigma.hat2),
      lty = 2, lwd = 3, col = "red")

lines(seq(min(liver$Direct_Bilirubin), max(liver$Direct_Bilirubin), length =
length(liver$Direct_Bilirubin)),
      mix.gam[["prob"]][1]*dGA(seq(min(liver$Direct_Bilirubin),
max(liver$Direct_Bilirubin),
      length = length(liver$Direct_Bilirubin)), mu = mu.hat1, sigma =
sigma.hat1)+
```

```
      mix.gam[["prob"]][2]*dRG(seq(min(liver$Direct_Bilirubin),
max(liver$Direct_Bilirubin),
      length = length(liver$Direct_Bilirubin)), mu = mu.hat2, sigma =
sigma.hat2),
      lty = 3, lwd = 3, col = "green")
```

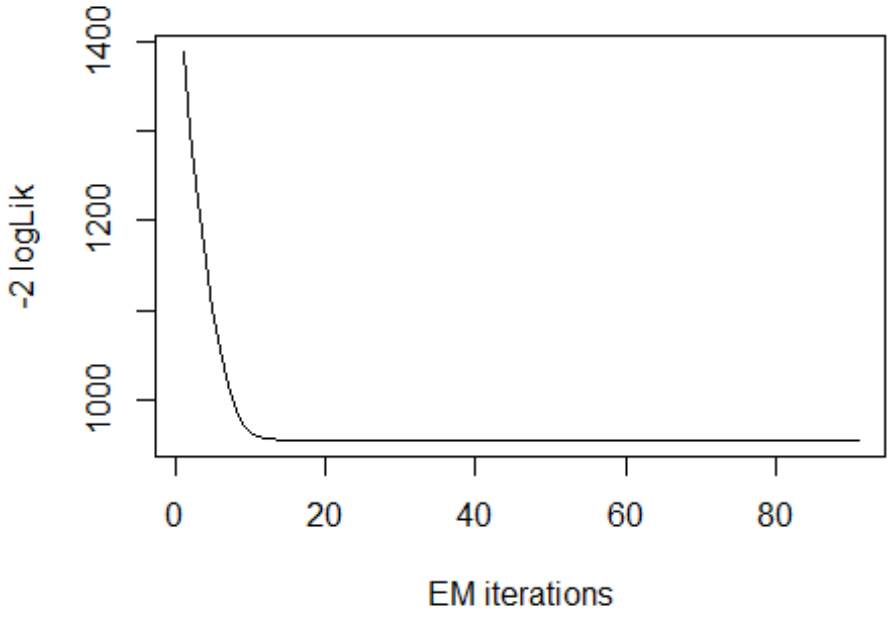


Now, I will again apply the fitting criteria for the distributions with Gamma mixture with the k = 3 as follows:

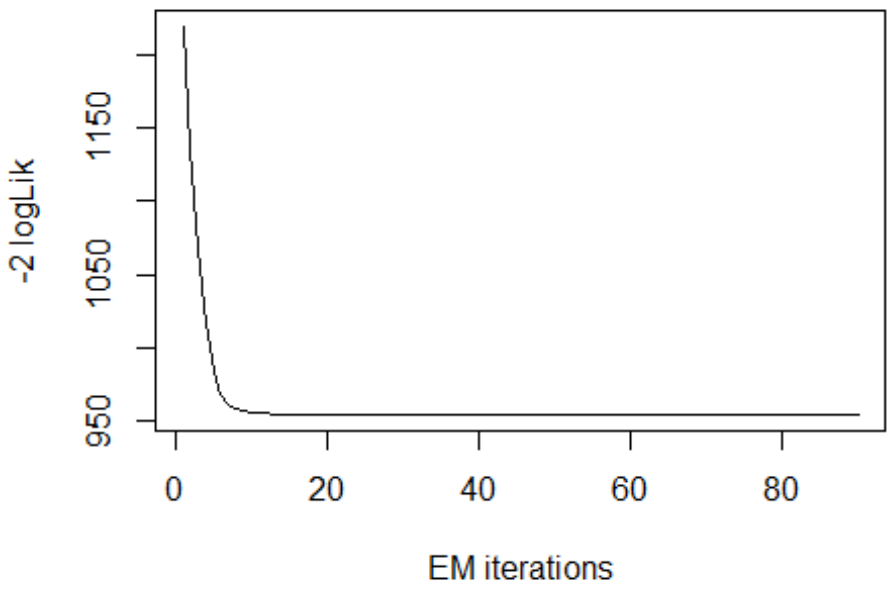
```
mix.gam.3 <- gamlssMXfits(n = 5, liver$Direct_Bilirubin~1, family = GA, K =
3, data = liver)
```



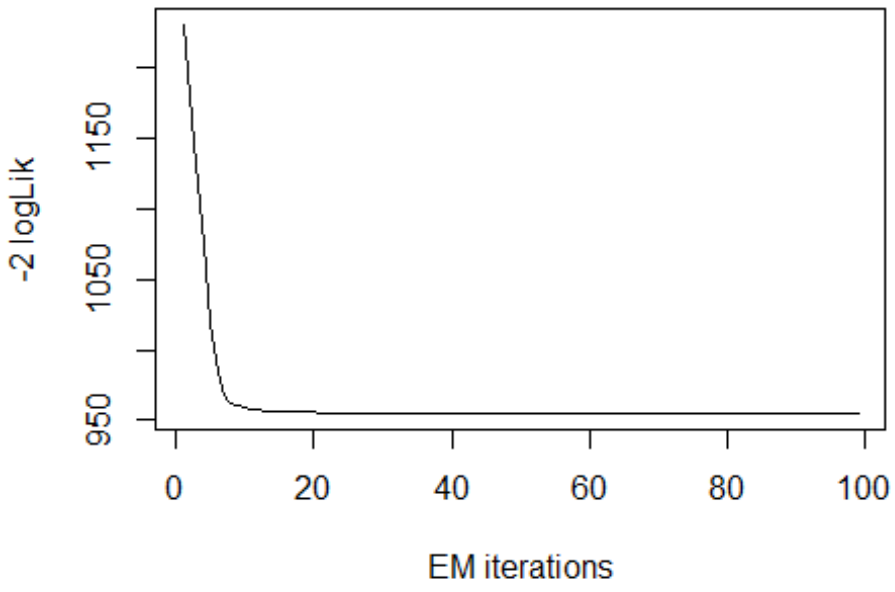

```
## model= 1
```



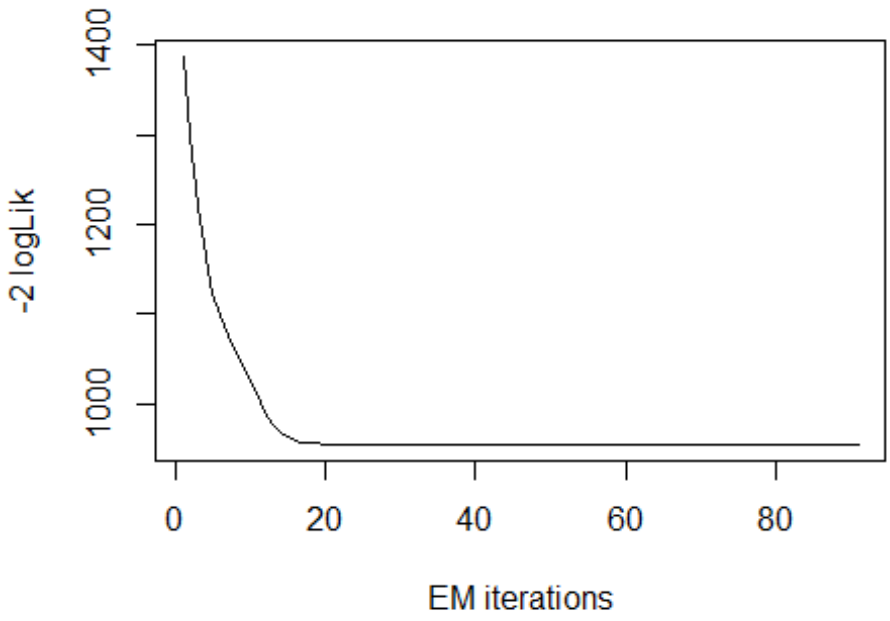

```
## model= 2
```




```
## model= 3
```




```
## model= 4
```




```
## model= 5

print(mix.gam.3)

##
## Mixing Family:  c("GA", "GA", "GA")
##
## Fitting method: EM algorithm
##
## Call:  gamlssMX(formula = liver$Direct_Bilirubin ~ 1, family = GA,
##     K = 3, data = liver)
##
## Mu Coefficients for model: 1
## (Intercept)
##    -0.06072
## Sigma Coefficients for model: 1
## (Intercept)
##     -0.4776
## Mu Coefficients for model: 2
## (Intercept)
##       1.748
## Sigma Coefficients for model: 2
## (Intercept)
##      -0.275
## Mu Coefficients for model: 3
## (Intercept)
##      -1.619
```

```
## Sigma Coefficients for model: 3
## (Intercept)
##      -1.065
##
## Estimated probabilities: 0.3069872 0.1924769 0.5005359
##
## Degrees of Freedom for the fit: 8 Residual Deg. of Freedom   571
## Global Deviance:     954.941
##             AIC:     970.941
##             SBC:     1005.83
```

```
logLik(mix.gam.3)
```

```
## 'log Lik.' -477.4706 (df=8)
```

```
mix.gam.3$prob
```

```
## [1] 0.3069872 0.1924769 0.5005359
```

```
fitted(mix.gam.3, "mu")[1]
```

```
## [1] 1.493881
```

```
fitted(mix.gam.3, "sigma")[2]
```

```
## [1] 1.493881
```

```
hist(liver$Direct_Bilirubin, breaks = 80, xlab = "Direct Bilirubin",
main="Mixture of Gamma with k=3", freq = FALSE)

mu.hat1 <- exp(mix.gam.3[["models"]][[1]][["mu.coefficients"]])

sigma.hat1 <- exp(mix.gam.3[["models"]][[1]][["sigma.coefficients"]])

mu.hat2 <- exp(mix.gam.3[["models"]][[2]][["mu.coefficients"]])

sigma.hat2 <- exp(mix.gam.3[["models"]][[2]][["sigma.coefficients"]])

hist(liver$Direct_Bilirubin, breaks = 80, freq = FALSE, plot = FALSE)
```

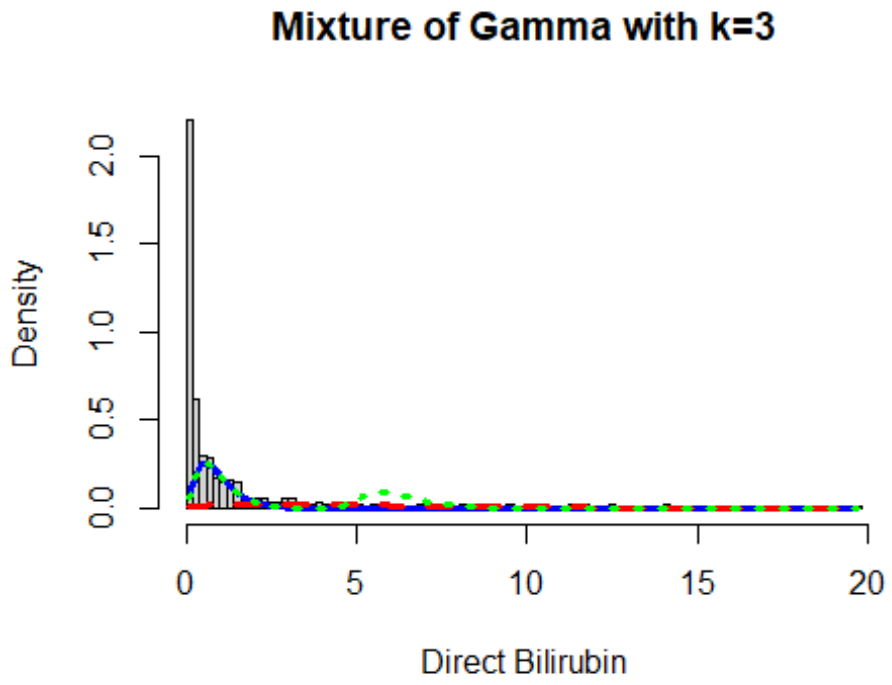
```
## Warning in hist.default(liver$Direct_Bilirubin, breaks = 80, freq = FALSE,
:
## argument 'freq' is not made use of
```

```
## $breaks
##   [1]  0.0  0.2  0.4  0.6  0.8  1.0  1.2  1.4  1.6  1.8  2.0  2.2  2.4
2.6  2.8
##  [16]  3.0  3.2  3.4  3.6  3.8  4.0  4.2  4.4  4.6  4.8  5.0  5.2  5.4
5.6  5.8
##  [31]  6.0  6.2  6.4  6.6  6.8  7.0  7.2  7.4  7.6  7.8  8.0  8.2  8.4
8.6  8.8
##  [46]  9.0  9.2  9.4  9.6  9.8 10.0 10.2 10.4 10.6 10.8 11.0 11.2 11.4
```

```
11.6 11.8
##  [61] 12.0 12.2 12.4 12.6 12.8 13.0 13.2 13.4 13.6 13.8 14.0 14.2 14.4
14.6 14.8
##  [76] 15.0 15.2 15.4 15.6 15.8 16.0 16.2 16.4 16.6 16.8 17.0 17.2 17.4
17.6 17.8
##  [91] 18.0 18.2 18.4 18.6 18.8 19.0 19.2 19.4 19.6 19.8
##
## $counts
##  [1] 255  71  35  33  20  17  19  17   5   5   6   6   4   4   6   6   1
4   2
## [20]   4   3   2   3   0   3   2   0   2   0   1   2   1   0   0   2   1
0   2
## [39]   2   0   2   2   2   2   3   0   0   3   0   1   1   1   0   1   0
0   2
## [58]   0   3   0   1   0   2   1   0   0   0   0   1   0   2   0   0   0
0   0
## [77]   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   1   0
0   0
## [96]   0   0   0   1
##
## $density
##  [1] 2.202072539 0.613126079 0.302245250 0.284974093 0.172711572
0.146804836
##  [7] 0.164075993 0.146804836 0.043177893 0.043177893 0.051813472
0.051813472
## [13] 0.034542314 0.034542314 0.051813472 0.051813472 0.008635579
0.034542314
## [19] 0.017271157 0.034542314 0.025906736 0.017271157 0.025906736
0.000000000
## [25] 0.025906736 0.017271157 0.000000000 0.017271157 0.000000000
0.008635579
## [31] 0.017271157 0.008635579 0.000000000 0.000000000 0.017271157
0.008635579
## [37] 0.000000000 0.017271157 0.017271157 0.000000000 0.017271157
0.017271157
## [43] 0.017271157 0.017271157 0.025906736 0.000000000 0.000000000
0.025906736
## [49] 0.000000000 0.008635579 0.008635579 0.008635579 0.000000000
0.008635579
## [55] 0.000000000 0.000000000 0.017271157 0.000000000 0.025906736
0.000000000
## [61] 0.008635579 0.000000000 0.017271157 0.008635579 0.000000000
0.000000000
## [67] 0.000000000 0.000000000 0.008635579 0.000000000 0.017271157
0.000000000
## [73] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [79] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
0.000000000
## [85] 0.000000000 0.008635579 0.000000000 0.000000000 0.000000000
```

```
0.000000000
## [91] 0.000000000 0.008635579 0.000000000 0.000000000 0.000000000
0.000000000
## [97] 0.000000000 0.000000000 0.008635579
##
## $mids
##  [1]  0.1  0.3  0.5  0.7  0.9  1.1  1.3  1.5  1.7  1.9  2.1  2.3  2.5  2.7
2.9
## [16]  3.1  3.3  3.5  3.7  3.9  4.1  4.3  4.5  4.7  4.9  5.1  5.3  5.5  5.7
5.9
## [31]  6.1  6.3  6.5  6.7  6.9  7.1  7.3  7.5  7.7  7.9  8.1  8.3  8.5  8.7
8.9
## [46]  9.1  9.3  9.5  9.7  9.9 10.1 10.3 10.5 10.7 10.9 11.1 11.3 11.5 11.7
11.9
## [61] 12.1 12.3 12.5 12.7 12.9 13.1 13.3 13.5 13.7 13.9 14.1 14.3 14.5 14.7
14.9
## [76] 15.1 15.3 15.5 15.7 15.9 16.1 16.3 16.5 16.7 16.9 17.1 17.3 17.5 17.7
17.9
## [91] 18.1 18.3 18.5 18.7 18.9 19.1 19.3 19.5 19.7
##
## $xname
## [1] "liver$Direct_Bilirubin"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"

lines(seq(min(liver$Direct_Bilirubin), max(liver$Direct_Bilirubin), length =
length(liver$Direct_Bilirubin)),
      mix.gam.3[["prob"]][1]*dGA(seq(min(liver$Direct_Bilirubin),
max(liver$Direct_Bilirubin),
      length = length(liver$Direct_Bilirubin)), mu = mu.hat1, sigma =
sigma.hat1),
      lty=1, lwd=3, col="blue")

lines(seq(min(liver$Direct_Bilirubin), max(liver$Direct_Bilirubin), length =
length(liver$Direct_Bilirubin)),
      mix.gam.3[["prob"]][2]*dGA(seq(min(liver$Direct_Bilirubin),
max(liver$Direct_Bilirubin),
      length = length(liver$Direct_Bilirubin)), mu = mu.hat2, sigma =
sigma.hat2),
      lty = 2, lwd = 3, col = "red")

lines(seq(min(liver$Direct_Bilirubin), max(liver$Direct_Bilirubin), length =
length(liver$Direct_Bilirubin)),
      mix.gam.3[["prob"]][1]*dGA(seq(min(liver$Direct_Bilirubin),
max(liver$Direct_Bilirubin),
      length = length(liver$Direct_Bilirubin)), mu = mu.hat1, sigma =
```

```
sigma.hat1)+
      mix.gam.3[["prob"]][2]*dRG(seq(min(liver$Direct_Bilirubin),
max(liver$Direct_Bilirubin),
      length = length(liver$Direct_Bilirubin)), mu = mu.hat2, sigma =
sigma.hat2),
      lty = 3, lwd = 3, col = "green")
```




```
mix.gm.tb <- data.frame(row.names=c('Gamma Mixture, K=3','Gamma Mixture,
K=2', "Generalized Gamma"),
                        AIC=c(mix.gam.3$aic, mix.gam$aic, AIC(age.GG)),
                        BIC=c(mix.gam.3$sbc, mix.gam$sbc, age.GG$sbc))
mix.gm.tb

##                            AIC      BIC
## Gamma Mixture, K=3  970.9412 1005.832
## Gamma Mixture, K=2 1038.4897 1060.296
## Generalized Gamma  4862.3309 4875.415
```

We can observe that the AIC value of the mixture of Gamma with k=3 and k=2 has improved, since values are lower than that of the single Gamma distribution. The previous AIC value of k=2 is `1038.4897`, whereas the current value which is lower is `970.9413` and the previous BIC value of k=2 was `1060.296`, whereas the current value which is lower is `1005.832`. Here we can clearly see that our data fits better in the Gamma Mixture Distribution with k=3.

Alkaline Phosphotase

Lets explore the `Alkaline Phosphotase` variable:

```
head(liver$Alkaline_Phosphotase)

## [1] 187 699 490 182 195 208

length(liver$Alkaline_Phosphotase)

## [1] 579

table(liver$Alkaline_Phosphotase)

##
##   63   75   90   92   97   98  100  102  103  105  108  110  114  115  116
120
##    1    1    1    2    1    1    2    1    1    1    1    2    1    1    1
1
##  123  125  127  128  130  134  135  137  138  140  142  143  144  145  146
147
##    1    1    1    1    2    1    2    2    1    4    3    1    2    9    3
1
##  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162
163
##    2    1    2    1    5    1    2    2    2    3    9    5    5    2    6
3
##  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178
179
##    1    8    1    1    6    1    4    4    2    3    3    6    3    1    5
2
##  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194
195
##    9    1    9    2    1    5    5    2    7    5   10    3    6    1    5
10
##  196  197  198  199  200  201  202  204  205  206  208  209  210  211  212
214
##    7    1   11    2    2    4    7    2    6    6    5    2    4    1    1
4
##  215  216  218  219  220  224  225  226  227  228  230  231  232  234  235
236
##   11    3    8    2    3    3    1    1    1    1    5    1    1    1    1
2
##  237  238  239  240  243  245  246  247  248  250  251  253  254  256  257
258
##    2    3    3    2    1    3    1    1    1    2    1    1    1    1    2
3
##  259  260  262  263  265  268  269  270  271  272  275  276  279  280  282
285
##    1    2    1    2    3    2    1    1    1    5    3    1    2    2    8
4
```

```
##  286  289  290  292  293  298  300  302  305  308  309  310  312  314  315
316
##    1    4    6    2    2   11    3    1    1    1    1    6    1    1    3
2
##  320  326  331  332  335  340  342  348  349  350  352  356  358  360  365
367
##    2    1    1    3    1    1    1    1    1    3    1    1    3    1    1
1
##  374  375  380  386  388  390  392  395  400  401  405  406  410  415  418
430
##    1    1    3    1    1    1    1    1    1    1    1    1    2    1    1
1
##  450  458  460  462  466  470  480  482  486  490  498  500  505  509  512
515
##    2    1    1    1    1    1    1    3    1    2    1    1    1    1    1
1
##  518  527  538  542  554  555  558  560  562  574  575  580  588  592  599
610
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
1
##  612  614  621  630  650  661  664  670  680  686  690  699  719  750  768
802
##    1    1    1    1    1    1    1    1    1    1    1    1    1    3    1
1
##  805  850  859  862  901  915  950  962 1020 1050 1100 1110 1124 1350 1420
1550
##    1    1    1    1    1    1    1    1    1    1    1    1    1    2    1
1
## 1580 1620 1630 1750 1896 2110
##    1    1    1    1    1    1
```

The table() function in R returns the absolute frequencies for each patient-specified value in the data set.

```
unique(liver$Alkaline_Phosphotase)
```

```
##   [1]  187  699  490  182  195  208  154  202  290  210  260  310  214
145  183
##  [16]  342  165  293  610  482  542  231  194  289  240  128  188  190
156  410
##  [31]  374  263  275  168  160  630  415  150  230  176  206  170  161
253  198
##  [46]  272  175  367  158  259  470  215  239  186  205  171  162  518
1620  146
##  [61]  670  915   75  148  258  237  269  320  298  538  238  308  204
282  265
##  [76]  312  243  224  225  486  257  179  661 1580 1630  280  300  178
177  201
##  [91]  802  248 1896  512  199 1110  380  159  332  189  392  286  180
218  462
```

```
## [106]  196  750 1050  599  292  962  950  200 1020  562  386  250  191
614  314
## [121]  209 1124  664  142  169 1420  135  163  285  350  220  219  401
100  116
## [136]  125  147  192  400  120  173  157 2110  360  316  498  480  680
152  859
## [151]  901  335  245  505  228  185  247  348  140  358  110  235  460
262  144
## [166]  123  575  155  315  174  340  234  430  588  527  574  216   63
302  211
## [181]  458  375  405  650  115  621  256  418  271  130  558  326  331
172  105
## [196]  102  149  580   92  719  554  555  509  690  862  592  450 1350
246  166
## [211] 1750  236  212  279  181 1550 1100  686  309  164  270  137   90
167  197
## [226]  226  352  103  850  276  193  805  151  349  365  305  127  254
108  268
## [241]  138  466  227  395   97  406  114  153  768  232  390  356  388
143  251
## [256]  134  612  515  560  500   98  184
```

```
length(unique(liver$Alkaline_Phosphotase))
```

```
## [1] 262
```

```
min(liver$Alkaline_Phosphotase)
```

```
## [1] 63
```

```
max(liver$Alkaline_Phosphotase)
```

```
## [1] 2110
```

Here the total observation of Age is 579 and is a continuous variable that range lies between 0.08-82. Now, we will see the summary of the Age variable and also age is a continues variable, for further analysis as follows:

```
summary(liver$Alkaline_Phosphotase)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    63.0   175.5   208.0   291.4   298.0  2110.0
```

```
sd(liver$Alkaline_Phosphotase)
```

```
## [1] 243.5619
```

```
var(liver$Alkaline_Phosphotase)
```

```
## [1] 59322.38
```

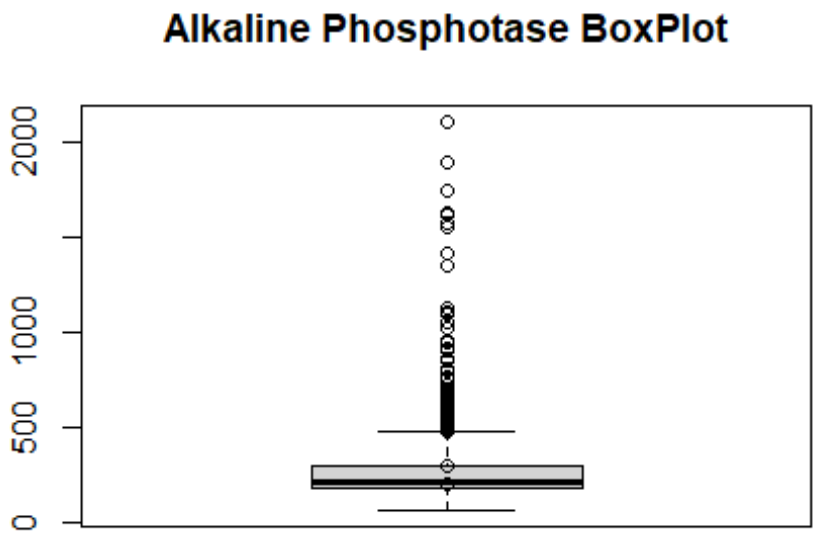
```
v <- c(liver$Alkaline_Phosphotase)
mode <- getMode(v)
print(mode)

## [1] 198
```

From the above summary we can observe that the Mean (291.4), Median (208.0) and Mode (198) are not equal so the distribution is asymmetrical. Here Mean > Median > Mode so the distribution could be positive and skewed to the right. Now we also confirm this as follows:

```
library(moments)
skewness(liver$Alkaline_Phosphotase)

## [1] 3.743771
```

Hence, the skewness is the positive so the distribution is positively skewed was rightly observed. Also I will plot the Boxplot to comparing the distribution of data across dataset as follows:

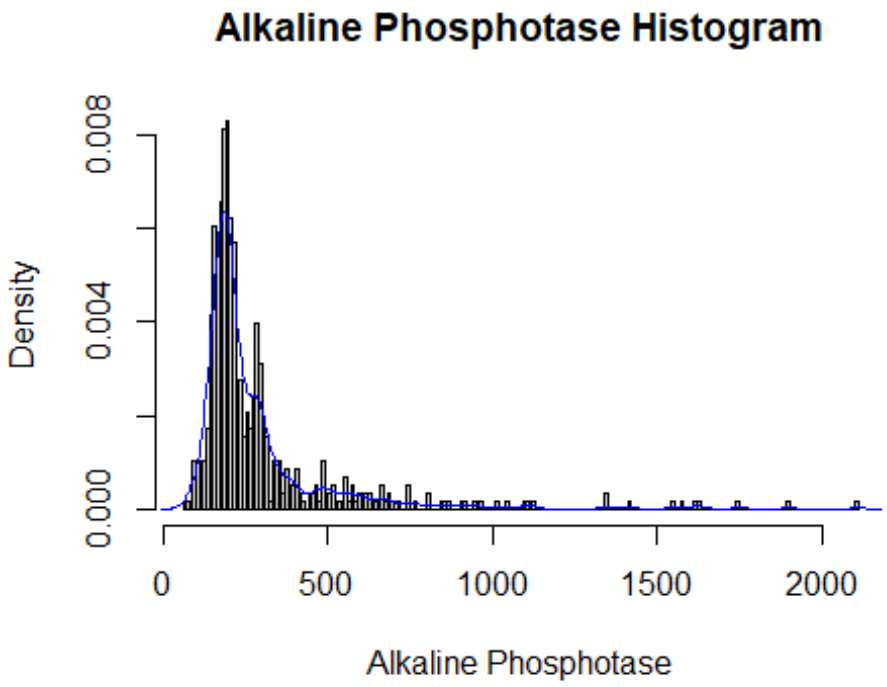
```
boxplot(liver$Alkaline_Phosphotase, main = "Alkaline Phosphotase BoxPlot")
points(mean(liver$Alkaline_Phosphotase))
points(mode)
```



Box plots are graphical displays of the five number summary, So they describe the Minimum (lowest mean relative) lower whisker, Quartile 1 (It separates the lowest 25% observation to the rest of the observations), Median (Which divides the lower 50% observation from the upper 50% observations), Quartile 3 (It divides the upper 25%

observation to the rest of the observations) and Maximum. In the above Box plot diagram, there are many outliers and the upper whisker shows the highest mean relative. Now, I will plot a histogram a graphical display of data using bars of different heights to measures distribution of continuous data as follows:

```
hist(liver$Alkaline_Phosphotase, prob = TRUE, breaks = 262, xlab = "Alkaline
Phosphotase", main = "Alkaline Phosphotase Histogram")
lines(density(liver$Alkaline_Phosphotase), col='blue')
```



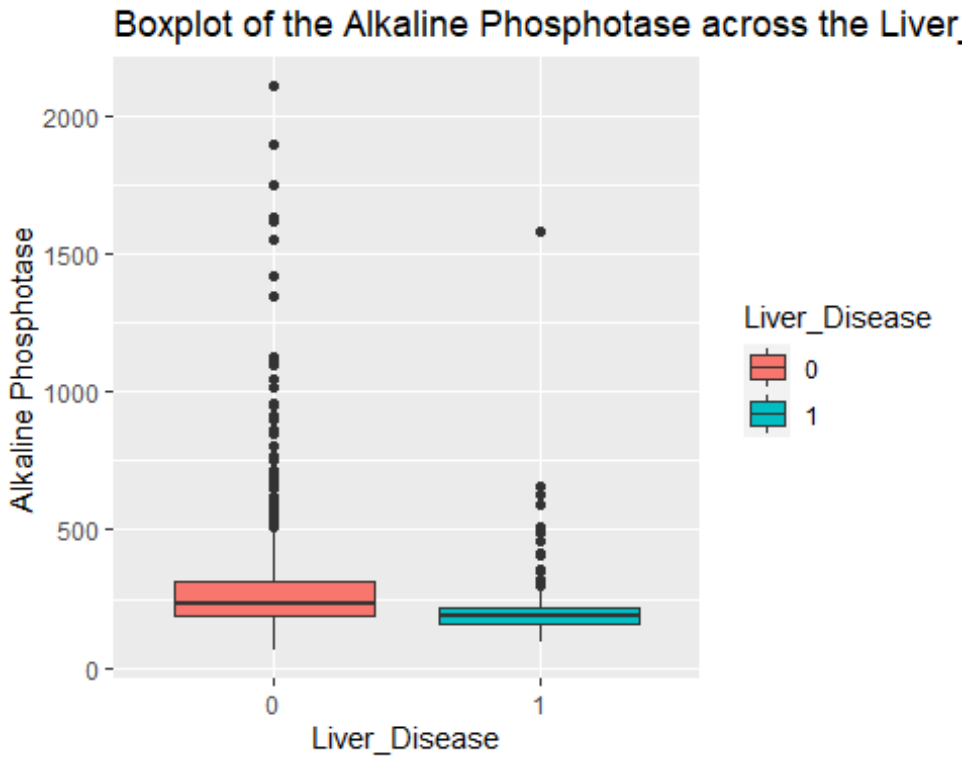
From the above histogram, we can observe that the diagram has many peaks.Using the density lines, I have approximated the histogram graph. Density provides information about the type of distribution under consideration and allows us to determine whether the attribute is distributed normally or not. From the graph above we can see how the `Alkaline_Phosphotase` variable does not seem to fit to a Normal distribution, there is a peak of the frequency distributions around many values and also a right skewed. We have already observed a positive value of `3.743771`, so we will affirm that the distribution is right skewed. To check whether the distribution is `Platykurtic` or not, we will check this by following R method as follows:

```
kurtosis(liver$Alkaline_Phosphotase)

## [1] 20.47242
```

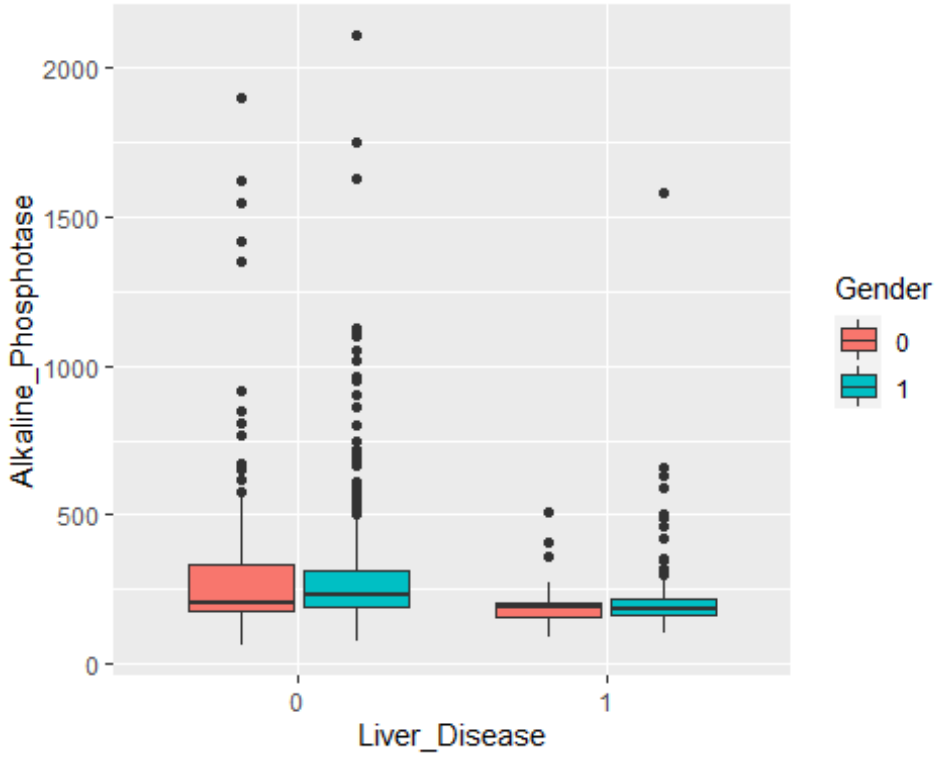
The distribution is also `Leptokurtic`, since the value is greater than 3.

```
ggplot(liver, aes(x = Liver_Disease, y = Alkaline_Phosphotase, fill =
Liver_Disease)) +
```

```
  geom_boxplot() +
  ylab("Alkaline Phosphotase") +
  ggtitle("Boxplot of the Alkaline Phosphotase across the Liver_Disease")
```




```
ggplot(liver, aes(Liver_Disease, Alkaline_Phosphotase)) +
  geom_boxplot(aes(fill = Gender))
```



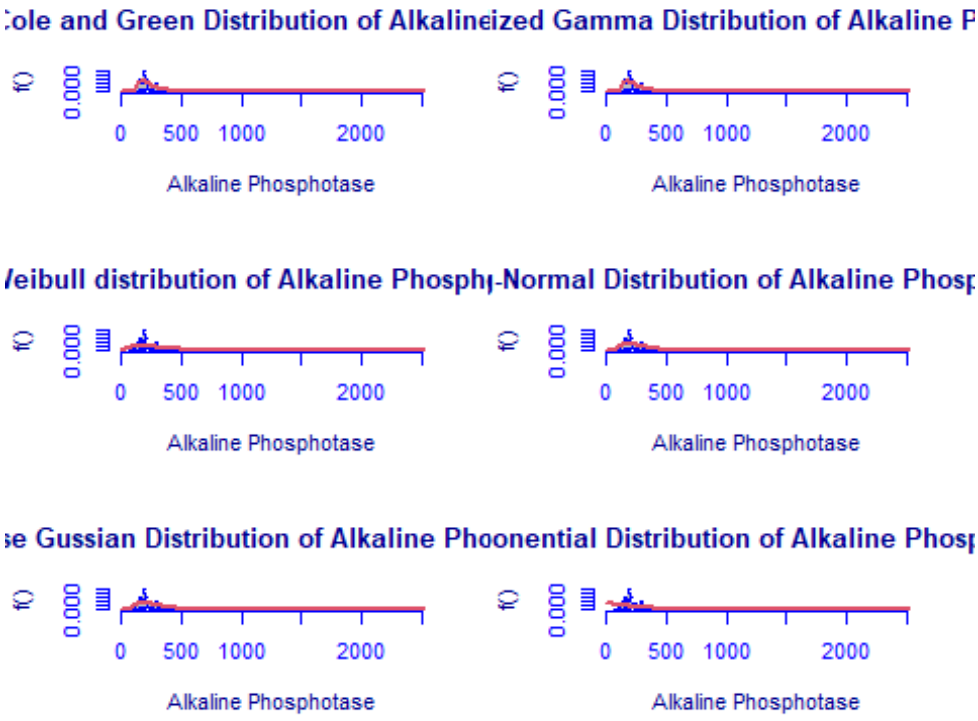
The range for the Alkaline Phosphotase is wide, with a mean of 290.6, median or 208, minimum of 63 and maximum of 2110. The numbers for Alkaline Phosphotase are also skewed as shown in the plot. When you look at the numbers across the response we see a difference across it. The mean is higher when the Liver_Disease is 0 and the range is also wider for Liver_Disease = 0. When you look further and compare by gender, there are differences in the mean within the Liver_Disease = 0, but not much for the Liver_Disease = 1 group.

Alkaline Phosphotase Fit for the Data

Now we will try to fit different models to Alkaline Phosphotase distribution evaluating the Akaike Information Criterion (AIC) and the Bayesian Information criterion (BIC), The lower are the indexes, the better is the fitting. I will evaluate both the single parameter distributions and mixture distributions as follows:

```
library(MASS)
library(gamlss)


par(mfrow=c(3,2))

ap.BCCG <- histDist(liver$Alkaline_Phosphotase, family=BCCG, nbins = 262,
xlab = "Alkaline Phosphotase", main="Box-Cox Cole and Green Distribution of
Alkaline Phosphotase")

ap.GG <- histDist(liver$Alkaline_Phosphotase, family=GG, nbins = 262, xlab =
```

```
"Alkaline Phosphotase", main="Generalized Gamma Distribution of Alkaline
Phosphotase")

ap.WEI <- histDist(liver$Alkaline_Phosphotase, family=WEI, nbins = 262, xlab
= "Alkaline Phosphotase", main="Weibull distribution of Alkaline
Phosphotase")

ap.LOGNO <- histDist(liver$Alkaline_Phosphotase, family=LOGNO, nbins = 262,
xlab = "Alkaline Phosphotase", main="Log-Normal Distribution of Alkaline
Phosphotase")

ap.IG <- histDist(liver$Alkaline_Phosphotase, family=IG, nbins=262, xlab =
"Alkaline Phosphotase", main = "Inverse Gussian Distribution of Alkaline
Phosphotase")

ap.EXP<- histDist(liver$Alkaline_Phosphotase, family=EXP, nbins=262, xlab =
"Alkaline Phosphotase", main = "Exponential Distribution of Alkaline
Phosphotase")
```




```
library(Matrix)
library(glmnet)
df <- data.frame(Rownames = c("Box-Cox Cole and Green","Generalized Gamma",
"Weibull",
                              "Log-Normal", "Inverse Gussian", "Exponential"),
                 AIC = c(AIC(ap.BCCG), AIC(ap.GG), AIC(ap.WEI),
AIC(ap.LOGNO),
```

```
                          AIC(ap.IG), AIC(ap.EXP)),
                    BIC = c(ap.BCCG$sbc, ap.GG$sbc, ap.WEI$sbc,
ap.LOGNO$sbc,
                            ap.IG$sbc, ap.EXP$sbc),
                    df = c(ap.BCCG$df.fit, ap.GG$df.fit, ap.WEI$df.fit,
                           ap.LOGNO$df.fit, ap.IG$df.fit, ap.EXP$df.fit),
                    LogLike = c(logLik(ap.BCCG), logLik(ap.GG),
logLik(ap.WEI),
                                logLik(ap.LOGNO), logLik(ap.IG),
logLik(ap.EXP)))
df
```

```
##                 Rownames      AIC      BIC df   LogLike
## 1 Box-Cox Cole and Green 7132.341 7145.425  3 -3563.171
## 2      Generalized Gamma 7140.210 7153.294  3 -3567.105
## 3                Weibull 7580.311 7589.034  2 -3788.156
## 4             Log-Normal 7272.052 7280.774  2 -3634.026
## 5        Inverse Gussian 7282.613 7291.335  2 -3639.306
## 6            Exponential 7731.164 7735.526  1 -3864.582
```

As we can see, the model with the highest log likelihood (-3563.171) and the lowest AIC (7132.341) and BIC (7145.425) is the Box-Cox Cole and Green Distribution. Therefore, based on the greatest likelihood technique, our data fits better in the Box-Cox Cole and Green Distribution. Now, I will also compare two models by means of the Likelihood ratio test as we performed above as follows:

```
library(zoo)
library(lmtest)
lrtest(ap.BCCG, ap.EXP)
```

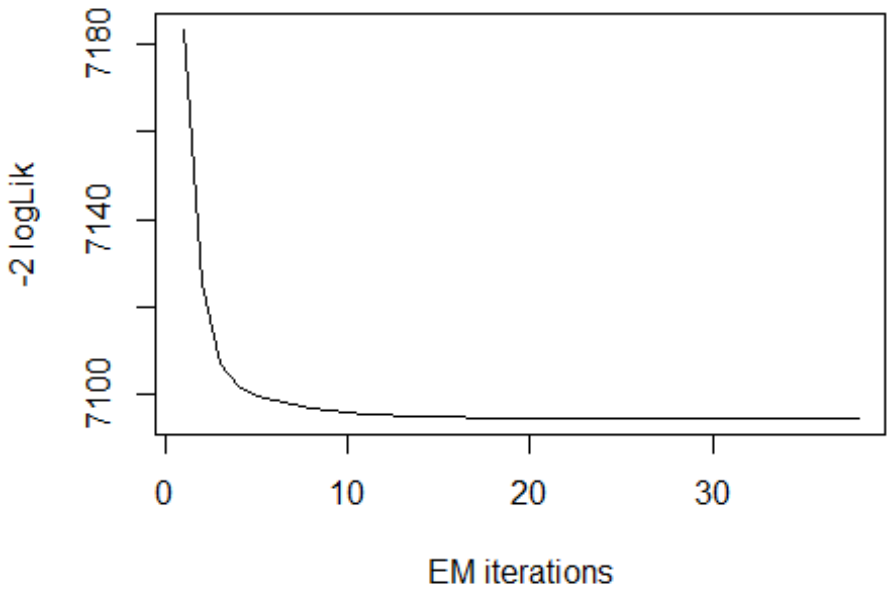
```
## Likelihood ratio test
##
## Model 1: gamlssML(formula = liver$Alkaline_Phosphotase, family = "BCCG")
## Model 2: gamlssML(formula = liver$Alkaline_Phosphotase, family = "EXP")
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   3 -3563.2
## 2   1 -3864.6 -2 602.82  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Under the null hypothesis, we compare the Box-Cox Cole and Green Distribution to the Exponential distribution under the alternative hypothesis. At any level of significance, we can reject the null hypothesis. Because our distributions fit better in the Box-Cox Cole and Green model, we'll use it.

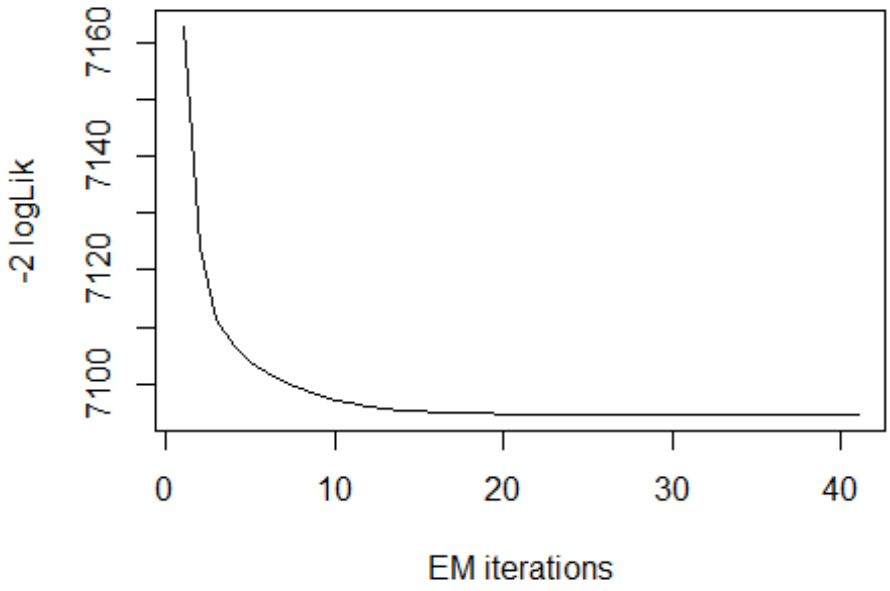
Distributions Mixture

Now, I will also apply the fitting criteria for the distributions with Gamma mixture as follows:

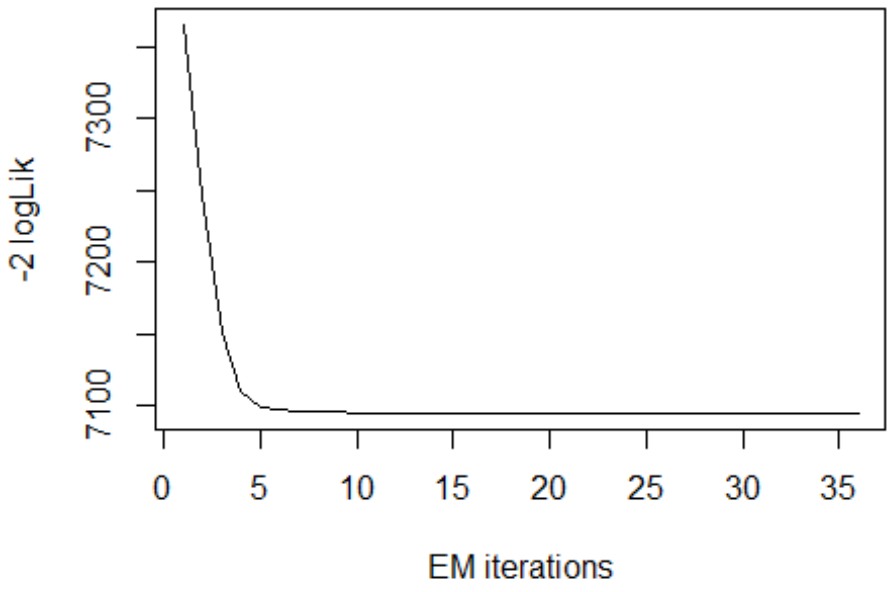
```
library(gamlss.mx)
mix.gam <- gamlssMXfits(n = 5, liver$Alkaline_Phosphotase~1, family = GA, K =
2, data = liver)
```



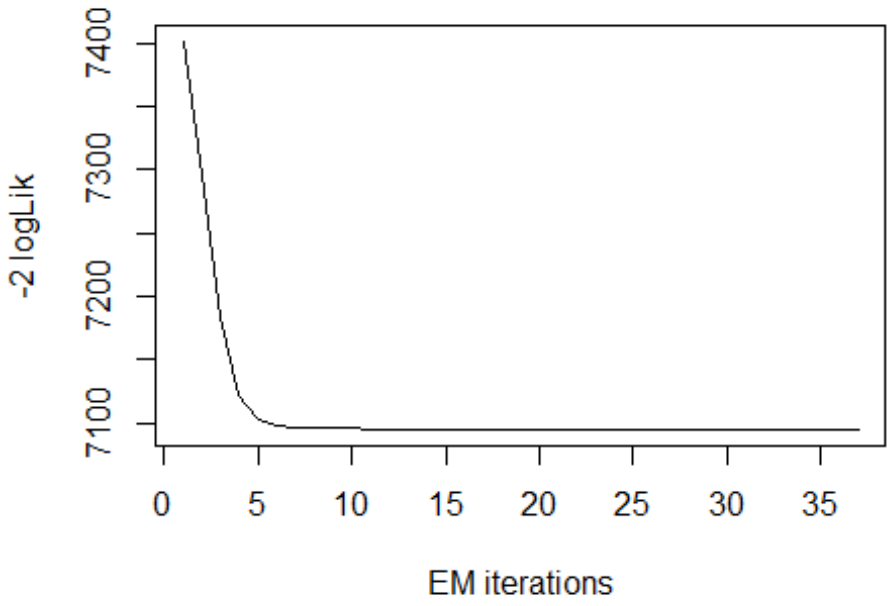

```
## model= 1
```



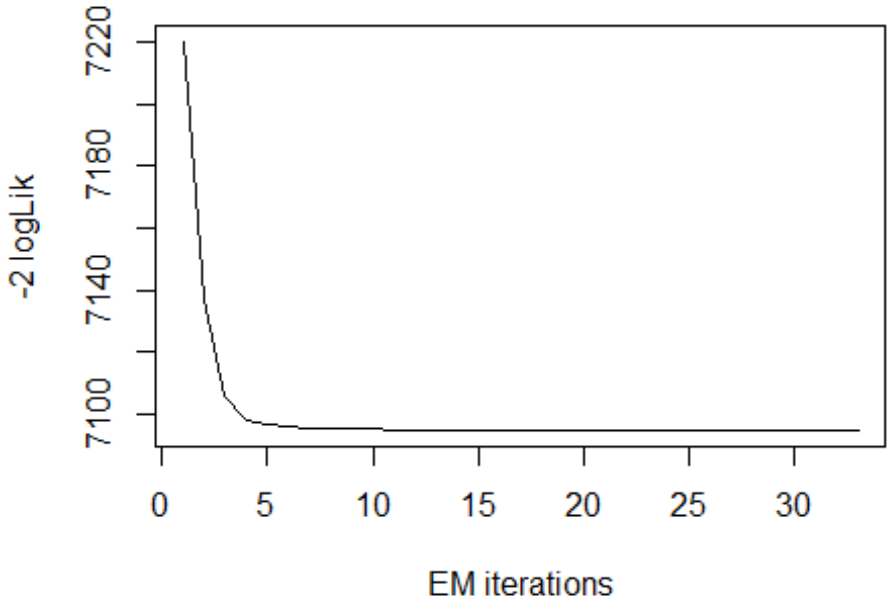

```
## model= 2
```




```
## model= 3
```




```
## model= 4
```




```
## model= 5
```

```
print(mix.gam)

##
## Mixing Family:  c("GA", "GA")
##
## Fitting method: EM algorithm
##
## Call:  gamlssMX(formula = liver$Alkaline_Phosphotase ~ 1,
##     family = GA, K = 2, data = liver)
##
## Mu Coefficients for model: 1
## (Intercept)
##       5.329
## Sigma Coefficients for model: 1
## (Intercept)
##      -1.339
## Mu Coefficients for model: 2
## (Intercept)
##       6.285
## Sigma Coefficients for model: 2
## (Intercept)
##     -0.4683
##
## Estimated probabilities: 0.7422594 0.2577406
##
## Degrees of Freedom for the fit: 5 Residual Deg. of Freedom   574
## Global Deviance:     7094.69
##             AIC:     7104.69
##             SBC:     7126.5
```

We can observe that the AIC value of the mixture of Gamma has improved, since it is lower than that of the Box-Cox Cole and Green Distribution. The current `AIC` value is `7104.69`, whereas the previous value was `7132.341` and the current value of `BIC` is `7126.5` whereas the previous value was `7145.425`.

```
logLik(mix.gam)

## 'log Lik.' -3547.345 (df=5)

mix.gam$prob

## [1] 0.7422594 0.2577406

fitted(mix.gam, "mu")[1]

## [1] 291.3451

fitted(mix.gam, "sigma")[2]

## [1] 291.3451
```

```r
hist(liver$Alkaline_Phosphotase, breaks = 262, xlab = "Alkaline Phosphotase",
main="Mixture of Gamma with k=2", freq = FALSE)

mu.hat1 <- exp(mix.gam[["models"]][[1]][["mu.coefficients"]])

sigma.hat1 <- exp(mix.gam[["models"]][[1]][["sigma.coefficients"]])

mu.hat2 <- exp(mix.gam[["models"]][[2]][["mu.coefficients"]])

sigma.hat2 <- exp(mix.gam[["models"]][[2]][["sigma.coefficients"]])

hist(liver$Alkaline_Phosphotase, breaks = 262, freq = FALSE, plot = FALSE)
```

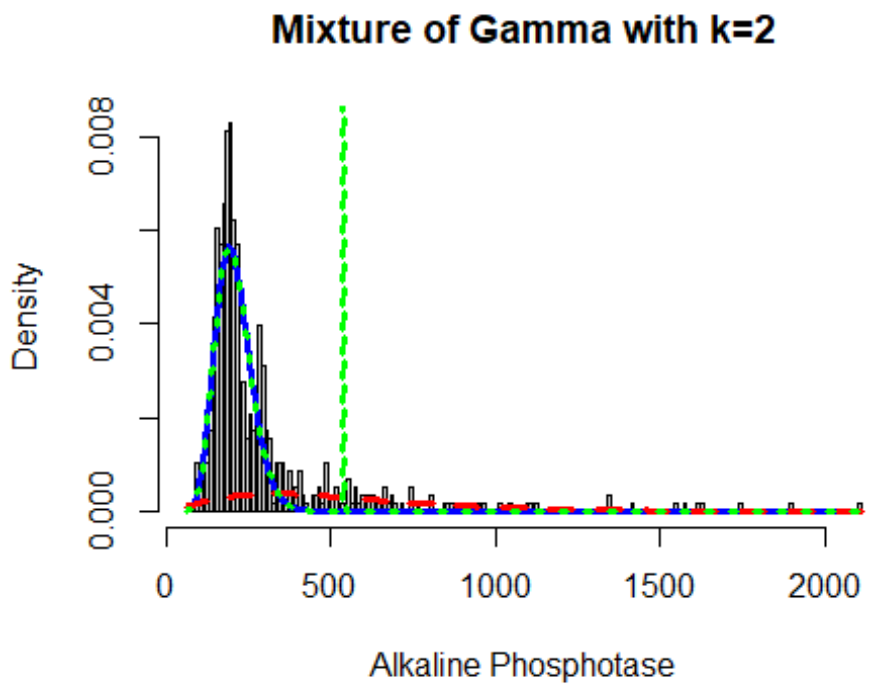
```
## Warning in hist.default(liver$Alkaline_Phosphotase, breaks = 262, freq =
## FALSE, : argument 'freq' is not made use of
```

```
## $breaks
##   [1]   60   70   80   90  100  110  120  130  140  150  160  170  180
190  200
##  [16]  210  220  230  240  250  260  270  280  290  300  310  320  330
340  350
##  [31]  360  370  380  390  400  410  420  430  440  450  460  470  480
490  500
##  [46]  510  520  530  540  550  560  570  580  590  600  610  620  630
640  650
##  [61]  660  670  680  690  700  710  720  730  740  750  760  770  780
790  800
##  [76]  810  820  830  840  850  860  870  880  890  900  910  920  930
940  950
##  [91]  960  970  980  990 1000 1010 1020 1030 1040 1050 1060 1070 1080
1090 1100
## [106] 1110 1120 1130 1140 1150 1160 1170 1180 1190 1200 1210 1220 1230
1240 1250
## [121] 1260 1270 1280 1290 1300 1310 1320 1330 1340 1350 1360 1370 1380
1390 1400
## [136] 1410 1420 1430 1440 1450 1460 1470 1480 1490 1500 1510 1520 1530
1540 1550
## [151] 1560 1570 1580 1590 1600 1610 1620 1630 1640 1650 1660 1670 1680
1690 1700
## [166] 1710 1720 1730 1740 1750 1760 1770 1780 1790 1800 1810 1820 1830
1840 1850
## [181] 1860 1870 1880 1890 1900 1910 1920 1930 1940 1950 1960 1970 1980
1990 2000
## [196] 2010 2020 2030 2040 2050 2060 2070 2080 2090 2100 2110
##
## $counts
##   [1]  1  1  1  6  6  4  6 10 24 35 33 38 47 48 36 33 12 16  9 12 10 14 23
18 10
##  [26]  9  1  6  6  6  2  5  3  3  5  2  1  0  2  2  3  1  6  2  2  3  1  1
1  4
```

```
##  [51]  1  3  1  2  1  2  2  0  1  0  3  1  2  1  0  1  0  0  3  0  1  0  0
0  2
##  [76]  0  0  0  1  1  1  0  0  0  1  1  0  0  1  0  1  0  0  0  0  1  0  0
1  0
## [101]  0  0  0  1  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0
## [126]  0  0  0  2  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0
1  0
## [151]  0  1  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0
0  0
## [176]  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0
## [201]  0  0  0  0  1
##
## $density
##   [1] 0.0001727116 0.0001727116 0.0001727116 0.0010362694 0.0010362694
##   [6] 0.0006908463 0.0010362694 0.0017271157 0.0041450777 0.0060449050
##  [11] 0.0056994819 0.0065630397 0.0081174439 0.0082901554 0.0062176166
##  [16] 0.0056994819 0.0020725389 0.0027633851 0.0015544041 0.0020725389
##  [21] 0.0017271157 0.0024179620 0.0039723661 0.0031088083 0.0017271157
##  [26] 0.0015544041 0.0001727116 0.0010362694 0.0010362694 0.0010362694
##  [31] 0.0003454231 0.0008635579 0.0005181347 0.0005181347 0.0008635579
##  [36] 0.0003454231 0.0001727116 0.0000000000 0.0003454231 0.0003454231
##  [41] 0.0005181347 0.0001727116 0.0010362694 0.0003454231 0.0003454231
##  [46] 0.0005181347 0.0001727116 0.0001727116 0.0001727116 0.0006908463
##  [51] 0.0001727116 0.0005181347 0.0001727116 0.0003454231 0.0001727116
##  [56] 0.0003454231 0.0003454231 0.0000000000 0.0001727116 0.0000000000
##  [61] 0.0005181347 0.0001727116 0.0003454231 0.0001727116 0.0000000000
##  [66] 0.0001727116 0.0000000000 0.0000000000 0.0005181347 0.0000000000
##  [71] 0.0001727116 0.0000000000 0.0000000000 0.0000000000 0.0003454231
##  [76] 0.0000000000 0.0000000000 0.0000000000 0.0001727116 0.0001727116
##  [81] 0.0001727116 0.0000000000 0.0000000000 0.0000000000 0.0001727116
##  [86] 0.0001727116 0.0000000000 0.0000000000 0.0001727116 0.0000000000
##  [91] 0.0001727116 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##  [96] 0.0001727116 0.0000000000 0.0000000000 0.0001727116 0.0000000000
## [101] 0.0000000000 0.0000000000 0.0000000000 0.0001727116 0.0001727116
## [106] 0.0000000000 0.0001727116 0.0000000000 0.0000000000 0.0000000000
## [111] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [116] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [121] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [126] 0.0000000000 0.0000000000 0.0000000000 0.0003454231 0.0000000000
## [131] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [136] 0.0001727116 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [141] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [146] 0.0000000000 0.0000000000 0.0000000000 0.0001727116 0.0000000000
## [151] 0.0000000000 0.0001727116 0.0000000000 0.0000000000 0.0000000000
## [156] 0.0001727116 0.0001727116 0.0000000000 0.0000000000 0.0000000000
## [161] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [166] 0.0000000000 0.0000000000 0.0000000000 0.0001727116 0.0000000000
## [171] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
```

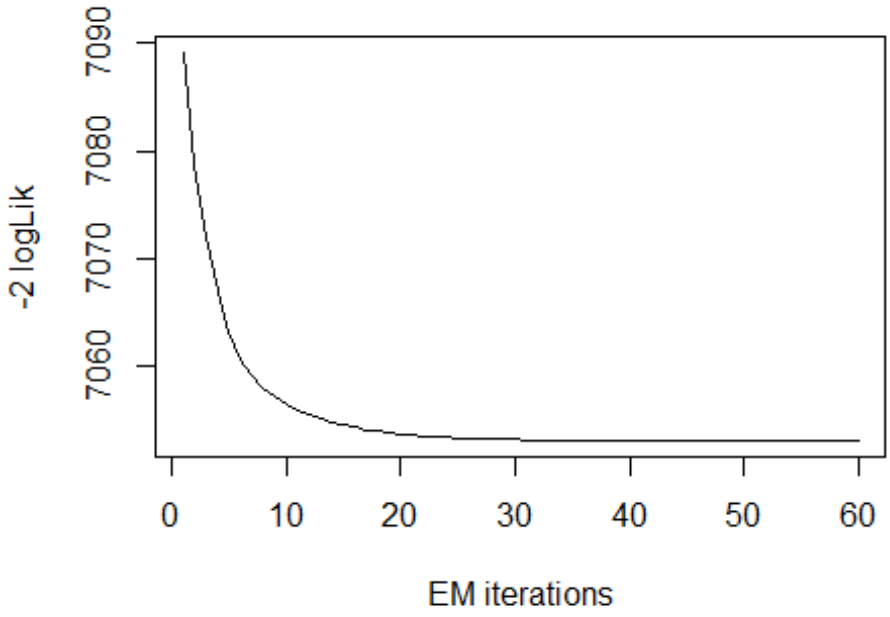
```
## [176] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [181] 0.0000000000 0.0000000000 0.0000000000 0.0001727116 0.0000000000
## [186] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [191] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [196] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [201] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0001727116
##
## $mids
##   [1]   65   75   85   95  105  115  125  135  145  155  165  175  185
195  205
##  [16]  215  225  235  245  255  265  275  285  295  305  315  325  335
345  355
##  [31]  365  375  385  395  405  415  425  435  445  455  465  475  485
495  505
##  [46]  515  525  535  545  555  565  575  585  595  605  615  625  635
645  655
##  [61]  665  675  685  695  705  715  725  735  745  755  765  775  785
795  805
##  [76]  815  825  835  845  855  865  875  885  895  905  915  925  935
945  955
##  [91]  965  975  985  995 1005 1015 1025 1035 1045 1055 1065 1075 1085
1095 1105
## [106] 1115 1125 1135 1145 1155 1165 1175 1185 1195 1205 1215 1225 1235
1245 1255
## [121] 1265 1275 1285 1295 1305 1315 1325 1335 1345 1355 1365 1375 1385
1395 1405
## [136] 1415 1425 1435 1445 1455 1465 1475 1485 1495 1505 1515 1525 1535
1545 1555
## [151] 1565 1575 1585 1595 1605 1615 1625 1635 1645 1655 1665 1675 1685
1695 1705
## [166] 1715 1725 1735 1745 1755 1765 1775 1785 1795 1805 1815 1825 1835
1845 1855
## [181] 1865 1875 1885 1895 1905 1915 1925 1935 1945 1955 1965 1975 1985
1995 2005
## [196] 2015 2025 2035 2045 2055 2065 2075 2085 2095 2105
##
## $xname
## [1] "liver$Alkaline_Phosphotase"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"

lines(seq(min(liver$Alkaline_Phosphotase), max(liver$Alkaline_Phosphotase),
      length = length(liver$Alkaline_Phosphotase)),
      mix.gam[["prob"]][1]*dGA(seq(min(liver$Alkaline_Phosphotase),
      max(liver$Alkaline_Phosphotase), length =
length(liver$Alkaline_Phosphotase)),
```

```
      mu = mu.hat1, sigma = sigma.hat1), lty=1, lwd=3, col="blue")

lines(seq(min(liver$Alkaline_Phosphotase), max(liver$Alkaline_Phosphotase),
      length = length(liver$Alkaline_Phosphotase)),
      mix.gam[["prob"]][2]*dGA(seq(min(liver$Alkaline_Phosphotase),
      max(liver$Alkaline_Phosphotase), length =
length(liver$Alkaline_Phosphotase)),
      mu = mu.hat2, sigma = sigma.hat2), lty = 2, lwd = 3, col = "red")

lines(seq(min(liver$Alkaline_Phosphotase), max(liver$Alkaline_Phosphotase),
      length = length(liver$Alkaline_Phosphotase)),
      mix.gam[["prob"]][1]*dGA(seq(min(liver$Alkaline_Phosphotase),
      max(liver$Alkaline_Phosphotase), length =
length(liver$Alkaline_Phosphotase)),
      mu = mu.hat1, sigma = sigma.hat1)+
      mix.gam[["prob"]][2]*dRG(seq(min(liver$Alkaline_Phosphotase),
      max(liver$Alkaline_Phosphotase), length =
length(liver$Alkaline_Phosphotase)),
      mu = mu.hat2, sigma = sigma.hat2), lty = 3, lwd = 3, col = "green")
```

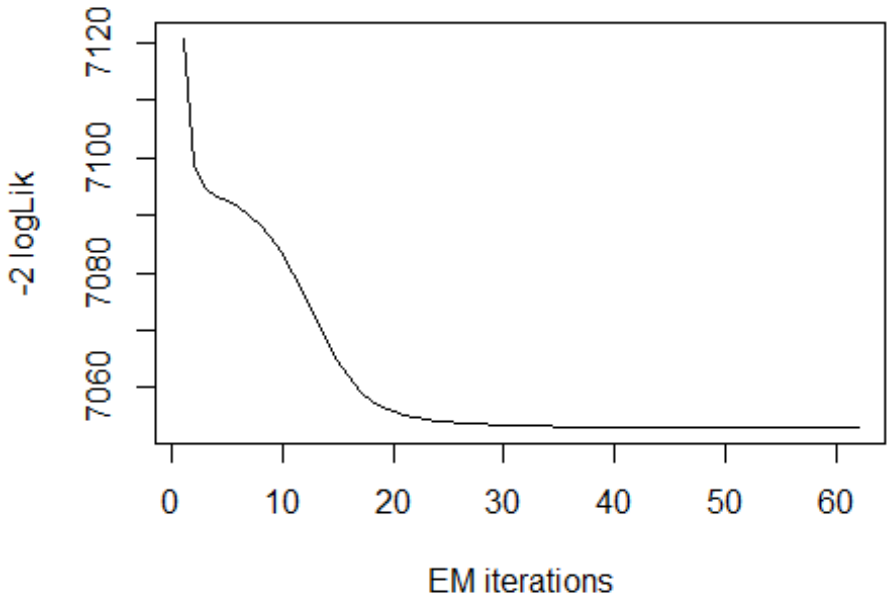


Now, I will again apply the fitting criteria for the distributions with Gamma mixture with the k = 3 as follows:

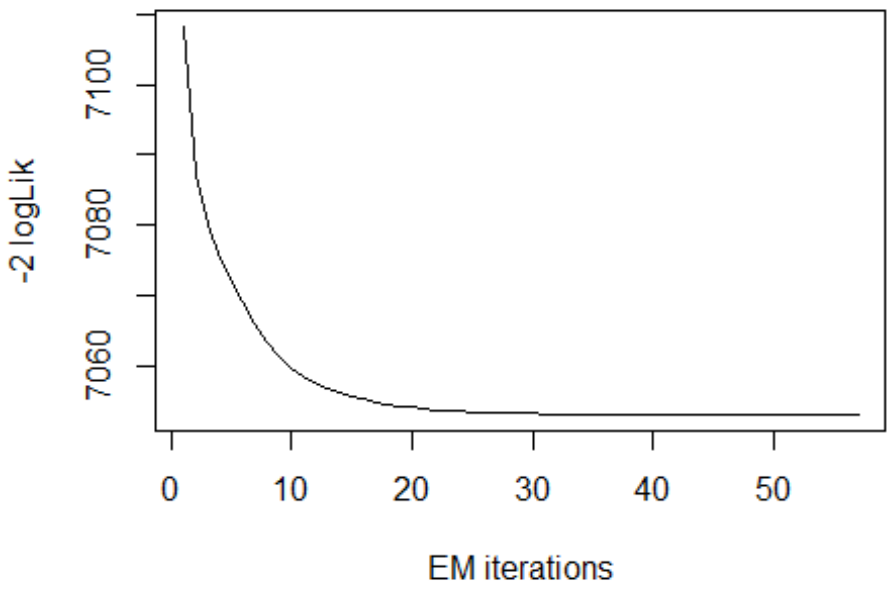
```
mix.gam.3 <- gamlssMXfits(n = 5, liver$Alkaline_Phosphotase~1, family = GA, K
= 3, data = liver)
```



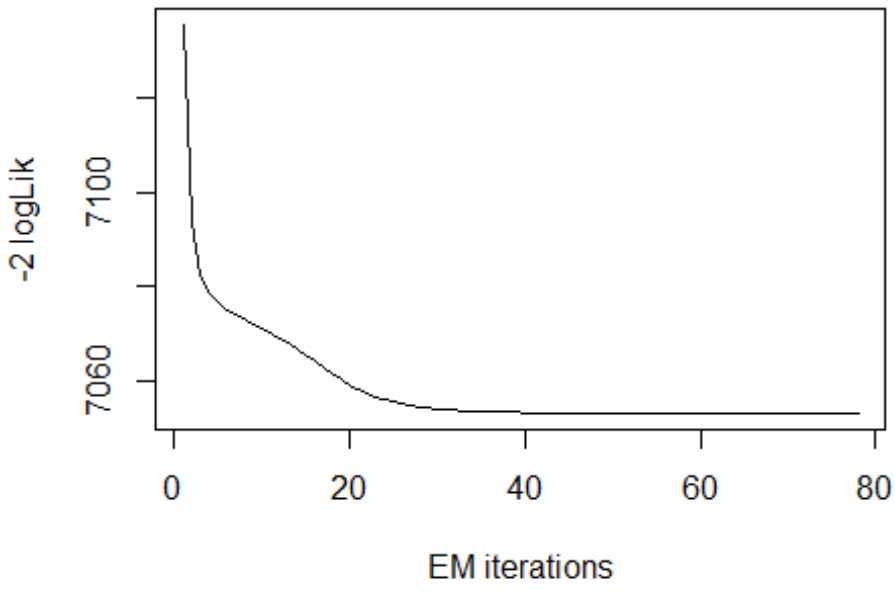

```
## model= 1
```



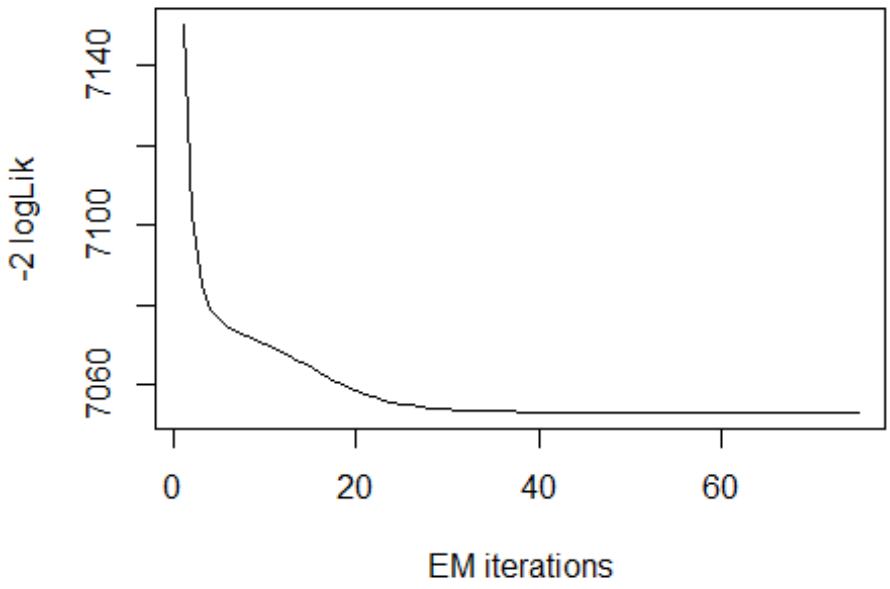

```
## model= 2
```




```
## model= 3
```




```
## model= 4
```




```
## model= 5

print(mix.gam.3)

##
## Mixing Family:  c("GA", "GA", "GA")
##
## Fitting method: EM algorithm
##
## Call:  gamlssMX(formula = liver$Alkaline_Phosphotase ~ 1,
##     family = GA, K = 3, data = liver)
##
## Mu Coefficients for model: 1
## (Intercept)
##       5.477
## Sigma Coefficients for model: 1
## (Intercept)
##      -1.057
## Mu Coefficients for model: 2
## (Intercept)
##       6.424
## Sigma Coefficients for model: 2
## (Intercept)
##     -0.4791
## Mu Coefficients for model: 3
## (Intercept)
##       5.208
```

```
## Sigma Coefficients for model: 3
## (Intercept)
##      -2.079
##
## Estimated probabilities: 0.4881806 0.1869186 0.3249008
##
## Degrees of Freedom for the fit: 8 Residual Deg. of Freedom   571
## Global Deviance:     7053.09
##             AIC:     7069.09
##             SBC:     7103.99
```

```
logLik(mix.gam.3)
```

```
## 'log Lik.' -3526.547 (df=8)
```

```
mix.gam.3$prob
```

```
## [1] 0.4881806 0.1869186 0.3249008
```

```
fitted(mix.gam.3, "mu")[1]
```

```
## [1] 291.3513
```

```
fitted(mix.gam.3, "sigma")[2]
```

```
## [1] 291.3513
```

```
hist(liver$Alkaline_Phosphotase, breaks = 262, xlab = "Alkaline Phosphotase",
main="Mixture of Gamma with k=3", freq = FALSE)
```

```
mu.hat1 <- exp(mix.gam.3[["models"]][[1]][["mu.coefficients"]])
```

```
sigma.hat1 <- exp(mix.gam.3[["models"]][[1]][["sigma.coefficients"]])
```

```
mu.hat2 <- exp(mix.gam.3[["models"]][[2]][["mu.coefficients"]])
```

```
sigma.hat2 <- exp(mix.gam.3[["models"]][[2]][["sigma.coefficients"]])
```

```
hist(liver$Alkaline_Phosphotase, breaks = 262, freq = FALSE, plot = FALSE)
```

```
## Warning in hist.default(liver$Alkaline_Phosphotase, breaks = 262, freq =
## FALSE, : argument 'freq' is not made use of
```

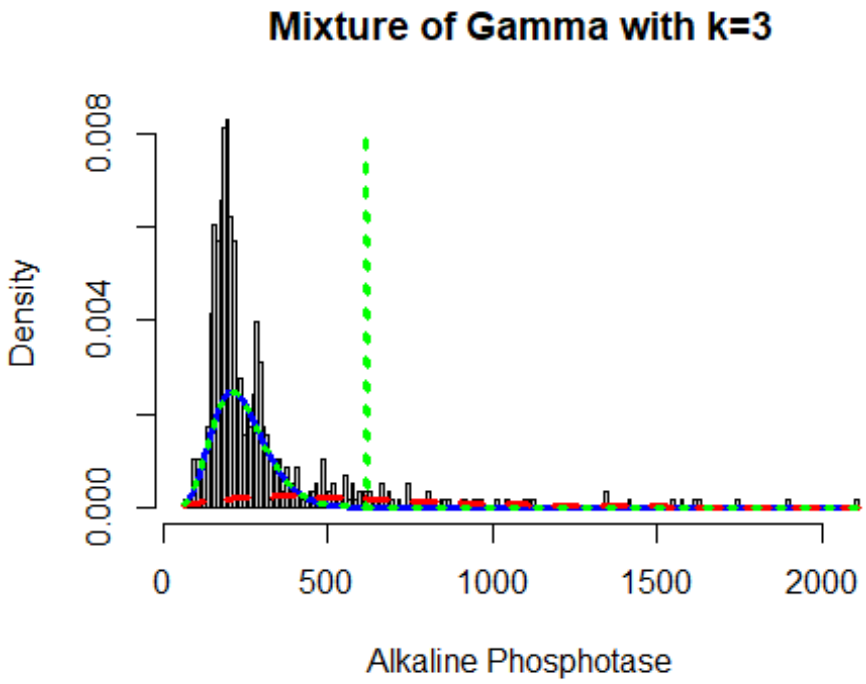
```
## $breaks
##   [1]   60   70   80   90  100  110  120  130  140  150  160  170  180
190  200
##  [16]  210  220  230  240  250  260  270  280  290  300  310  320  330
340  350
##  [31]  360  370  380  390  400  410  420  430  440  450  460  470  480
490  500
##  [46]  510  520  530  540  550  560  570  580  590  600  610  620  630
640  650
```

```
##  [61]  660  670  680  690  700  710  720  730  740  750  760  770  780
790  800
##  [76]  810  820  830  840  850  860  870  880  890  900  910  920  930
940  950
##  [91]  960  970  980  990 1000 1010 1020 1030 1040 1050 1060 1070 1080
1090 1100
## [106] 1110 1120 1130 1140 1150 1160 1170 1180 1190 1200 1210 1220 1230
1240 1250
## [121] 1260 1270 1280 1290 1300 1310 1320 1330 1340 1350 1360 1370 1380
1390 1400
## [136] 1410 1420 1430 1440 1450 1460 1470 1480 1490 1500 1510 1520 1530
1540 1550
## [151] 1560 1570 1580 1590 1600 1610 1620 1630 1640 1650 1660 1670 1680
1690 1700
## [166] 1710 1720 1730 1740 1750 1760 1770 1780 1790 1800 1810 1820 1830
1840 1850
## [181] 1860 1870 1880 1890 1900 1910 1920 1930 1940 1950 1960 1970 1980
1990 2000
## [196] 2010 2020 2030 2040 2050 2060 2070 2080 2090 2100 2110
##
## $counts
##   [1]  1  1  1  6  6  4  6 10 24 35 33 38 47 48 36 33 12 16  9 12 10 14 23
18 10
##  [26]  9  1  6  6  6  2  5  3  3  5  2  1  0  2  2  3  1  6  2  2  3  1  1
1  4
##  [51]  1  3  1  2  1  2  2  0  1  0  3  1  2  1  0  1  0  0  3  0  1  0  0
0  2
##  [76]  0  0  0  1  1  1  0  0  0  1  1  0  0  1  0  1  0  0  0  0  1  0  0
1  0
## [101]  0  0  0  1  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0
## [126]  0  0  0  2  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0
1  0
## [151]  0  1  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0
0  0
## [176]  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0
## [201]  0  0  0  0  1
##
## $density
##   [1] 0.0001727116 0.0001727116 0.0001727116 0.0010362694 0.0010362694
##   [6] 0.0006908463 0.0010362694 0.0017271157 0.0041450777 0.0060449050
##  [11] 0.0056994819 0.0065630397 0.0081174439 0.0082901554 0.0062176166
##  [16] 0.0056994819 0.0020725389 0.0027633851 0.0015544041 0.0020725389
##  [21] 0.0017271157 0.0024179620 0.0039723661 0.0031088083 0.0017271157
##  [26] 0.0015544041 0.0001727116 0.0010362694 0.0010362694 0.0010362694
##  [31] 0.0003454231 0.0008635579 0.0005181347 0.0005181347 0.0008635579
##  [36] 0.0003454231 0.0001727116 0.0000000000 0.0003454231 0.0003454231
##  [41] 0.0005181347 0.0001727116 0.0010362694 0.0003454231 0.0003454231
##  [46] 0.0005181347 0.0001727116 0.0001727116 0.0001727116 0.0006908463
```

```
##  [51] 0.0001727116 0.0005181347 0.0001727116 0.0003454231 0.0001727116
##  [56] 0.0003454231 0.0003454231 0.0000000000 0.0001727116 0.0000000000
##  [61] 0.0005181347 0.0001727116 0.0003454231 0.0001727116 0.0000000000
##  [66] 0.0001727116 0.0000000000 0.0000000000 0.0005181347 0.0000000000
##  [71] 0.0001727116 0.0000000000 0.0000000000 0.0000000000 0.0003454231
##  [76] 0.0000000000 0.0000000000 0.0000000000 0.0001727116 0.0001727116
##  [81] 0.0001727116 0.0000000000 0.0000000000 0.0000000000 0.0001727116
##  [86] 0.0001727116 0.0000000000 0.0000000000 0.0001727116 0.0000000000
##  [91] 0.0001727116 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##  [96] 0.0001727116 0.0000000000 0.0000000000 0.0001727116 0.0000000000
## [101] 0.0000000000 0.0000000000 0.0000000000 0.0001727116 0.0001727116
## [106] 0.0000000000 0.0001727116 0.0000000000 0.0000000000 0.0000000000
## [111] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [116] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [121] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [126] 0.0000000000 0.0000000000 0.0000000000 0.0003454231 0.0000000000
## [131] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [136] 0.0001727116 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [141] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [146] 0.0000000000 0.0000000000 0.0000000000 0.0001727116 0.0000000000
## [151] 0.0000000000 0.0001727116 0.0000000000 0.0000000000 0.0000000000
## [156] 0.0001727116 0.0001727116 0.0000000000 0.0000000000 0.0000000000
## [161] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [166] 0.0000000000 0.0000000000 0.0000000000 0.0001727116 0.0000000000
## [171] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [176] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [181] 0.0000000000 0.0000000000 0.0000000000 0.0001727116 0.0000000000
## [186] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [191] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [196] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [201] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0001727116
##
## $mids
##   [1]   65   75   85   95  105  115  125  135  145  155  165  175  185
195  205
##  [16]  215  225  235  245  255  265  275  285  295  305  315  325  335
345  355
##  [31]  365  375  385  395  405  415  425  435  445  455  465  475  485
495  505
##  [46]  515  525  535  545  555  565  575  585  595  605  615  625  635
645  655
##  [61]  665  675  685  695  705  715  725  735  745  755  765  775  785
795  805
##  [76]  815  825  835  845  855  865  875  885  895  905  915  925  935
945  955
##  [91]  965  975  985  995 1005 1015 1025 1035 1045 1055 1065 1075 1085
1095 1105
## [106] 1115 1125 1135 1145 1155 1165 1175 1185 1195 1205 1215 1225 1235
1245 1255
## [121] 1265 1275 1285 1295 1305 1315 1325 1335 1345 1355 1365 1375 1385
```

```
1395 1405
## [136] 1415 1425 1435 1445 1455 1465 1475 1485 1495 1505 1515 1525 1535
1545 1555
## [151] 1565 1575 1585 1595 1605 1615 1625 1635 1645 1655 1665 1675 1685
1695 1705
## [166] 1715 1725 1735 1745 1755 1765 1775 1785 1795 1805 1815 1825 1835
1845 1855
## [181] 1865 1875 1885 1895 1905 1915 1925 1935 1945 1955 1965 1975 1985
1995 2005
## [196] 2015 2025 2035 2045 2055 2065 2075 2085 2095 2105
##
## $xname
## [1] "liver$Alkaline_Phosphotase"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```r
lines(seq(min(liver$Alkaline_Phosphotase), max(liver$Alkaline_Phosphotase),
      length = length(liver$Alkaline_Phosphotase)),
      mix.gam.3[["prob"]][1]*dGA(seq(min(liver$Alkaline_Phosphotase),
      max(liver$Alkaline_Phosphotase), length =
length(liver$Alkaline_Phosphotase)),
      mu = mu.hat1, sigma = sigma.hat1), lty=1, lwd=3, col="blue")

lines(seq(min(liver$Alkaline_Phosphotase), max(liver$Alkaline_Phosphotase),
      length = length(liver$Alkaline_Phosphotase)),
      mix.gam.3[["prob"]][2]*dGA(seq(min(liver$Alkaline_Phosphotase),
      max(liver$Alkaline_Phosphotase), length =
length(liver$Alkaline_Phosphotase)),
      mu = mu.hat2, sigma = sigma.hat2), lty = 2, lwd = 3, col = "red")

lines(seq(min(liver$Alkaline_Phosphotase), max(liver$Alkaline_Phosphotase),
      length = length(liver$Alkaline_Phosphotase)),
      mix.gam.3[["prob"]][1]*dGA(seq(min(liver$Alkaline_Phosphotase),
      max(liver$Alkaline_Phosphotase), length =
length(liver$Alkaline_Phosphotase)),
      mu = mu.hat1, sigma = sigma.hat1)+
      mix.gam.3[["prob"]][2]*dRG(seq(min(liver$Alkaline_Phosphotase),
      max(liver$Alkaline_Phosphotase), length =
length(liver$Alkaline_Phosphotase)),
      mu = mu.hat2, sigma = sigma.hat2), lty = 3, lwd = 3, col = "green")
```




```
mix.gm.tb <- data.frame(row.names=c('Gamma Mixture, K=3','Gamma Mixture,
K=2', "Box-Cox Cole and Green Distribution"),
                        AIC=c(mix.gam.3$aic, mix.gam$aic, AIC(ap.BCCG)),
                        BIC=c(mix.gam.3$sbc, mix.gam$sbc, ap.BCCG$sbc))
mix.gm.tb
```

```
##                                          AIC      BIC
## Gamma Mixture, K=3                  7069.095 7103.985
## Gamma Mixture, K=2                  7104.690 7126.497
## Box-Cox Cole and Green Distribution 7132.341 7145.425
```

We can observe that the AIC value of the mixture of Gamma with k=3 and k=2 has improved, since values are lower than that of the Box-Cox Cole and Green Distribution and Gamma mixture with k=2. The previous AIC value was `7104.690`, whereas the current value which is lower is `7069.095` and the previous BIC value was `7126.497`, whereas the current value which is lower is `7103.986`. Here we can clearly see that our data fits better in the Gamma Mixture Distribution with k=3.

Alamine Aminotransferase

Lets explore the `Alamine Aminotransferase` variable:

```
head(liver$Alamine_Aminotransferase)
```

```
## [1] 16 64 60 14 27 19
```

```
length(liver$Alamine_Aminotransferase)
```

```
## [1] 579
```

```
table(liver$Alamine_Aminotransferase)
```

```
##
##   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24
25
##    4    2    9    4    8   14    8    8   17    6   23   17   18    8   12
24
##   26   27   28   29   30   31   32   33   34   35   36   37   38   39   40
41
##   10    9   17   12   15   12   12   10    3    9   11    9    8    4    9
5
##   42   43   44   45   46   47   48   49   50   51   52   53   54   55   56
57
##    9    4    3    7    5    3   14    1   10    1    6    5    3    5    4
2
##   58   59   60   61   62   63   64   65   67   68   69   70   71   72   74
75
##    4    3    7    3    5    2    4    2    2    1    3    2    1    2    4
1
##   76   78   79   80   82   84   85   86   88   89   90   91   93   94   95
96
##    1    1    3    3    1    2    3    1    2    1    1    3    1    1    2
2
##   97   99  102  107  110  112  114  115  116  118  119  120  123  126  131
132
##    1    1    4    1    2    1    1    1    1    1    3    1    1    1    1
1
##  133  137  139  140  141  142  148  149  152  154  155  157  159  160  166
168
##    2    1    1    2    1    1    1    1    1    1    1    1    1    1    1
2
##  173  178  179  181  189  190  194  196  198  205  213  220  230  232  233
284
##    1    1    1    1    1    2    1    1    1    1    1    1    1    2    1
1
##  308  321  322  349  378  382  390  404  407  412  425  440  482  509  622
779
##    1    1    1    1    1    1    1    1    1    2    1    1    1    1    1
1
##  790  875  950 1250 1350 1630 1680 2000
##    1    2    1    2    1    1    1    1
```

The table() function in R returns the absolute frequencies for each patient-specified value in the data set.

```
unique(liver$Alamine_Aminotransferase)
```

```
##   [1]   16   64   60   14   27   19   22   53   51   31   61   91  168
15  232
```

```
##  [16]   17  116   52  875 1680   20   13   45   35   59  102   18   38
123   33
##  [31]   42   25  407   48   36 1630   39   21   80   86   26   24   37
40   62
##  [46]   55  166  189   95   12  194   58   28  119  412  404  220  126
190   97
##  [61]  308   32   29   11   63  181   88   74 2000 1350 1250  482  322
133   46
##  [76]   57   50   34   72   84   30   70  140   99   43  378  112   71
23   79
##  [91]  114  118  107  790  950   82   41   56   85  149  230   69   90
89  148
## [106]   65  205   96  152  390   10  120   78  178  179   47  160   54
198   44
## [121]  349  110  115   94  142  137  155  157  141  284  440   93   76
49  425
## [136]  159  622  779  132  154  196   68  509   67  139  382   75  321
233  173
## [151]  213  131
```

```
length(unique(liver$Alamine_Aminotransferase))
```

```
## [1] 152
```

```
min(liver$Alamine_Aminotransferase)
```

```
## [1] 10
```

```
max(liver$Alamine_Aminotransferase)
```

```
## [1] 2000
```

Here the total observation of Age is 579 and is a continuous variable that range lies between 10-2000. Now, we will see the summary of the Alamine_Aminotransferase variable, for further analysis as follows:

```
summary(liver$Alamine_Aminotransferase)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.00   23.00   35.00   81.13   61.00 2000.00
```

```
sd(liver$Alamine_Aminotransferase)
```

```
## [1] 183.1828
```

```
var(liver$Alamine_Aminotransferase)
```

```
## [1] 33555.95
```

```
v <- c(liver$Alamine_Aminotransferase)
mode <- getMode(v)
print(mode)
```

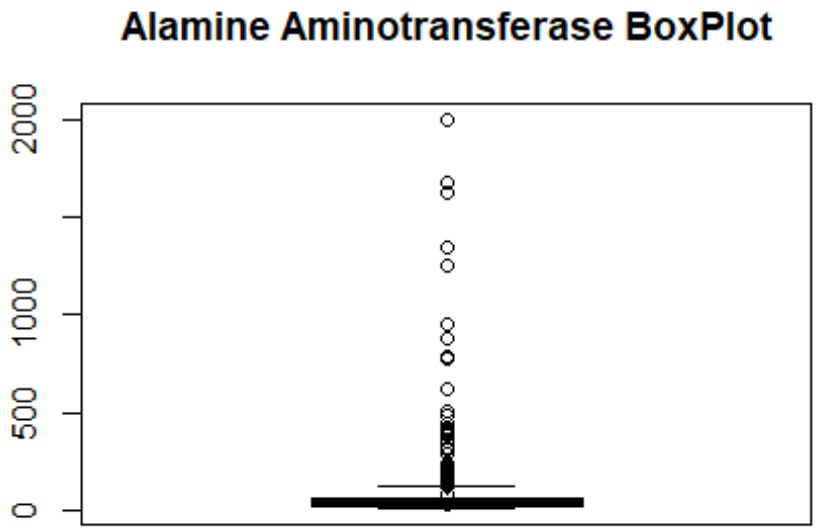
```
## [1] 25
```

From the above summary we can observe that the Mean (81.13), Median (35.00) and Mode (25) are not equal so the distribution is asymmetrical. Here Mean > Median > Mode so the distribution could be positive and skewed to the right. Now we also confirm this as follows:

```
library(moments)
skewness(liver$Alamine_Aminotransferase)

## [1] 6.510652
```

Hence, the skewness is the positive so the distribution is positively skewed was rightly observed. Also I will plot the Boxplot to comparing the distribution of data across data set as follows:

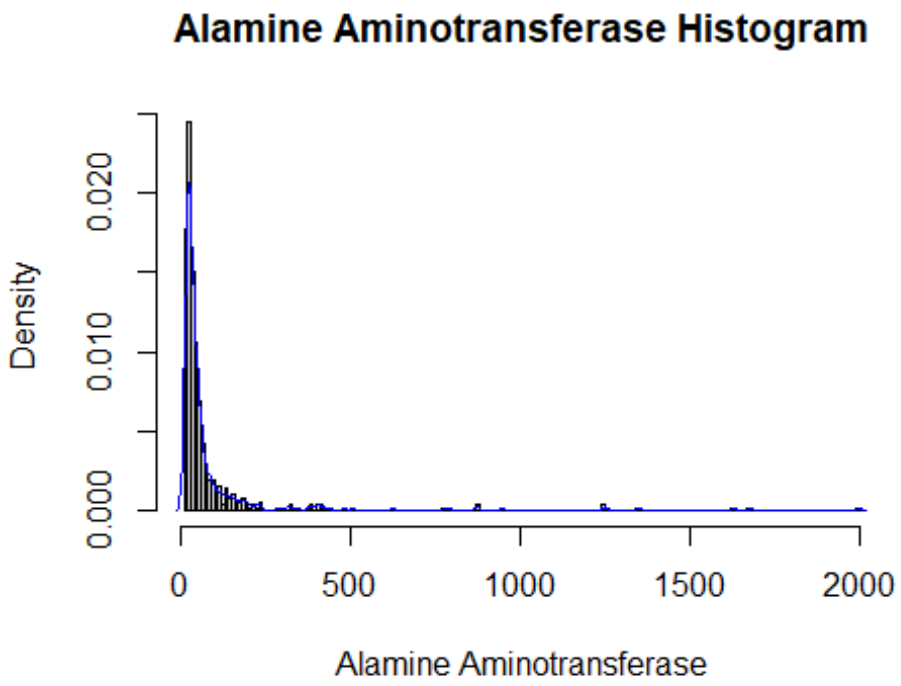
```
boxplot(liver$Alamine_Aminotransferase, main = "Alamine Aminotransferase
BoxPlot")
points(mean(liver$Alamine_Aminotransferase))
points(mode)
```



Box plots are graphical displays of the five number summary, So they describe the Minimum (lowest mean relative) lower whisker, Quartile 1 (It separates the lowest 25% observation to the rest of the observations), Median (Which divides the lower 50% observation from the upper 50% observations), Quartile 3 (It divides the upper 25% observation to the rest of the observations) and Maximum. In the above Box plot diagram, there are many outliers and the upper whisker shows the highest mean relative. Now, I will

plot a histogram a graphical display of data using bars of different heights to measures distribution of continuous data as follows:

```
hist(liver$Alamine_Aminotransferase, prob = TRUE, breaks = 152, xlab =
"Alamine Aminotransferase", main = "Alamine Aminotransferase Histogram")
lines(density(liver$Alamine_Aminotransferase), col='blue')
```



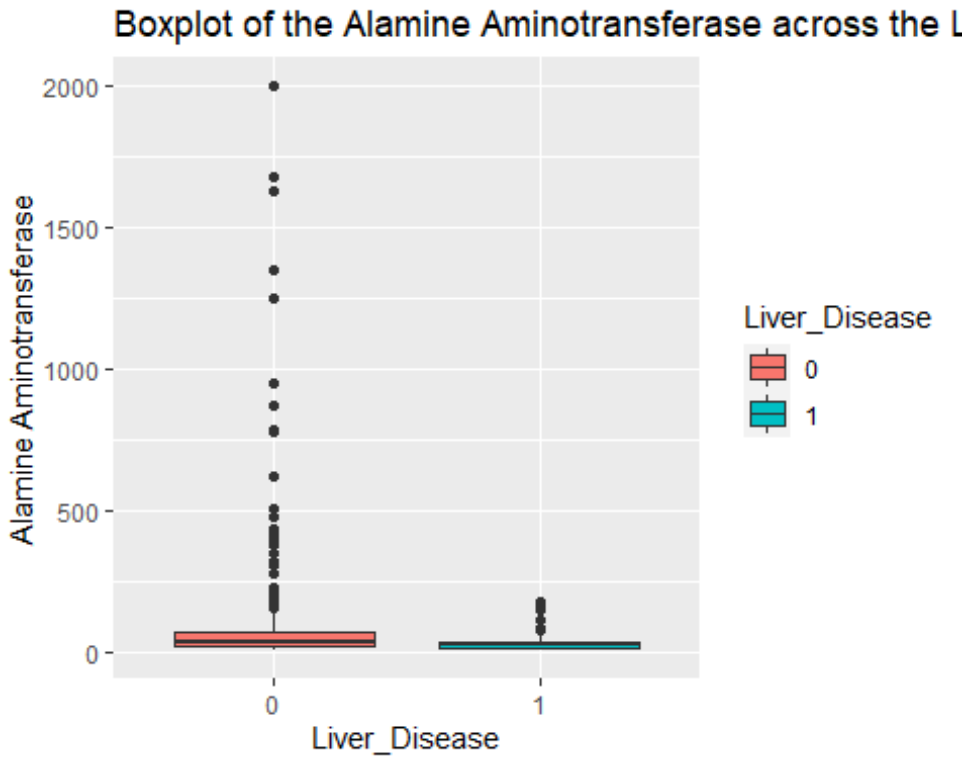
From the above histogram, we can observe that the diagram has many peaks.Using the density lines, I have approximated the histogram graph. Density provides information about the type of distribution under consideration and allows us to determine whether the attribute is distributed normally or not. From the graph above we can see how the `Alamine_Aminotransferase` variable does not seem to fit to a Normal distribution, there is a peak of the frequency distributions around many values and also a right skewed. We have already observed a positive value of `6.510652`, so we will affirm that the distribution is right skewed. To check whether the distribution is `Platykurtic` or not, we will check this by following R method as follows:

```
kurtosis(liver$Alamine_Aminotransferase)

## [1] 52.79182
```

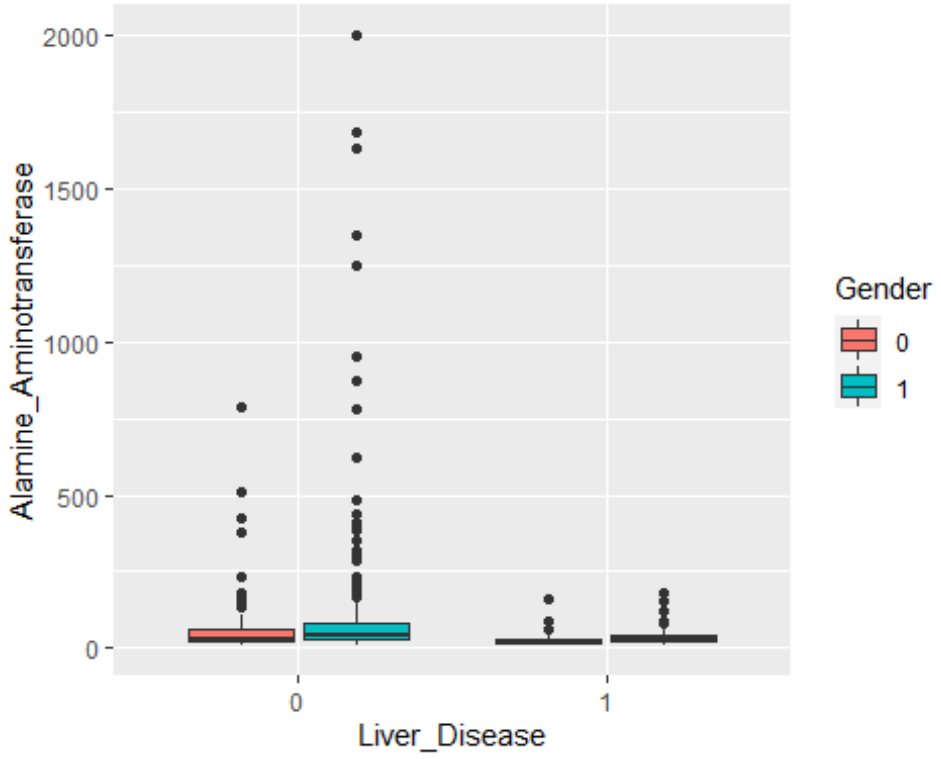
The distribution is also `Leptokurtic`, since the value is less than 3.

```
ggplot(liver, aes(x = Liver_Disease, y = Alamine_Aminotransferase, fill =
Liver_Disease)) +
  geom_boxplot() +
```

```
  ylab("Alamine Aminotransferase") +
  ggtitle("Boxplot of the Alamine Aminotransferase across the Liver_Disease")
```




```
ggplot(liver, aes(Liver_Disease, Alamine_Aminotransferase)) +
  geom_boxplot(aes(fill = Gender))
```



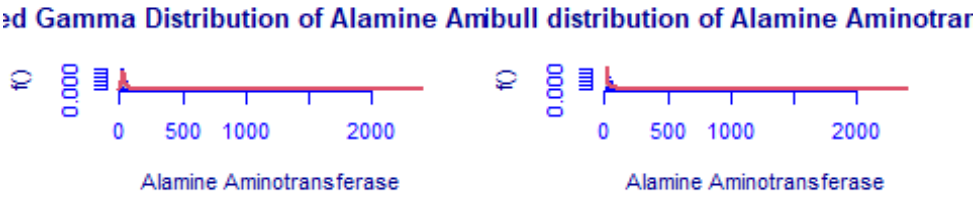
Another skewed variable. Range is from 10 to 2000, with a mean of 81.12, median of 35.00. There is a wider range within the Liver_Disease = 0 group than the Liver_Disease = 1 group, and the means are clearly different between the two groups. We again see a difference between the genders within the Liver_Disease. The male groups within each group have a higher mean of Alamine Aminotransferase.

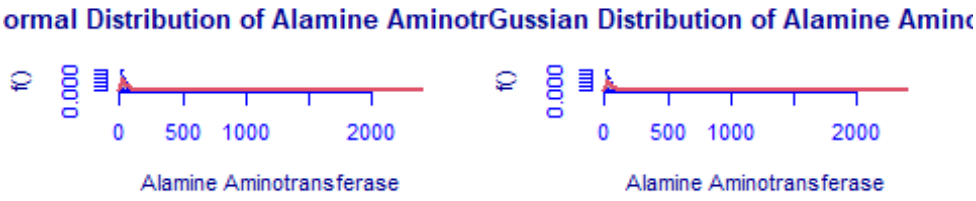
Alamine Aminotransferase Fit for the Data

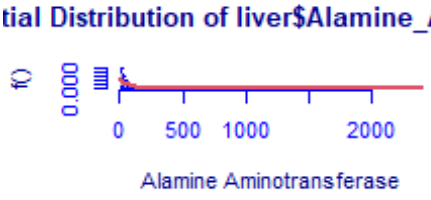
Now we will try to fit different models to Alamine Aminotransferase distribution evaluating the Akaike Information Criterion (AIC) and the Bayesian Information criterion (BIC), The lower are the indexes, the better is the fitting. I will evaluate both the single parameter distributions and mixture distributions as follows:

```
library(MASS)
library(gamlss)


par(mfrow=c(3,2))

aa.GG <- histDist(liver$Alamine_Aminotransferase, family=GG, nbins = 152,
xlab = "Alamine Aminotransferase", main="Generalized Gamma Distribution of
Alamine Aminotransferase")

aa.WEI <- histDist(liver$Alamine_Aminotransferase, family=WEI, nbins = 152,
xlab = "Alamine Aminotransferase", main="Weibull distribution of Alamine
Aminotransferase")
```

```
aa.LOGNO <- histDist(liver$Alamine_Aminotransferase, family=LOGNO, nbins =
152, xlab = "Alamine Aminotransferase", main="Log-Normal Distribution of
Alamine Aminotransferase")

aa.IG <- histDist(liver$Alamine_Aminotransferase, family=IG, nbins=152, xlab
= "Alamine Aminotransferase", main = "Inverse Gussian Distribution of Alamine
Aminotransferase")

aa.EXP<- histDist(liver$Alamine_Aminotransferase, family=EXP, nbins=152, xlab
= "Alamine Aminotransferase", main = "Exponential Distribution of
liver$Alamine_Aminotransferase")
```








```
library(Matrix)
library(glmnet)
df <- data.frame(Rownames = c("Generalized Gamma", "Weibull", "Log-Normal",
                              "Inverse Gussian", "Exponential"),
                 AIC = c(AIC(aa.GG), AIC(aa.WEI), AIC(aa.LOGNO),
                         AIC(aa.IG), AIC(aa.EXP)),
                 BIC = c(aa.GG$sbc, aa.WEI$sbc, aa.LOGNO$sbc,
                         aa.IG$sbc, aa.EXP$sbc),
                 df = c(aa.GG$df.fit, aa.WEI$df.fit,
                        aa.LOGNO$df.fit, aa.IG$df.fit, aa.EXP$df.fit),
                 LogLike = c(logLik(aa.GG), logLik(aa.WEI),
                             logLik(aa.LOGNO), logLik(aa.IG),
```

```
logLik(aa.EXP)))
df

##             Rownames      AIC      BIC df   LogLike
## 1 Generalized Gamma 5697.111 5710.195  3 -2845.555
## 2           Weibull 6202.697 6211.420  2 -3099.349
## 3        Log-Normal 5875.799 5884.522  2 -2935.900
## 4   Inverse Gussian 5872.108 5880.830  2 -2934.054
## 5       Exponential 6250.573 6254.934  1 -3124.287
```

As we can see, the model with the highest log likelihood (-2845.555) and the lowest AIC (5697.111) and BIC (5710.195) is the Generalized Gamma Distribution. Therefore, based on the greatest likelihood technique, our data fits better in the Generalized Gamma Distribution. Now, I will also compare two models by means of the Likelihood ratio test as we performed above as follows:

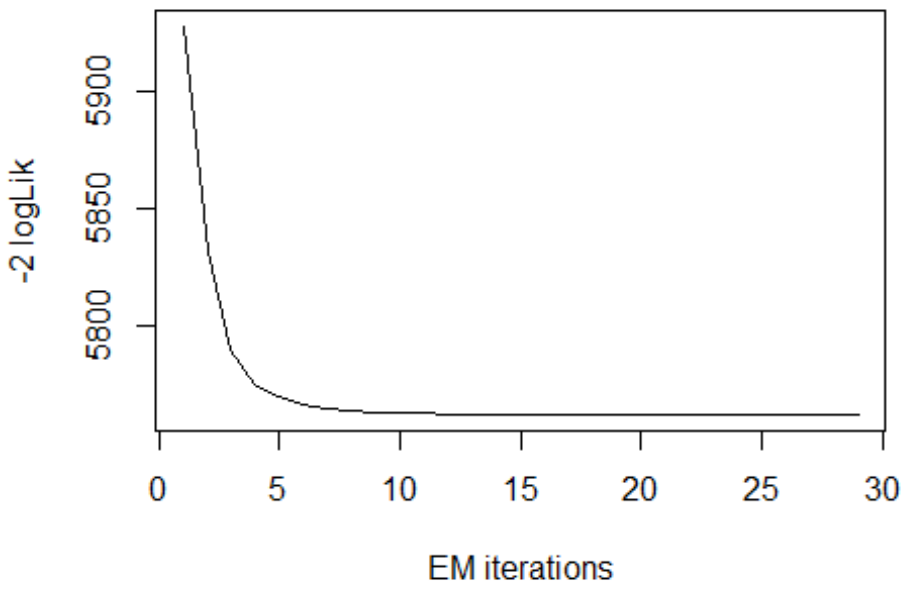
```
library(zoo)
library(lmtest)
lrtest(aa.GG, aa.EXP)

## Likelihood ratio test
##
## Model 1: gamlssML(formula = liver$Alamine_Aminotransferase, family = "GG")
## Model 2: gamlssML(formula = liver$Alamine_Aminotransferase, family =
"EXP")
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   3 -2845.6
## 2   1 -3124.3 -2 557.46  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Under the null hypothesis, we compare the Generalized Gamma distribution to the Exponential distribution under the alternative hypothesis. At any level of significance, we can reject the null hypothesis. Because our distributions fit better in the Generalized Gamma model, we'll use it.

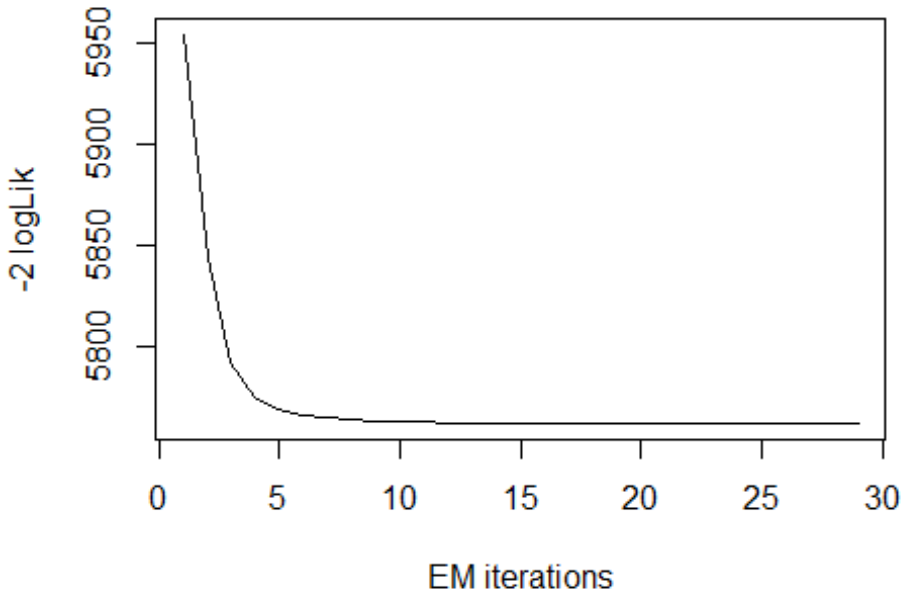
Distributions Mixture

Now, I will also apply the fitting criteria for the distributions with Gamma mixture as follows:

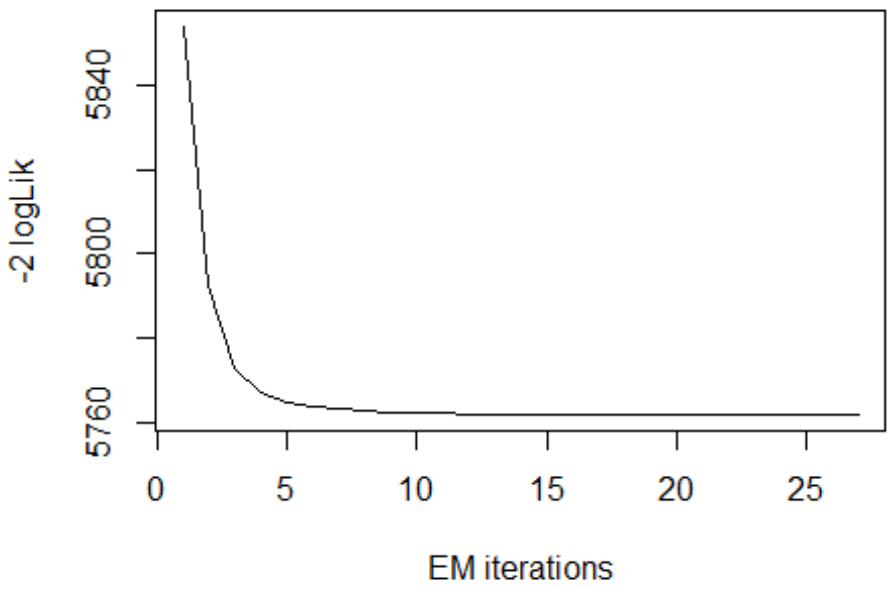
```
library(gamlss.mx)
mix.gam <- gamlssMXfits(n = 5, liver$Alamine_Aminotransferase~1, family = GA,
K = 2, data = liver)
```



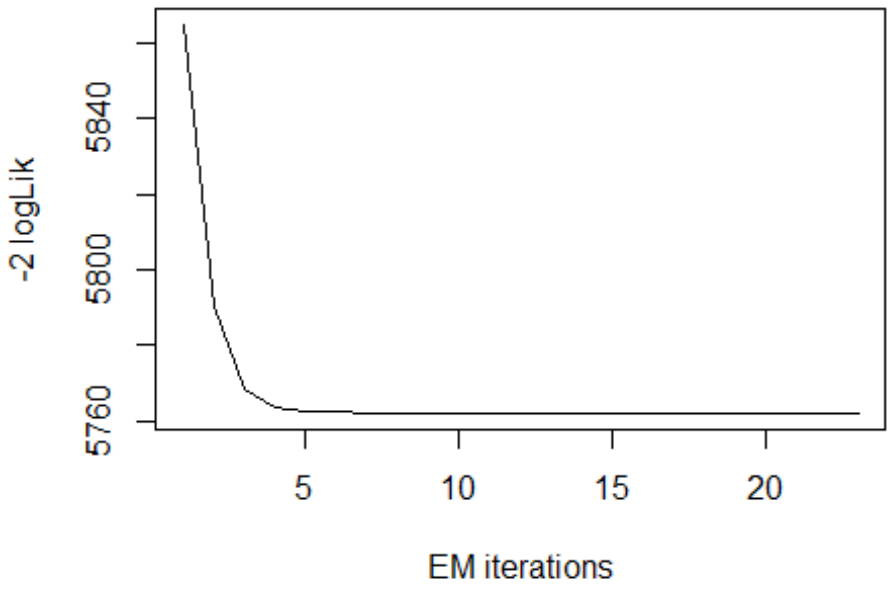

```
## model= 1
```



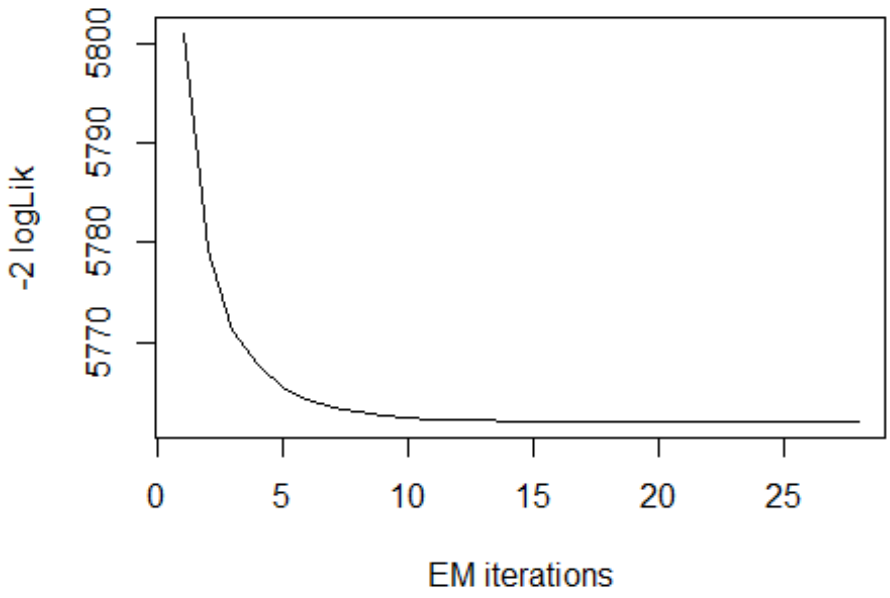

```
## model= 2
```




```
## model= 3
```




```
## model= 4
```




```
## model= 5

print(mix.gam)

##
## Mixing Family:  c("GA", "GA")
##
## Fitting method: EM algorithm
##
## Call:  gamlssMX(formula = liver$Alamine_Aminotransferase ~
##     1, family = GA, K = 2, data = liver)
##
## Mu Coefficients for model: 1
## (Intercept)
##       5.521
## Sigma Coefficients for model: 1
## (Intercept)
##    0.005752
## Mu Coefficients for model: 2
## (Intercept)
##       3.559
## Sigma Coefficients for model: 2
## (Intercept)
##     -0.7189
##
## Estimated probabilities: 0.2141273 0.7858727
##
```

```
## Degrees of Freedom for the fit: 5 Residual Deg. of Freedom   574
## Global Deviance:     5761.99
##             AIC:     5771.99
##             SBC:     5793.8
```

We can observe that the AIC value of the mixture of Gamma has improved, since it is higher than that of the single Gamma distribution. The current `AIC` value is `5771.99`, whereas the previous value was `5697.111` and the current `BIC` value is `5793.8`, whereas the previous value was `5710.195`.

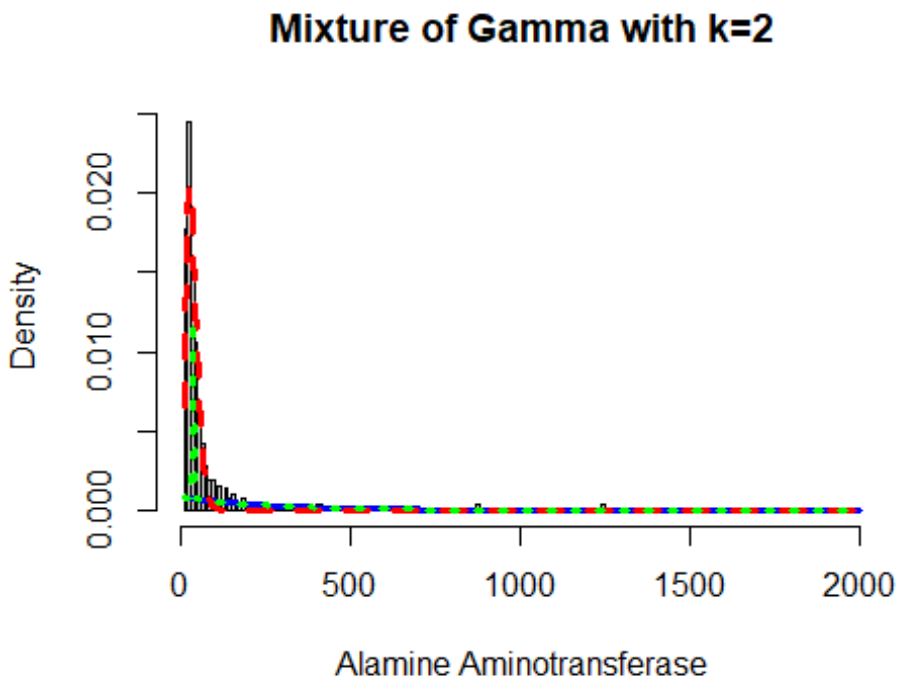
```
logLik(mix.gam)

## 'log Lik.' -2880.997 (df=5)

mix.gam$prob

## [1] 0.2141273 0.7858727

fitted(mix.gam, "mu")[1]

## [1] 81.11484

fitted(mix.gam, "sigma")[2]

## [1] 81.11484

hist(liver$Alamine_Aminotransferase, breaks = 152, xlab = "Alamine
Aminotransferase", main="Mixture of Gamma with k=2", freq = FALSE)

mu.hat1 <- exp(mix.gam[["models"]][[1]][["mu.coefficients"]])

sigma.hat1 <- exp(mix.gam[["models"]][[1]][["sigma.coefficients"]])

mu.hat2 <- exp(mix.gam[["models"]][[2]][["mu.coefficients"]])

sigma.hat2 <- exp(mix.gam[["models"]][[2]][["sigma.coefficients"]])

hist(liver$Alamine_Aminotransferase, breaks = 152, freq = FALSE, plot =
FALSE)

## Warning in hist.default(liver$Alamine_Aminotransferase, breaks = 152, freq
=
## FALSE, : argument 'freq' is not made use of

## $breaks
##   [1]   10   20   30   40   50   60   70   80   90  100  110  120  130
140  150
##  [16]  160  170  180  190  200  210  220  230  240  250  260  270  280
290  300
##  [31]  310  320  330  340  350  360  370  380  390  400  410  420  430
440  450
##  [46]  460  470  480  490  500  510  520  530  540  550  560  570  580
```

```
590  600
##  [61]  610  620  630  640  650  660  670  680  690  700  710  720  730
740  750
##  [76]  760  770  780  790  800  810  820  830  840  850  860  870  880
890  900
##  [91]  910  920  930  940  950  960  970  980  990 1000 1010 1020 1030
1040 1050
## [106] 1060 1070 1080 1090 1100 1110 1120 1130 1140 1150 1160 1170 1180
1190 1200
## [121] 1210 1220 1230 1240 1250 1260 1270 1280 1290 1300 1310 1320 1330
1340 1350
## [136] 1360 1370 1380 1390 1400 1410 1420 1430 1440 1450 1460 1470 1480
1490 1500
## [151] 1510 1520 1530 1540 1550 1560 1570 1580 1590 1600 1610 1620 1630
1640 1650
## [166] 1660 1670 1680 1690 1700 1710 1720 1730 1740 1750 1760 1770 1780
1790 1800
## [181] 1810 1820 1830 1840 1850 1860 1870 1880 1890 1900 1910 1920 1930
1940 1950
## [196] 1960 1970 1980 1990 2000
##
## $counts
##   [1] 103 142  87  61  40  24  16  11  11   7   9   2   8   4   6   3   3
4
##  [19]   3   1   2   1   3   0   0   0   0   1   0   1   0   2   0   1   0
0
##  [37]   1   2   0   2   2   1   1   0   0   0   0   1   0   1   0   0   0
0
##  [55]   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
0
##  [73]   0   0   0   0   1   1   0   0   0   0   0   0   0   0   2   0   0
0
##  [91]   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [109]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0
0
## [127]   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
0
## [145]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
1
## [163]   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
0
## [181]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [199]   1
##
## $density
##   [1] 0.0177892919 0.0245250432 0.0150259067 0.0105354059 0.0069084629
##   [6] 0.0041450777 0.0027633851 0.0018998273 0.0018998273 0.0012089810
##  [11] 0.0015544041 0.0003454231 0.0013816926 0.0006908463 0.0010362694
```

```
##  [16] 0.0005181347 0.0005181347 0.0006908463 0.0005181347 0.0001727116
##  [21] 0.0003454231 0.0001727116 0.0005181347 0.0000000000 0.0000000000
##  [26] 0.0000000000 0.0000000000 0.0001727116 0.0000000000 0.0001727116
##  [31] 0.0000000000 0.0003454231 0.0000000000 0.0001727116 0.0000000000
##  [36] 0.0000000000 0.0001727116 0.0003454231 0.0000000000 0.0003454231
##  [41] 0.0003454231 0.0001727116 0.0001727116 0.0000000000 0.0000000000
##  [46] 0.0000000000 0.0000000000 0.0001727116 0.0000000000 0.0001727116
##  [51] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##  [56] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##  [61] 0.0000000000 0.0001727116 0.0000000000 0.0000000000 0.0000000000
##  [66] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##  [71] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##  [76] 0.0000000000 0.0001727116 0.0001727116 0.0000000000 0.0000000000
##  [81] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##  [86] 0.0000000000 0.0003454231 0.0000000000 0.0000000000 0.0000000000
##  [91] 0.0000000000 0.0000000000 0.0000000000 0.0001727116 0.0000000000
##  [96] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [101] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [106] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [111] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [116] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [121] 0.0000000000 0.0000000000 0.0000000000 0.0003454231 0.0000000000
## [126] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [131] 0.0000000000 0.0000000000 0.0000000000 0.0001727116 0.0000000000
## [136] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [141] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [146] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [151] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [156] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [161] 0.0000000000 0.0001727116 0.0000000000 0.0000000000 0.0000000000
## [166] 0.0000000000 0.0001727116 0.0000000000 0.0000000000 0.0000000000
## [171] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [176] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [181] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [186] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [191] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [196] 0.0000000000 0.0000000000 0.0000000000 0.0001727116
##
## $mids
##   [1]   15   25   35   45   55   65   75   85   95  105  115  125  135
145  155
##  [16]  165  175  185  195  205  215  225  235  245  255  265  275  285
295  305
##  [31]  315  325  335  345  355  365  375  385  395  405  415  425  435
445  455
##  [46]  465  475  485  495  505  515  525  535  545  555  565  575  585
595  605
##  [61]  615  625  635  645  655  665  675  685  695  705  715  725  735
745  755
##  [76]  765  775  785  795  805  815  825  835  845  855  865  875  885
```

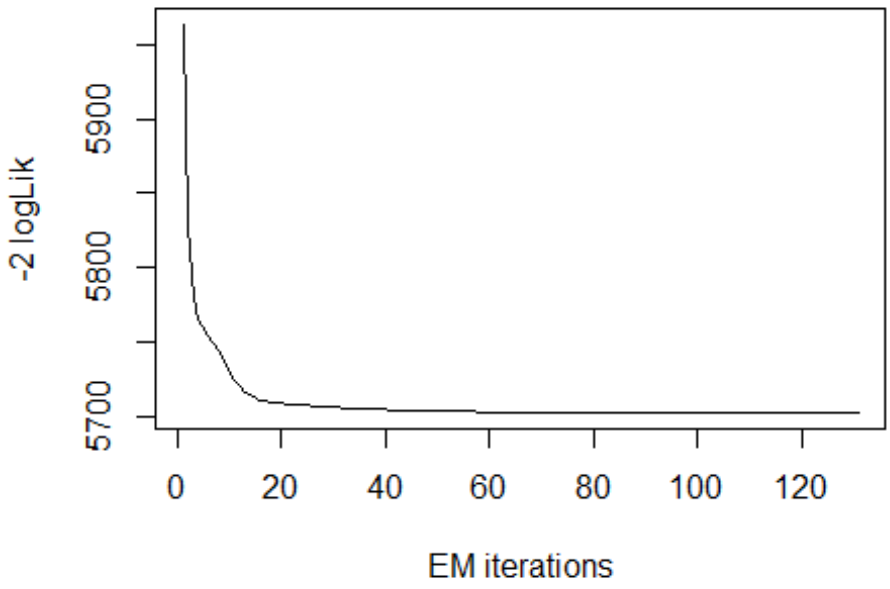
```
895  905
##  [91]  915  925  935  945  955  965  975  985  995 1005 1015 1025 1035
1045 1055
## [106] 1065 1075 1085 1095 1105 1115 1125 1135 1145 1155 1165 1175 1185
1195 1205
## [121] 1215 1225 1235 1245 1255 1265 1275 1285 1295 1305 1315 1325 1335
1345 1355
## [136] 1365 1375 1385 1395 1405 1415 1425 1435 1445 1455 1465 1475 1485
1495 1505
## [151] 1515 1525 1535 1545 1555 1565 1575 1585 1595 1605 1615 1625 1635
1645 1655
## [166] 1665 1675 1685 1695 1705 1715 1725 1735 1745 1755 1765 1775 1785
1795 1805
## [181] 1815 1825 1835 1845 1855 1865 1875 1885 1895 1905 1915 1925 1935
1945 1955
## [196] 1965 1975 1985 1995
##
## $xname
## [1] "liver$Alamine_Aminotransferase"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"

lines(seq(min(liver$Alamine_Aminotransferase),
max(liver$Alamine_Aminotransferase),
      length = length(liver$Alamine_Aminotransferase)),
      mix.gam[["prob"]][1]*dGA(seq(min(liver$Alamine_Aminotransferase),
      max(liver$Alamine_Aminotransferase), length
=length(liver$Alamine_Aminotransferase)),
      mu = mu.hat1, sigma = sigma.hat1), lty=1, lwd=3, col="blue")

lines(seq(min(liver$Alamine_Aminotransferase),
max(liver$Alamine_Aminotransferase),
      length = length(liver$Alamine_Aminotransferase)),
      mix.gam[["prob"]][2]*dGA(seq(min(liver$Alamine_Aminotransferase),
      max(liver$Alamine_Aminotransferase), length
=length(liver$Alamine_Aminotransferase)),
      mu = mu.hat2, sigma = sigma.hat2), lty = 2, lwd = 3, col = "red")

lines(seq(min(liver$Alamine_Aminotransferase),
max(liver$Alamine_Aminotransferase),
      length = length(liver$Alamine_Aminotransferase)),
      mix.gam[["prob"]][1]*dGA(seq(min(liver$Alamine_Aminotransferase),
      max(liver$Alamine_Aminotransferase),
      length = length(liver$Alamine_Aminotransferase)), mu = mu.hat1, sigma =
sigma.hat1) +
      mix.gam[["prob"]][2]*dRG(seq(min(liver$Alamine_Aminotransferase),
```

```
      max(liver$Alamine_Aminotransferase),
      length = length(liver$Alamine_Aminotransferase)), mu = mu.hat2, sigma =
sigma.hat2),
      lty = 3, lwd = 3, col = "green")
```

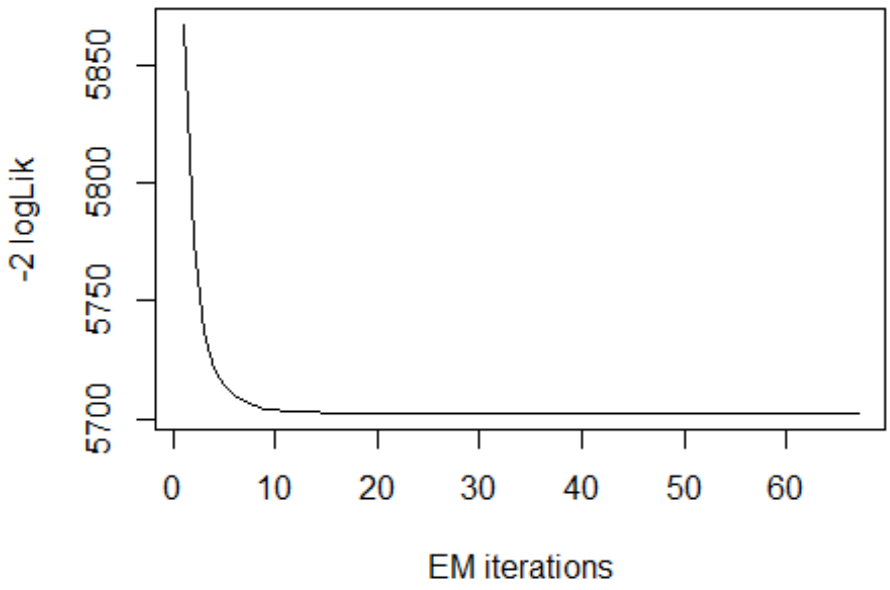


Now, I will again apply the fitting criteria for the distributions with Gamma mixture with the k = 3 as follows:

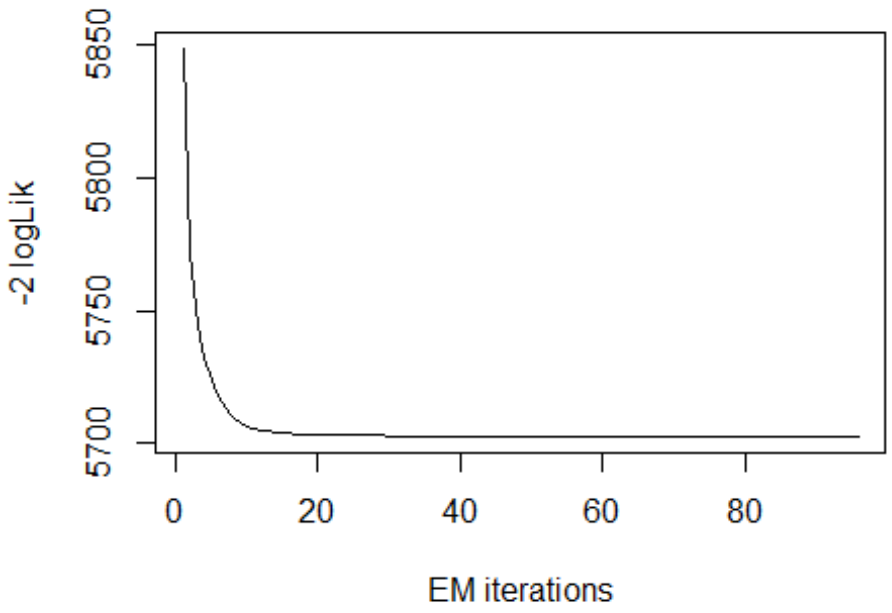
```
mix.gam.3 <- gamlssMXfits(n = 5, liver$Alamine_Aminotransferase~1, family =
GA, K = 3, data = liver)
```



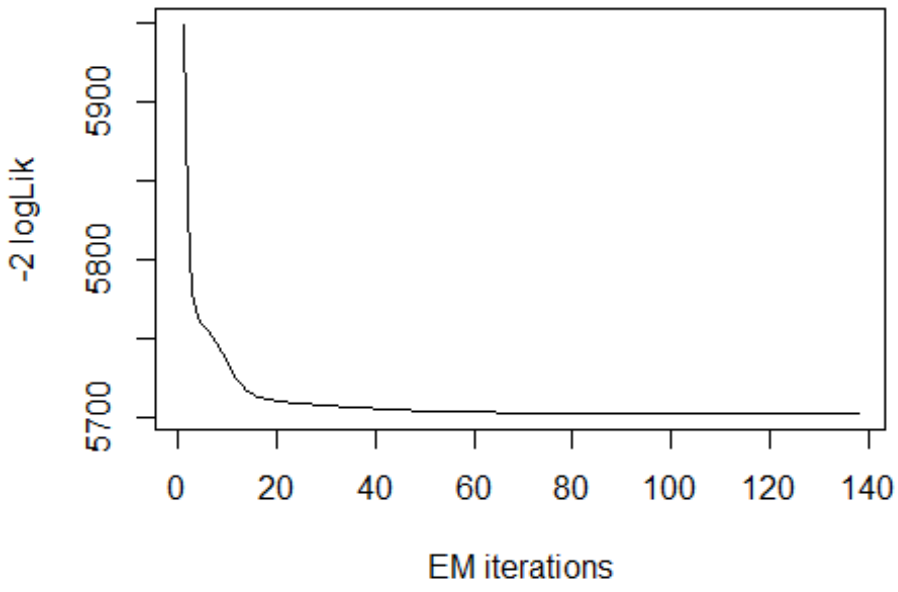

```
## model= 1
```



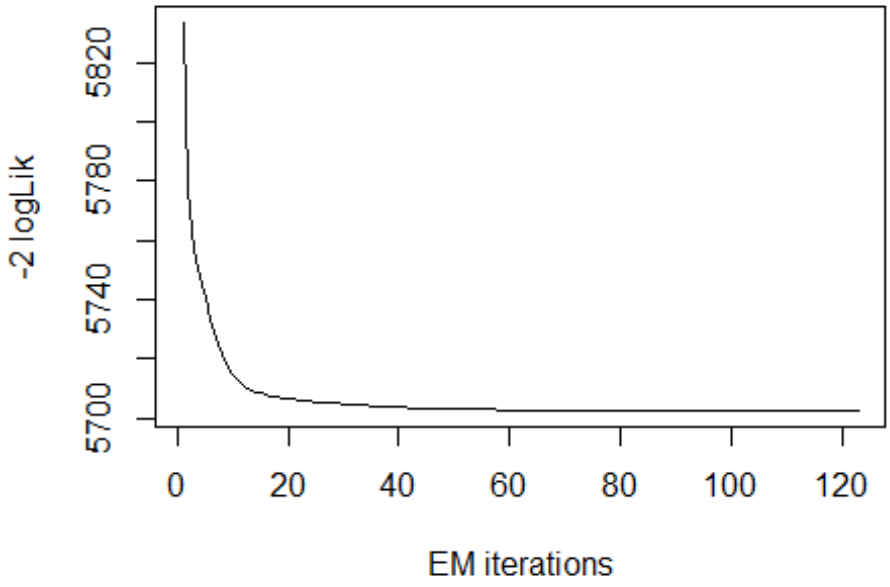

```
## model= 2
```




```
## model= 3
```




```
## model= 4
```




```
## model= 5

print(mix.gam.3)

##
## Mixing Family:  c("GA", "GA", "GA")
##
## Fitting method: EM algorithm
##
## Call:  gamlssMX(formula = liver$Alamine_Aminotransferase ~
##     1, family = GA, K = 3, data = liver)
##
## Mu Coefficients for model: 1
## (Intercept)
##       3.393
## Sigma Coefficients for model: 1
## (Intercept)
##     -0.8918
## Mu Coefficients for model: 2
## (Intercept)
##       6.256
## Sigma Coefficients for model: 2
## (Intercept)
##     -0.1241
## Mu Coefficients for model: 3
## (Intercept)
##       4.432
```

```
## Sigma Coefficients for model: 3
## (Intercept)
##     -0.5418
##
## Estimated probabilities: 0.6458668 0.07345744 0.2806758
##
## Degrees of Freedom for the fit: 8 Residual Deg. of Freedom   571
## Global Deviance:     5702.46
##             AIC:     5718.46
##             SBC:     5753.35
```

```
logLik(mix.gam.3)
```

```
## 'log Lik.' -2851.231 (df=8)
```

```
mix.gam.3$prob
```

```
## [1] 0.64586680 0.07345744 0.28067576
```

```
fitted(mix.gam.3, "mu")[1]
```

```
## [1] 81.11845
```

```
fitted(mix.gam.3, "sigma")[2]
```

```
## [1] 81.11845
```

```
hist(liver$Alamine_Aminotransferase, breaks = 152, xlab = "Alamine
Aminotransferase", main="Mixture of Gamma with k=3", freq = FALSE)

mu.hat1 <- exp(mix.gam.3[["models"]][[1]][["mu.coefficients"]])

sigma.hat1 <- exp(mix.gam.3[["models"]][[1]][["sigma.coefficients"]])

mu.hat2 <- exp(mix.gam.3[["models"]][[2]][["mu.coefficients"]])

sigma.hat2 <- exp(mix.gam.3[["models"]][[2]][["sigma.coefficients"]])

hist(liver$Alamine_Aminotransferase, breaks = 152, freq = FALSE, plot =
FALSE)
```

```
## Warning in hist.default(liver$Alamine_Aminotransferase, breaks = 152, freq
=
## FALSE, : argument 'freq' is not made use of

## $breaks
##   [1]   10   20   30   40   50   60   70   80   90  100  110  120  130
140  150
##  [16]  160  170  180  190  200  210  220  230  240  250  260  270  280
290  300
##  [31]  310  320  330  340  350  360  370  380  390  400  410  420  430
440  450
```

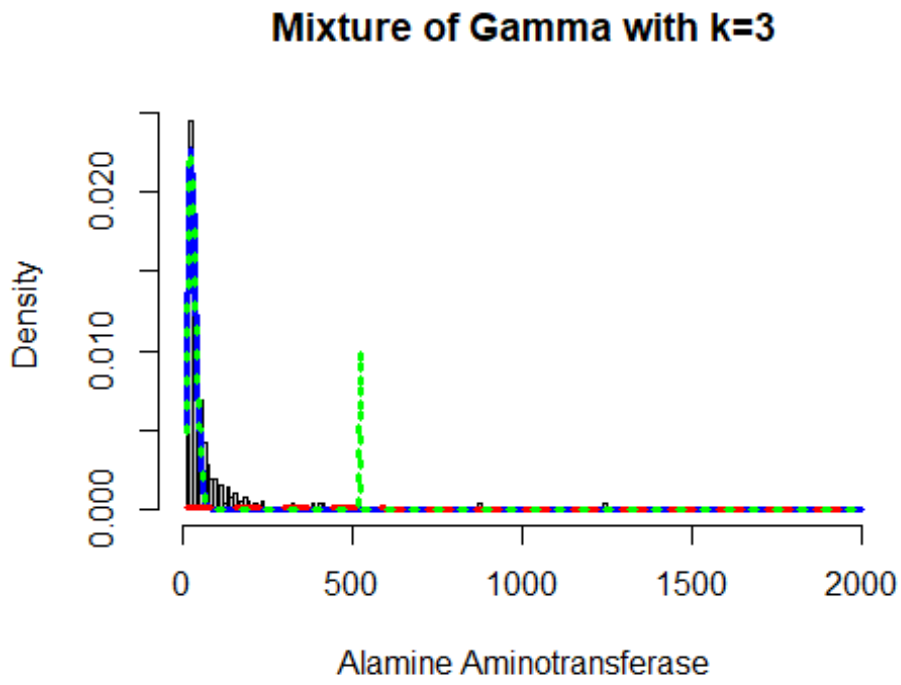
```
##  [46]  460  470  480  490  500  510  520  530  540  550  560  570  580
590  600
##  [61]  610  620  630  640  650  660  670  680  690  700  710  720  730
740  750
##  [76]  760  770  780  790  800  810  820  830  840  850  860  870  880
890  900
##  [91]  910  920  930  940  950  960  970  980  990 1000 1010 1020 1030
1040 1050
## [106] 1060 1070 1080 1090 1100 1110 1120 1130 1140 1150 1160 1170 1180
1190 1200
## [121] 1210 1220 1230 1240 1250 1260 1270 1280 1290 1300 1310 1320 1330
1340 1350
## [136] 1360 1370 1380 1390 1400 1410 1420 1430 1440 1450 1460 1470 1480
1490 1500
## [151] 1510 1520 1530 1540 1550 1560 1570 1580 1590 1600 1610 1620 1630
1640 1650
## [166] 1660 1670 1680 1690 1700 1710 1720 1730 1740 1750 1760 1770 1780
1790 1800
## [181] 1810 1820 1830 1840 1850 1860 1870 1880 1890 1900 1910 1920 1930
1940 1950
## [196] 1960 1970 1980 1990 2000
##
## $counts
##   [1] 103 142  87  61  40  24  16  11  11   7   9   2   8   4   6   3   3
4
##  [19]   3   1   2   1   3   0   0   0   0   1   0   1   0   2   0   1   0
0
##  [37]   1   2   0   2   2   1   1   0   0   0   0   1   0   1   0   0   0
0
##  [55]   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
0
##  [73]   0   0   0   0   1   1   0   0   0   0   0   0   0   0   2   0   0
0
##  [91]   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [109]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0
0
## [127]   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
0
## [145]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
1
## [163]   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
0
## [181]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [199]   1
##
## $density
##   [1] 0.0177892919 0.0245250432 0.0150259067 0.0105354059 0.0069084629
##   [6] 0.0041450777 0.0027633851 0.0018998273 0.0018998273 0.0012089810
```

```
##  [11] 0.0015544041 0.0003454231 0.0013816926 0.0006908463 0.0010362694
##  [16] 0.0005181347 0.0005181347 0.0006908463 0.0005181347 0.0001727116
##  [21] 0.0003454231 0.0001727116 0.0005181347 0.0000000000 0.0000000000
##  [26] 0.0000000000 0.0000000000 0.0001727116 0.0000000000 0.0001727116
##  [31] 0.0000000000 0.0003454231 0.0000000000 0.0001727116 0.0000000000
##  [36] 0.0000000000 0.0001727116 0.0003454231 0.0000000000 0.0003454231
##  [41] 0.0003454231 0.0001727116 0.0001727116 0.0000000000 0.0000000000
##  [46] 0.0000000000 0.0000000000 0.0001727116 0.0000000000 0.0001727116
##  [51] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##  [56] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##  [61] 0.0000000000 0.0001727116 0.0000000000 0.0000000000 0.0000000000
##  [66] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##  [71] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##  [76] 0.0000000000 0.0001727116 0.0001727116 0.0000000000 0.0000000000
##  [81] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##  [86] 0.0000000000 0.0003454231 0.0000000000 0.0000000000 0.0000000000
##  [91] 0.0000000000 0.0000000000 0.0000000000 0.0001727116 0.0000000000
##  [96] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [101] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [106] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [111] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [116] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [121] 0.0000000000 0.0000000000 0.0000000000 0.0003454231 0.0000000000
## [126] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [131] 0.0000000000 0.0000000000 0.0000000000 0.0001727116 0.0000000000
## [136] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [141] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [146] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [151] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [156] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [161] 0.0000000000 0.0001727116 0.0000000000 0.0000000000 0.0000000000
## [166] 0.0000000000 0.0001727116 0.0000000000 0.0000000000 0.0000000000
## [171] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [176] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [181] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [186] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [191] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [196] 0.0000000000 0.0000000000 0.0000000000 0.0001727116
##
## $mids
##   [1]   15   25   35   45   55   65   75   85   95  105  115  125  135
145  155
##  [16]  165  175  185  195  205  215  225  235  245  255  265  275  285
295  305
##  [31]  315  325  335  345  355  365  375  385  395  405  415  425  435
445  455
##  [46]  465  475  485  495  505  515  525  535  545  555  565  575  585
595  605
##  [61]  615  625  635  645  655  665  675  685  695  705  715  725  735
745  755
```

```
##  [76]  765  775  785  795  805  815  825  835  845  855  865  875  885
895  905
##  [91]  915  925  935  945  955  965  975  985  995 1005 1015 1025 1035
1045 1055
## [106] 1065 1075 1085 1095 1105 1115 1125 1135 1145 1155 1165 1175 1185
1195 1205
## [121] 1215 1225 1235 1245 1255 1265 1275 1285 1295 1305 1315 1325 1335
1345 1355
## [136] 1365 1375 1385 1395 1405 1415 1425 1435 1445 1455 1465 1475 1485
1495 1505
## [151] 1515 1525 1535 1545 1555 1565 1575 1585 1595 1605 1615 1625 1635
1645 1655
## [166] 1665 1675 1685 1695 1705 1715 1725 1735 1745 1755 1765 1775 1785
1795 1805
## [181] 1815 1825 1835 1845 1855 1865 1875 1885 1895 1905 1915 1925 1935
1945 1955
## [196] 1965 1975 1985 1995
##
## $xname
## [1] "liver$Alamine_Aminotransferase"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```
lines(seq(min(liver$Alamine_Aminotransferase),
max(liver$Alamine_Aminotransferase),
      length = length(liver$Alamine_Aminotransferase)),
      mix.gam.3[["prob"]][1]*dGA(seq(min(liver$Alamine_Aminotransferase),
      max(liver$Alamine_Aminotransferase),
      length = length(liver$Alamine_Aminotransferase)), mu = mu.hat1, sigma =
sigma.hat1),
      lty=1, lwd=3, col="blue")

lines(seq(min(liver$Alamine_Aminotransferase),
max(liver$Alamine_Aminotransferase),
      length = length(liver$Alamine_Aminotransferase)),
      mix.gam.3[["prob"]][2]*dGA(seq(min(liver$Alamine_Aminotransferase),
      max(liver$Alamine_Aminotransferase),
      length = length(liver$Alamine_Aminotransferase)), mu = mu.hat2, sigma =
sigma.hat2),
      lty = 2, lwd = 3, col = "red")

lines(seq(min(liver$Alamine_Aminotransferase),
max(liver$Alamine_Aminotransferase),
      length = length(liver$Alamine_Aminotransferase)),
      mix.gam.3[["prob"]][1]*dGA(seq(min(liver$Alamine_Aminotransferase),
      max(liver$Alamine_Aminotransferase),
```

```
      length = length(liver$Alamine_Aminotransferase)),
      mu = mu.hat1, sigma = sigma.hat1)+
      mix.gam.3[["prob"]][2]*dRG(seq(min(liver$Alamine_Aminotransferase),
      max(liver$Alamine_Aminotransferase),
      length = length(liver$Alamine_Aminotransferase)), mu = mu.hat2, sigma =
sigma.hat2),
      lty = 3, lwd = 3, col = "green")
```




```
mix.gm.tb <- data.frame(row.names=c('Gamma Mixture, K=3','Gamma Mixture,
K=2', "Generalized Gamma"),
                        AIC=c(mix.gam.3$aic, mix.gam$aic, AIC(aa.GG)),
                        BIC=c(mix.gam.3$sbc, mix.gam$sbc, aa.GG$sbc))
mix.gm.tb

##                           AIC      BIC
## Gamma Mixture, K=3 5718.463 5753.353
## Gamma Mixture, K=2 5771.995 5793.801
## Generalized Gamma  5697.111 5710.195
```

We can observe that the AIC value of the mixture of Gamma with k=3 and k=2 has increased, since values are higher than that of the single Gamma distribution. The previous AIC value is `5697.111`, whereas the current value which is higher is `5718.463` and the previous BIC value was `5710.195`, whereas the current value which is higher is `5753.353`. Here we can clearly see that our data fits better in the single Gamma Distribution.

Aspartate Aminotransferase

Lets explore the `Aspartate Aminotransferase` variable:

```
head(liver$Aspartate_Aminotransferase)
```

```
## [1]  18 100  68  20  59  14
```

```
length(liver$Aspartate_Aminotransferase)
```

```
## [1] 579
```

```
table(liver$Aspartate_Aminotransferase)
```

```
##
##   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24
25
##    1    2    5    3    8   10    9    8    9   11   14   14   13   16   12
13
##   26   27   28   29   30   31   32   33   34   35   36   37   38   39   40
41
##    9    8   13   11   14    8   12    6   12    8    5    4    5    7   11
7
##   42   43   44   45   46   47   48   49   50   51   52   53   54   55   56
57
##    9    7    6    4    3    5    5    1    4    4    2    5    6    2    5
6
##   58   59   60   61   62   63   64   65   66   67   68   70   71   72   73
74
##    9    4    1    2    3    2    1    3    6    2    5    3    2    1    3
2
##   75   76   77   78   79   80   81   82   83   84   85   86   87   88   89
90
##    2    1    1    1    3    3    1    2    2    1    1    2    4    4    1
4
##   91   92   95   97   98   99  100  101  102  103  104  105  108  110  111
113
##    1    5    2    1    1    1    1    1    1    2    2    2    3    1    2
3
##  114  116  125  126  127  130  134  135  138  139  140  141  142  143  145
148
##    1    1    2    1    2    1    1    1    3    2    4    2    1    2    2
1
##  149  150  152  155  156  161  168  176  178  180  181  185  186  187  188
190
##    1    1    2    1    1    1    1    1    2    2    1    1    2    1    1
2
##  200  202  220  221  230  231  232  233  235  236  245  247  248  250  275
285
##    1    1    1    1    1    2    1    1    1    1    2    1    1    1    1
1
```

```
##  298  330  348  350  367  368  384  397  400  401  405  406  441  497  500
511
##    1    1    1    2    1    1    1    1    1    1    1    1    1    1    1
1
##  540  562  576  602  623  630  731  794  844  850  950  960 1050 1500 1600
2946
##    1    1    1    1    1    2    2    1    1    4    1    1    2    1    1
1
## 4929
##    1
```

The table() function in R returns the absolute frequencies for each patient-specified value in the data set.

```
unique(liver$Aspartate_Aminotransferase)
```

```
##   [1]   18  100   68   20   59   14   12   11   19   58   56   30   41
53  441
##  [16]   23  245   28   34   66   55   45  731  850   21  111   44   57
80   36
##  [31]   77   73   50  110   47  576   15  178   27  960  406  150   61
54   24
##  [46]   16   43   97   86   88   95   26   17  397   29   22  127   79
142  152
##  [61]   31  350  794  400  202  630  950  161  405   92   39   10  116
98  285
##  [76]   64  149 2946 1600 1050  275  113   84   25   40   83   65 4929
90  140
##  [91]  139   87   38   42  233  138   82   35   32  187   62   74   67
37  602
## [106]   63   99  103  145  247  114  104   51   60 1500  180  148   46
13   85
## [121]  231  156   89  298   48  130   75  500  105  250  232   33  143
176   70
## [136]   52   91  236  108  190   71  126  141  102   81  511   72  135
497  844
## [151]  368  188  248  401   76  221  235  185  230  540  181  155  200
186  623
## [166]  220   78  348  125  330  562  384  367  101  168  134   49
```

```
length(unique(liver$Aspartate_Aminotransferase))
```

```
## [1] 177
```

```
min(liver$Aspartate_Aminotransferase)
```

```
## [1] 10
```

```
max(liver$Aspartate_Aminotransferase)
```

```
## [1] 4929
```

Here the total observation of Aspartate Aminotransferase is 579 and is a continuous variable that range lies between 10-4929. Now, we will see the summary of the Aspartate_Aminotransferase variable, for further analysis as follows:

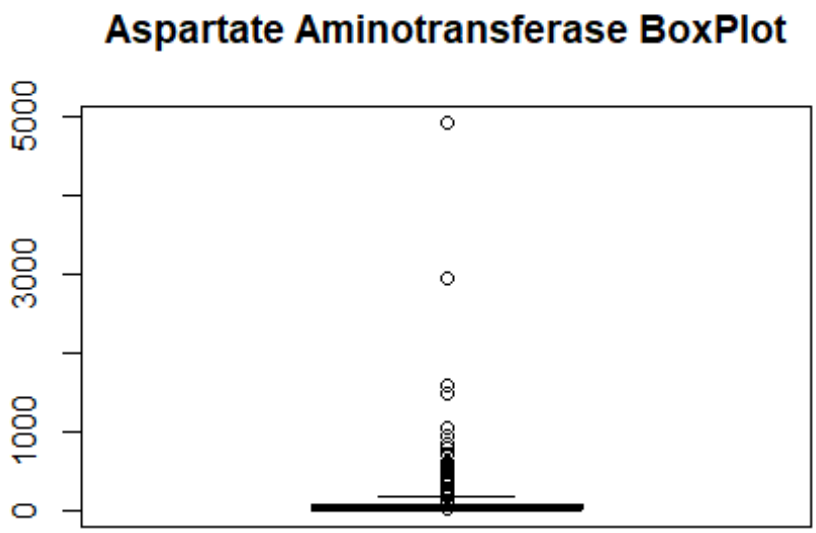
```
summary(liver$Aspartate_Aminotransferase)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    10.0    25.0    42.0   110.4    87.0  4929.0

sd(liver$Aspartate_Aminotransferase)

## [1] 289.85

var(liver$Aspartate_Aminotransferase)

## [1] 84013.04

v <- c(liver$Aspartate_Aminotransferase)
mode <- getMode(v)
print(mode)

## [1] 23
```

From the above summary we can observe that the Mean (110.4), Median (42.0) and Mode (23) are not equal so the distribution is asymmetrical. Here Mean > Median > Mode so the distribution could be positive and skewed to the right. Now we also confirm this as follows:

```
library(moments)
skewness(liver$Aspartate_Aminotransferase)

## [1] 10.485
```

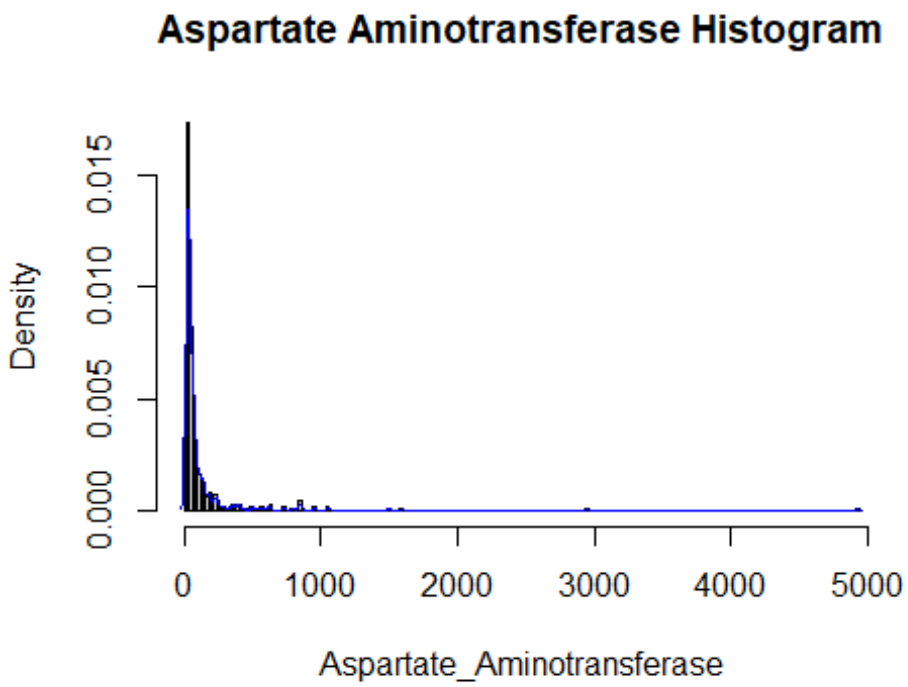
Hence, the skewness is the positive so the distribution is positively skewed was rightly observed. Also I will plot the Boxplot to comparing the distribution of data across data set as follows:

```
boxplot(liver$Aspartate_Aminotransferase, main = "Aspartate Aminotransferase
BoxPlot")
points(mean(liver$Aspartate_Aminotransferase))
points(mode)
```



Box plots are graphical displays of the five number summary, So they describe the Minimum (lowest mean relative) lower whisker, Quartile 1 (It separates the lowest 25% observation to the rest of the observations), Median (Which divides the lower 50% observation from the upper 50% observations), Quartile 3 (It divides the upper 25% observation to the rest of the observations) and Maximum. In the above Box plot diagram, there are many outliers and the upper whisker shows the highest mean relative. Now, I will plot a histogram a graphical display of data using bars of different heights to measures distribution of continuous data as follows:

```
hist(liver$Aspartate_Aminotransferase, prob = TRUE, breaks = 177, xlab =
"Aspartate_Aminotransferase", main = "Aspartate Aminotransferase Histogram")
lines(density(liver$Aspartate_Aminotransferase), col='blue')
```



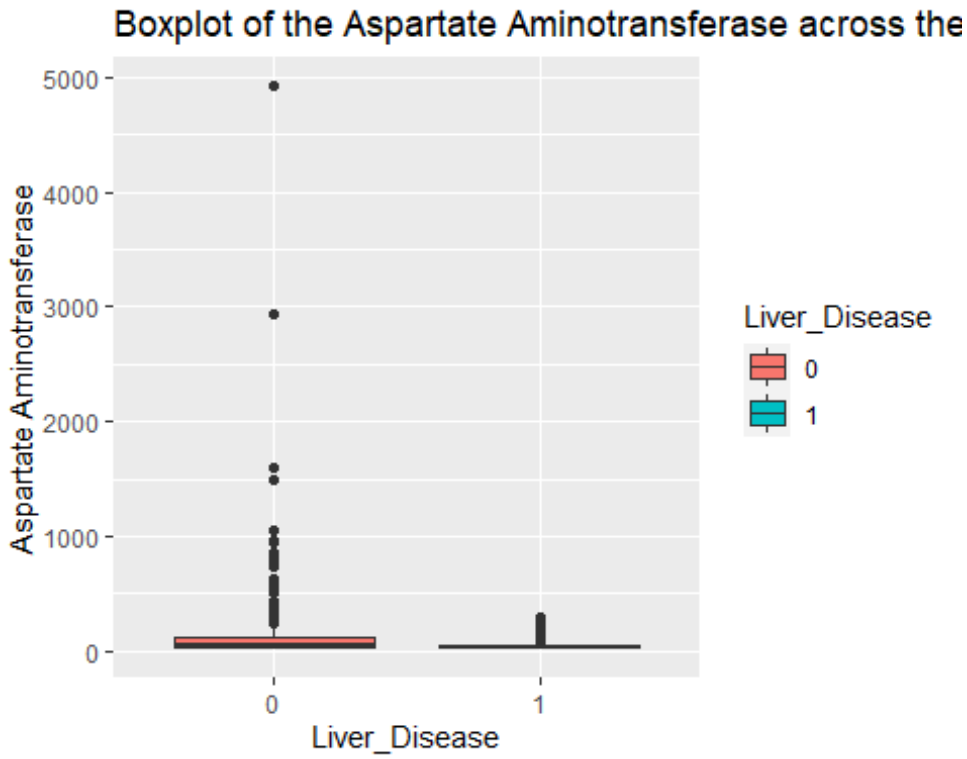
From the above histogram, we can observe that the diagram has many peaks.Using the density lines, I have approximated the histogram graph. Density provides information about the type of distribution under consideration and allows us to determine whether the attribute is distributed normally or not. From the graph above we can see how the `Aspartate_Aminotransferase` variable does not seem to fit to a Normal distribution, there is a peak of the frequency distributions around many values and also a right skewed. We have already observed a positive value of `10.485`, so we will affirm that the distribution is right skewed. To check whether the distribution is `Platykurtic` or not, we will check this by following R method as follows:

```
kurtosis(liver$Aspartate_Aminotransferase)
```

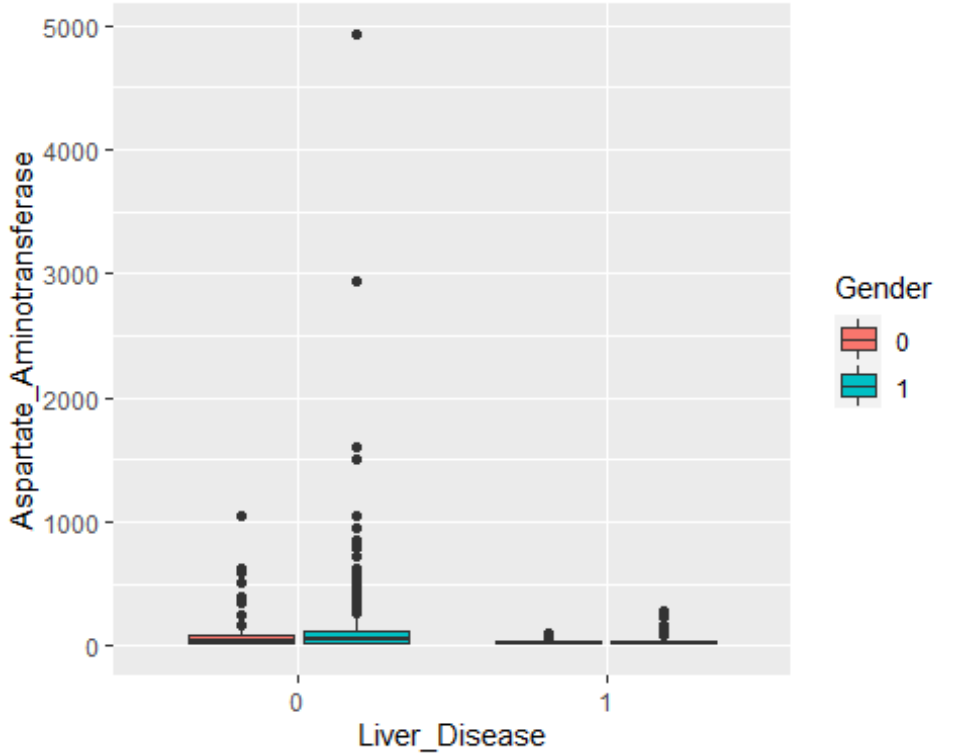
```
## [1] 151.6374
```

The distribution is also `Leptokurtic`, since the value is greater than 3.

```
ggplot(liver, aes(x = Liver_Disease, y = Aspartate_Aminotransferase,
                  fill = Liver_Disease))  +
  geom_boxplot() +
  ylab("Aspartate Aminotransferase") +
  ggtitle("Boxplot of the Aspartate Aminotransferase across the
Liver_Disease")
```




```
ggplot(liver, aes(Liver_Disease, Aspartate_Aminotransferase)) +
  geom_boxplot(aes(fill = Gender))
```

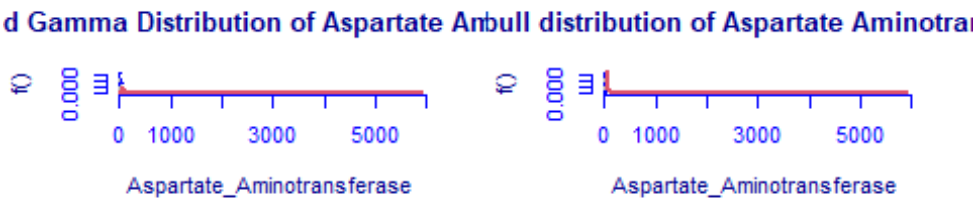


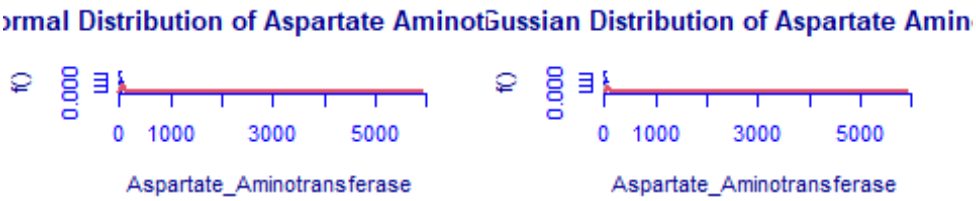
Another skewed variable with a range from 10 to 4929, with a mean of 110.4145, and median of 42. The range is wider within the Liver_Disease = 0 group in comparison to the Liver_Disease = 1 group. The mean is also higher in the Liver_Disease = 0 group. The box plots are hard to really read but they are there for you to look at.

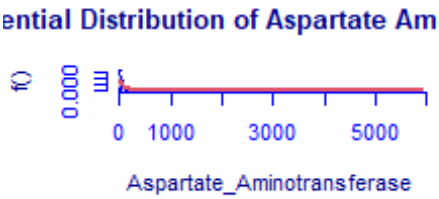
Aspartate Aminotransferase Fit for the Data

Now we will try to fit different models to Aspartate_Aminotransferase distribution evaluating the Akaike Information Criterion (AIC) and the Bayesian Information criterion (BIC), The lower are the indexes, the better is the fitting. I will evaluate both the single parameter distributions and mixture distributions as follows:

```
library(MASS)
library(gamlss)


par(mfrow=c(3,2))

aam.GG <- histDist(liver$Aspartate_Aminotransferase, family=GG, nbins = 177,
xlab = "Aspartate_Aminotransferase", main="Generalized Gamma Distribution of
Aspartate Aminotransferase")

aam.WEI <- histDist(liver$Aspartate_Aminotransferase, family=WEI, nbins =
177, xlab = "Aspartate_Aminotransferase", main="Weibull distribution of
Aspartate Aminotransferase")

aam.LOGNO <- histDist(liver$Aspartate_Aminotransferase, family=LOGNO, nbins =
177, xlab = "Aspartate_Aminotransferase", main="Log-Normal Distribution of
Aspartate Aminotransferase")

aam.IG <- histDist(liver$Aspartate_Aminotransferase, family=IG, nbins=177,
xlab = "Aspartate_Aminotransferase", main = "Inverse Gussian Distribution of
Aspartate Aminotransferase")

aam.EXP<- histDist(liver$Aspartate_Aminotransferase, family=EXP, nbins=177,
xlab = "Aspartate_Aminotransferase", main = "Exponential Distribution of
Aspartate Aminotransferase")
```








```
library(Matrix)
library(glmnet)
df <- data.frame(Rownames = c("Generalized Gamma", "Weibull",
                              "Log-Normal", "Inverse Gussian", "Exponential"),
                 AIC = c(AIC(aam.GG), AIC(aam.WEI), AIC(aam.LOGNO),
                         AIC(aam.IG), AIC(aam.EXP)),
                 BIC = c(aam.GG$sbc, aam.WEI$sbc, aam.LOGNO$sbc,
                         aam.IG$sbc, aam.EXP$sbc),
                 df = c(aam.GG$df.fit, aam.WEI$df.fit,
                        aam.LOGNO$df.fit, aam.IG$df.fit, aam.EXP$df.fit),
                 LogLike = c(logLik(aam.GG), logLik(aam.WEI),
                             logLik(aam.LOGNO), logLik(aam.IG),
logLik(aam.EXP)))
df

##            Rownames      AIC      BIC df   LogLike
## 1 Generalized Gamma 6072.943 6086.027  3 -3033.472
## 2           Weibull 6518.690 6527.413  2 -3257.345
## 3        Log-Normal 6230.840 6239.562  2 -3113.420
## 4   Inverse Gussian 6204.105 6212.827  2 -3100.052
## 5       Exponential 6607.512 6611.873  1 -3302.756
```

As we can see, the model with the highest log likelihood (-3033.472) and the lowest AIC (6072.943) and BIC (6086.027) is the Generalized Gamma Distribution. Therefore, based on the greatest likelihood technique, our data fits better in the Generalized Gamma Distribution. Now, I will also compare two models by means of the Likelihood ratio test as we performed above as follows:

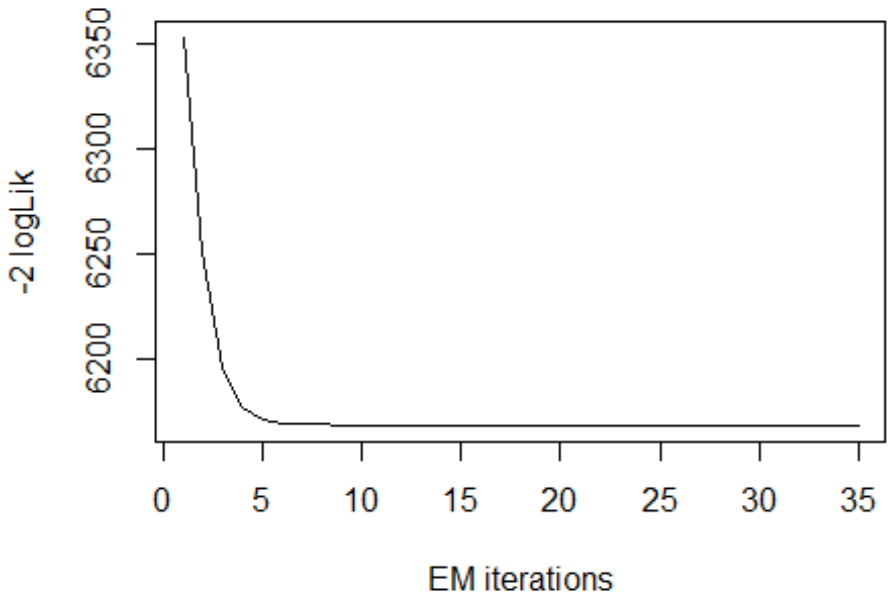
```
library(zoo)
library(lmtest)
lrtest(aam.GG, aam.EXP)

## Likelihood ratio test
##
## Model 1: gamlssML(formula = liver$Aspartate_Aminotransferase, family =
"GG")
## Model 2: gamlssML(formula = liver$Aspartate_Aminotransferase, family =
"EXP")
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   3 -3033.5
## 2   1 -3302.8 -2 538.57  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Under the null hypothesis, we compare the Generalized Gamma distribution to the Exponential distribution under the alternative hypothesis. At any level of significance, we can reject the null hypothesis. Because our distributions fit better in the Generalized Gamma model, we'll use it.

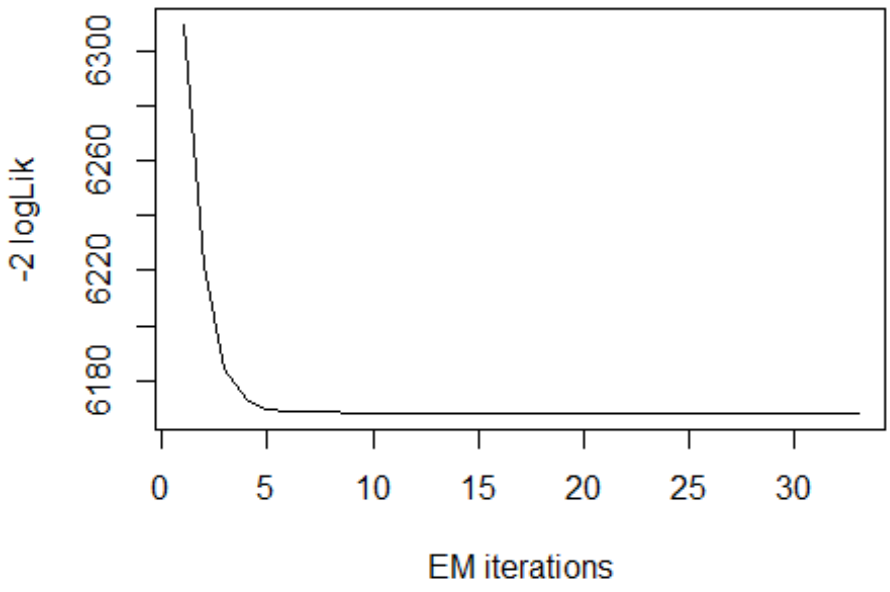
Distributions Mixture

Now, I will also apply the fitting criteria for the distributions with Gamma mixture as follows:

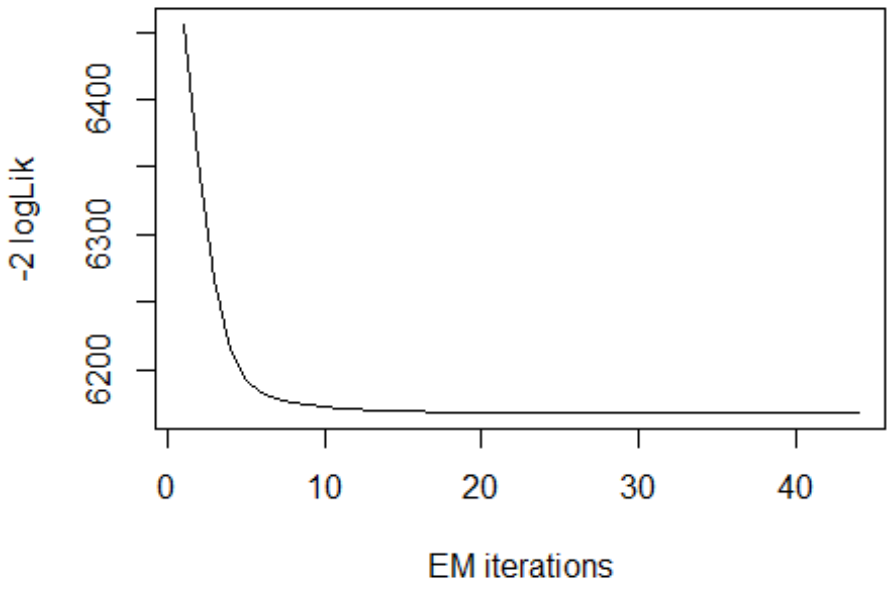
```
library(gamlss.mx)
mix.gam <- gamlssMXfits(n = 5, liver$Aspartate_Aminotransferase~1, family =
GA, K = 2, data = liver)
```



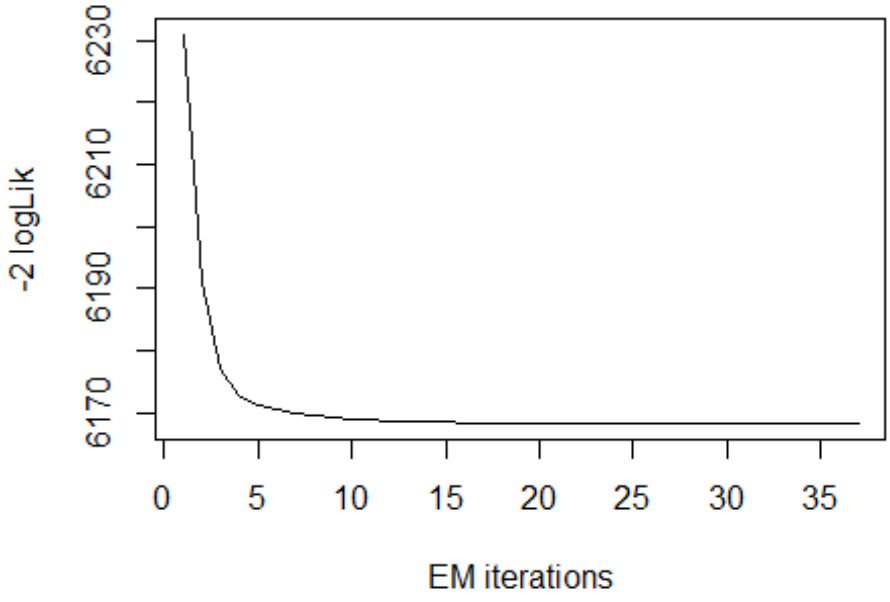

```
## model= 1
```



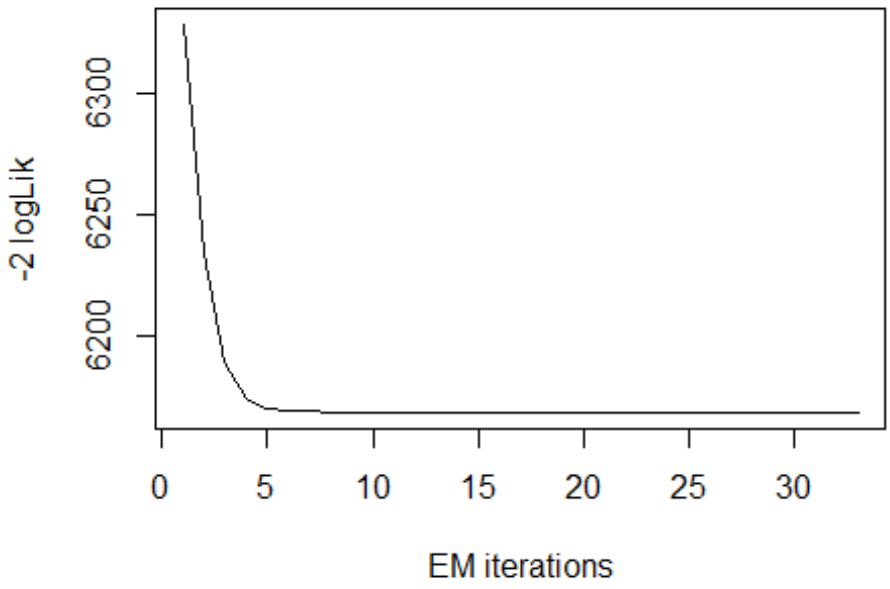

```
## model= 2
```




```
## model= 3
```




```
## model= 4
```




```
## model= 5

print(mix.gam)

##
## Mixing Family:  c("GA", "GA")
##
## Fitting method: EM algorithm
##
## Call:  gamlssMX(formula = liver$Aspartate_Aminotransferase ~
##     1, family = GA, K = 2, data = liver)
##
## Mu Coefficients for model: 1
## (Intercept)
##       5.737
## Sigma Coefficients for model: 1
## (Intercept)
##     0.03021
## Mu Coefficients for model: 2
## (Intercept)
##       3.712
## Sigma Coefficients for model: 2
## (Intercept)
##      -0.609
##
## Estimated probabilities: 0.2580254 0.7419746
##
```

```
## Degrees of Freedom for the fit: 5 Residual Deg. of Freedom   574
## Global Deviance:     6168.29
##             AIC:     6178.29
##             SBC:     6200.1
```

We can observe that the AIC value of the mixture of Gamma has improved, since it is higher than that of the single Gamma distribution. The current `AIC` value is `6178.29`, whereas the previous value was `6072.943` and the current `BIC` value is `6200.1`, whereas the previous value was `6086.027`.

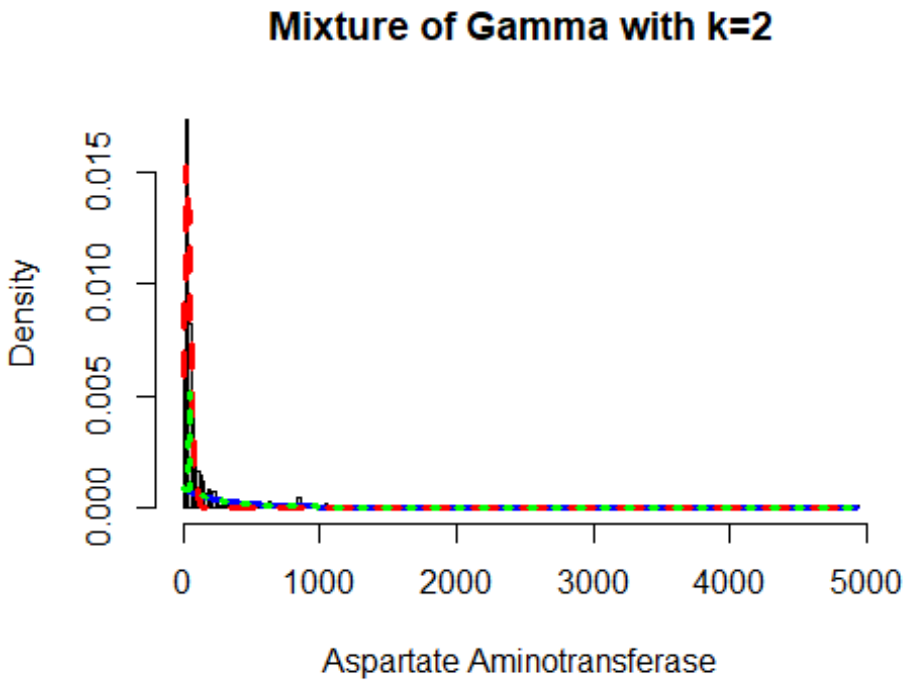
```
logLik(mix.gam)

## 'log Lik.' -3084.147 (df=5)

mix.gam$prob

## [1] 0.2580254 0.7419746

fitted(mix.gam, "mu")[1]

## [1] 110.3985

fitted(mix.gam, "sigma")[2]

## [1] 110.3985

hist(liver$Aspartate_Aminotransferase, breaks = 177, xlab = "Aspartate
Aminotransferase", main="Mixture of Gamma with k=2", freq = FALSE)

mu.hat1 <- exp(mix.gam[["models"]][[1]][["mu.coefficients"]])

sigma.hat1 <- exp(mix.gam[["models"]][[1]][["sigma.coefficients"]])

mu.hat2 <- exp(mix.gam[["models"]][[2]][["mu.coefficients"]])

sigma.hat2 <- exp(mix.gam[["models"]][[2]][["sigma.coefficients"]])

hist(liver$Aspartate_Aminotransferase, breaks = 177, freq = FALSE, plot =
FALSE)

## Warning in hist.default(liver$Aspartate_Aminotransferase, breaks = 177, :
## argument 'freq' is not made use of

## $breaks
##   [1]    0   20   40   60   80  100  120  140  160  180  200  220  240
260  280
##  [16]  300  320  340  360  380  400  420  440  460  480  500  520  540
560  580
##  [31]  600  620  640  660  680  700  720  740  760  780  800  820  840
860  880
##  [46]  900  920  940  960  980 1000 1020 1040 1060 1080 1100 1120 1140
1160 1180
```

```
##  [61] 1200 1220 1240 1260 1280 1300 1320 1340 1360 1380 1400 1420 1440
1460 1480
##  [76] 1500 1520 1540 1560 1580 1600 1620 1640 1660 1680 1700 1720 1740
1760 1780
##  [91] 1800 1820 1840 1860 1880 1900 1920 1940 1960 1980 2000 2020 2040
2060 2080
## [106] 2100 2120 2140 2160 2180 2200 2220 2240 2260 2280 2300 2320 2340
2360 2380
## [121] 2400 2420 2440 2460 2480 2500 2520 2540 2560 2580 2600 2620 2640
2660 2680
## [136] 2700 2720 2740 2760 2780 2800 2820 2840 2860 2880 2900 2920 2940
2960 2980
## [151] 3000 3020 3040 3060 3080 3100 3120 3140 3160 3180 3200 3220 3240
3260 3280
## [166] 3300 3320 3340 3360 3380 3400 3420 3440 3460 3480 3500 3520 3540
3560 3580
## [181] 3600 3620 3640 3660 3680 3700 3720 3740 3760 3780 3800 3820 3840
3860 3880
## [196] 3900 3920 3940 3960 3980 4000 4020 4040 4060 4080 4100 4120 4140
4160 4180
## [211] 4200 4220 4240 4260 4280 4300 4320 4340 4360 4380 4400 4420 4440
4460 4480
## [226] 4500 4520 4540 4560 4580 4600 4620 4640 4660 4680 4700 4720 4740
4760 4780
## [241] 4800 4820 4840 4860 4880 4900 4920 4940
##
## $counts
##   [1]  80 201  95  46  34  19  17  14   7   9   2   8   5   1   2   0   1
3
##  [19]   2   3   3   0   1   0   2   1   1   0   2   0   1   3   0   0   0
0
##  [37]   2   0   0   1   0   0   5   0   0   0   0   2   0   0   0   0   2
0
##  [55]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
##  [73]   0   0   1   0   0   0   0   1   0   0   0   0   0   0   0   0   0
0
##  [91]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [109]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [127]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [145]   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [163]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [181]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [199]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

```
0
## [217]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [235]   0   0   0   0   0   0   0   0   0   0   0   0   1
##
## $density
##   [1] 6.908463e-03 1.735751e-02 8.203800e-03 3.972366e-03 2.936097e-03
##   [6] 1.640760e-03 1.468048e-03 1.208981e-03 6.044905e-04 7.772021e-04
##  [11] 1.727116e-04 6.908463e-04 4.317789e-04 8.635579e-05 1.727116e-04
##  [16] 0.000000e+00 8.635579e-05 2.590674e-04 1.727116e-04 2.590674e-04
##  [21] 2.590674e-04 0.000000e+00 8.635579e-05 0.000000e+00 1.727116e-04
##  [26] 8.635579e-05 8.635579e-05 0.000000e+00 1.727116e-04 0.000000e+00
##  [31] 8.635579e-05 2.590674e-04 0.000000e+00 0.000000e+00 0.000000e+00
##  [36] 0.000000e+00 1.727116e-04 0.000000e+00 0.000000e+00 8.635579e-05
##  [41] 0.000000e+00 0.000000e+00 4.317789e-04 0.000000e+00 0.000000e+00
##  [46] 0.000000e+00 0.000000e+00 1.727116e-04 0.000000e+00 0.000000e+00
##  [51] 0.000000e+00 0.000000e+00 1.727116e-04 0.000000e+00 0.000000e+00
##  [56] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [61] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [66] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [71] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 8.635579e-05
##  [76] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 8.635579e-05
##  [81] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [86] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [91] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [96] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [101] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [106] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [111] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [116] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [121] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [126] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [131] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [136] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [141] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [146] 0.000000e+00 0.000000e+00 8.635579e-05 0.000000e+00 0.000000e+00
## [151] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [156] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [161] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [166] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [171] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [176] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [181] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [186] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [191] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [196] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [201] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [206] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [211] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [216] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
```

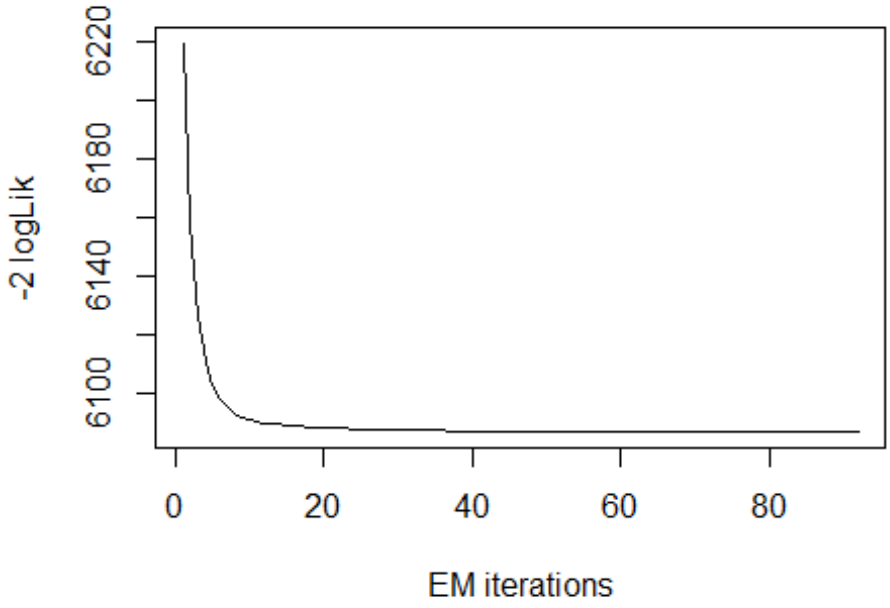
```
## [221] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [226] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [231] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [236] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [241] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [246] 0.000000e+00 8.635579e-05
##
## $mids
##   [1]   10   30   50   70   90  110  130  150  170  190  210  230  250
270  290
##  [16]  310  330  350  370  390  410  430  450  470  490  510  530  550
570  590
##  [31]  610  630  650  670  690  710  730  750  770  790  810  830  850
870  890
##  [46]  910  930  950  970  990 1010 1030 1050 1070 1090 1110 1130 1150
1170 1190
##  [61] 1210 1230 1250 1270 1290 1310 1330 1350 1370 1390 1410 1430 1450
1470 1490
##  [76] 1510 1530 1550 1570 1590 1610 1630 1650 1670 1690 1710 1730 1750
1770 1790
##  [91] 1810 1830 1850 1870 1890 1910 1930 1950 1970 1990 2010 2030 2050
2070 2090
## [106] 2110 2130 2150 2170 2190 2210 2230 2250 2270 2290 2310 2330 2350
2370 2390
## [121] 2410 2430 2450 2470 2490 2510 2530 2550 2570 2590 2610 2630 2650
2670 2690
## [136] 2710 2730 2750 2770 2790 2810 2830 2850 2870 2890 2910 2930 2950
2970 2990
## [151] 3010 3030 3050 3070 3090 3110 3130 3150 3170 3190 3210 3230 3250
3270 3290
## [166] 3310 3330 3350 3370 3390 3410 3430 3450 3470 3490 3510 3530 3550
3570 3590
## [181] 3610 3630 3650 3670 3690 3710 3730 3750 3770 3790 3810 3830 3850
3870 3890
## [196] 3910 3930 3950 3970 3990 4010 4030 4050 4070 4090 4110 4130 4150
4170 4190
## [211] 4210 4230 4250 4270 4290 4310 4330 4350 4370 4390 4410 4430 4450
4470 4490
## [226] 4510 4530 4550 4570 4590 4610 4630 4650 4670 4690 4710 4730 4750
4770 4790
## [241] 4810 4830 4850 4870 4890 4910 4930
##
## $xname
## [1] "liver$Aspartate_Aminotransferase"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```
lines(seq(min(liver$Aspartate_Aminotransferase),
max(liver$Aspartate_Aminotransferase), length =
length(liver$Aspartate_Aminotransferase)),
      mix.gam[["prob"]][1]*dGA(seq(min(liver$Aspartate_Aminotransferase),
max(liver$Aspartate_Aminotransferase),
      length = length(liver$Aspartate_Aminotransferase)), mu = mu.hat1, sigma
= sigma.hat1),
      lty=1, lwd=3, col="blue")

lines(seq(min(liver$Aspartate_Aminotransferase),
max(liver$Aspartate_Aminotransferase), length =
length(liver$Aspartate_Aminotransferase)),
      mix.gam[["prob"]][2]*dGA(seq(min(liver$Aspartate_Aminotransferase),
max(liver$Aspartate_Aminotransferase),
      length = length(liver$Aspartate_Aminotransferase)), mu = mu.hat2, sigma
= sigma.hat2),
      lty = 2, lwd = 3, col = "red")

lines(seq(min(liver$Aspartate_Aminotransferase),
max(liver$Aspartate_Aminotransferase), length =
length(liver$Aspartate_Aminotransferase)),
      mix.gam[["prob"]][1]*dGA(seq(min(liver$Aspartate_Aminotransferase),
max(liver$Aspartate_Aminotransferase),
      length = length(liver$Aspartate_Aminotransferase)), mu = mu.hat1, sigma
= sigma.hat1)+
      mix.gam[["prob"]][2]*dRG(seq(min(liver$Aspartate_Aminotransferase),
max(liver$Aspartate_Aminotransferase),
      length = length(liver$Aspartate_Aminotransferase)), mu = mu.hat2, sigma
= sigma.hat2),
      lty = 3, lwd = 3, col = "green")
```

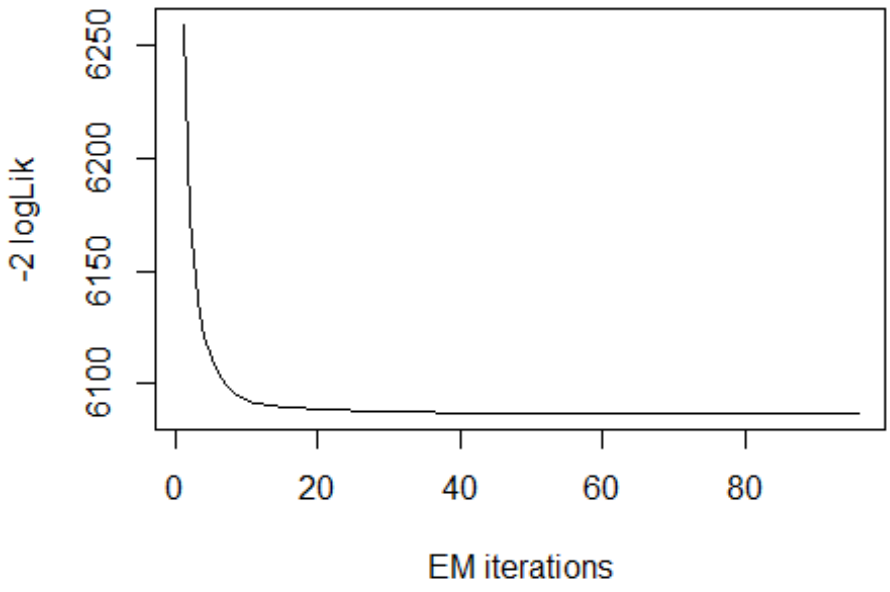


Now, I will again apply the fitting criteria for the distributions with Gamma mixture with the k = 3 as follows:

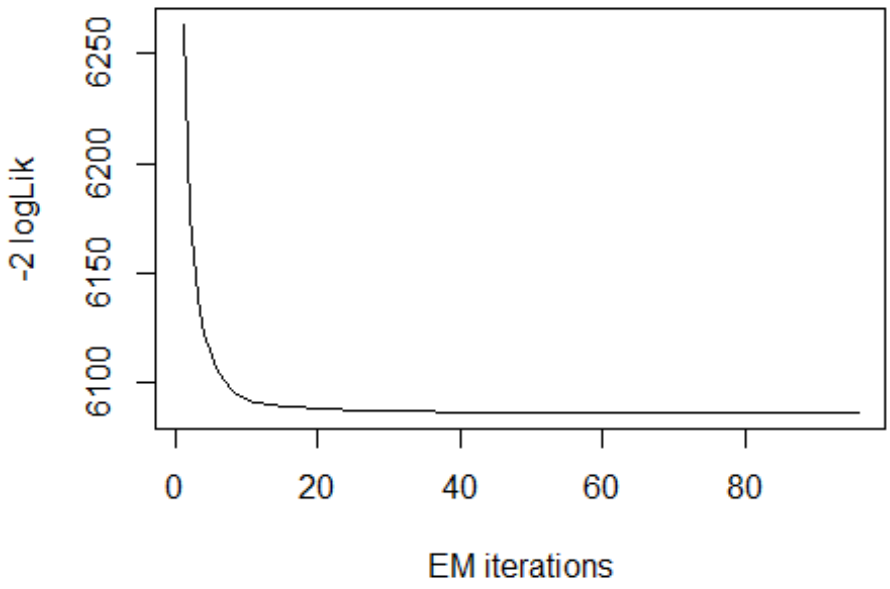
```
mix.gam.3 <- gamlssMXfits(n = 5, liver$Aspartate_Aminotransferase~1, family =
GA, K = 3, data = liver)
```



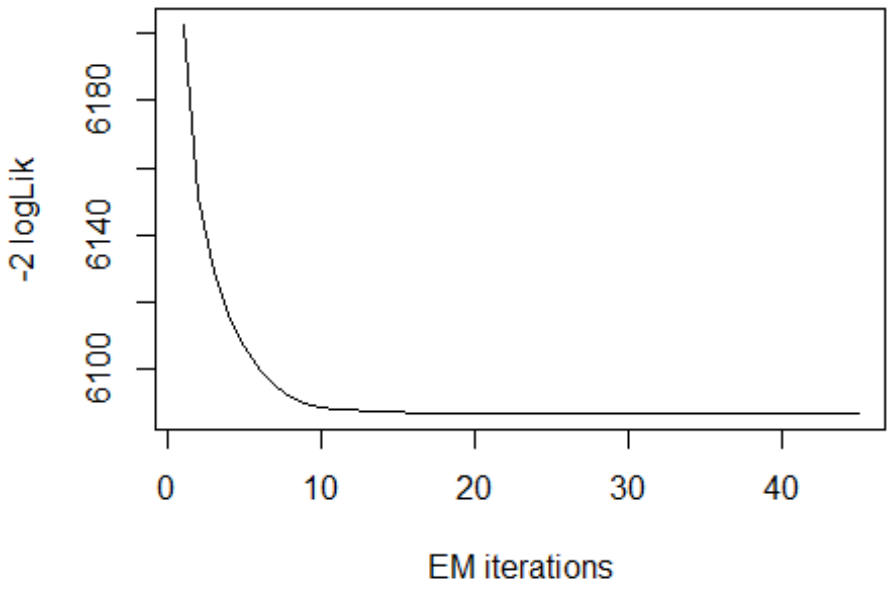

```
## model= 1
```



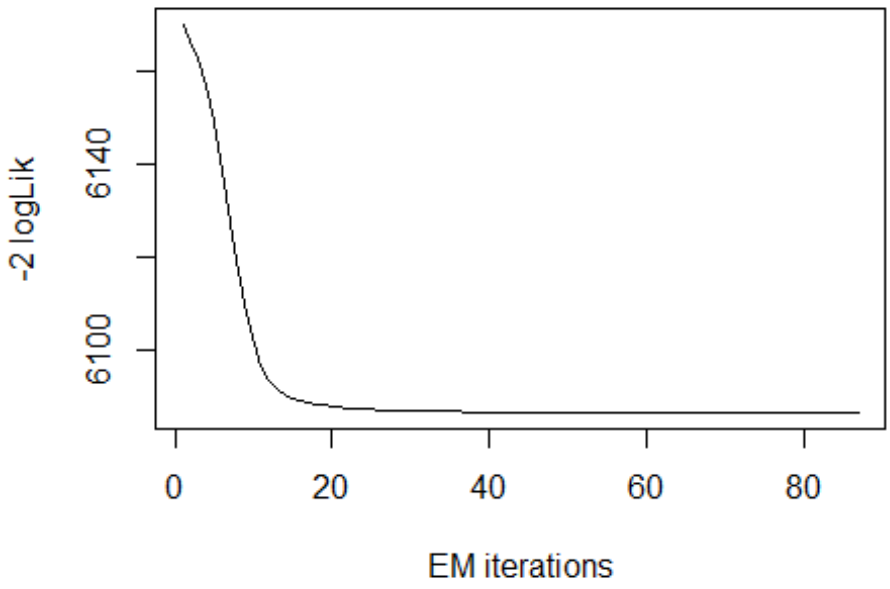

```
## model= 2
```




```
## model= 3
```




```
## model= 4
```




```
## model= 5

print(mix.gam.3)

##
## Mixing Family:  c("GA", "GA", "GA")
##
## Fitting method: EM algorithm
##
## Call:  gamlssMX(formula = liver$Aspartate_Aminotransferase ~
##     1, family = GA, K = 3, data = liver)
##
## Mu Coefficients for model: 1
## (Intercept)
##       3.347
## Sigma Coefficients for model: 1
## (Intercept)
##     -0.9957
## Mu Coefficients for model: 2
## (Intercept)
##        4.41
## Sigma Coefficients for model: 2
## (Intercept)
##     -0.5401
## Mu Coefficients for model: 3
## (Intercept)
##       6.256
```

```
## Sigma Coefficients for model: 3
## (Intercept)
##    -0.03811
##
## Estimated probabilities: 0.4764296 0.4009889 0.1225815
##
## Degrees of Freedom for the fit: 8 Residual Deg. of Freedom   571
## Global Deviance:     6086.75
##             AIC:     6102.75
##             SBC:     6137.64
```

```
logLik(mix.gam.3)
```

```
## 'log Lik.' -3043.376 (df=8)
```

```
mix.gam.3$prob
```

```
## [1] 0.4764296 0.4009889 0.1225815
```

```
fitted(mix.gam.3, "mu")[1]
```

```
## [1] 110.3993
```

```
fitted(mix.gam.3, "sigma")[2]
```

```
## [1] 110.3993
```

```
hist(liver$Aspartate_Aminotransferase, breaks = 177, xlab = "Aspartate
Aminotransferase", main="Mixture of Gamma with k=3", freq = FALSE)
```

```
mu.hat1 <- exp(mix.gam.3[["models"]][[1]][["mu.coefficients"]])
```

```
sigma.hat1 <- exp(mix.gam.3[["models"]][[1]][["sigma.coefficients"]])
```

```
mu.hat2 <- exp(mix.gam.3[["models"]][[2]][["mu.coefficients"]])
```

```
sigma.hat2 <- exp(mix.gam.3[["models"]][[2]][["sigma.coefficients"]])
```

```
hist(liver$Aspartate_Aminotransferase, breaks = 177, freq = FALSE, plot =
FALSE)
```

```
## Warning in hist.default(liver$Aspartate_Aminotransferase, breaks = 177, :
## argument 'freq' is not made use of
```

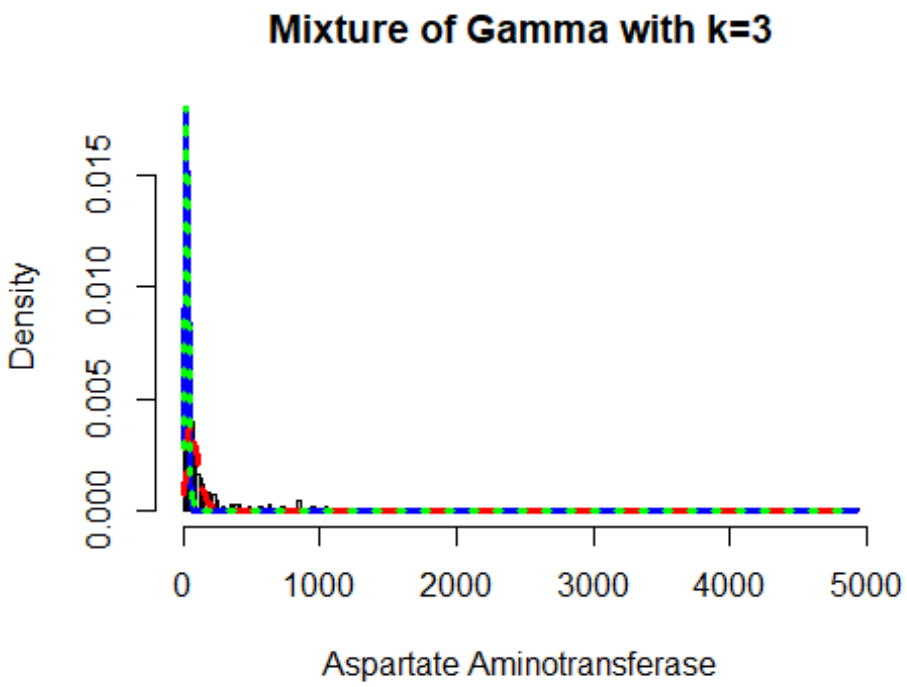
```
## $breaks
##   [1]    0   20   40   60   80  100  120  140  160  180  200  220  240
260  280
##  [16]  300  320  340  360  380  400  420  440  460  480  500  520  540
560  580
##  [31]  600  620  640  660  680  700  720  740  760  780  800  820  840
860  880
##  [46]  900  920  940  960  980 1000 1020 1040 1060 1080 1100 1120 1140
```

```
1160 1180
##  [61] 1200 1220 1240 1260 1280 1300 1320 1340 1360 1380 1400 1420 1440
1460 1480
##  [76] 1500 1520 1540 1560 1580 1600 1620 1640 1660 1680 1700 1720 1740
1760 1780
##  [91] 1800 1820 1840 1860 1880 1900 1920 1940 1960 1980 2000 2020 2040
2060 2080
## [106] 2100 2120 2140 2160 2180 2200 2220 2240 2260 2280 2300 2320 2340
2360 2380
## [121] 2400 2420 2440 2460 2480 2500 2520 2540 2560 2580 2600 2620 2640
2660 2680
## [136] 2700 2720 2740 2760 2780 2800 2820 2840 2860 2880 2900 2920 2940
2960 2980
## [151] 3000 3020 3040 3060 3080 3100 3120 3140 3160 3180 3200 3220 3240
3260 3280
## [166] 3300 3320 3340 3360 3380 3400 3420 3440 3460 3480 3500 3520 3540
3560 3580
## [181] 3600 3620 3640 3660 3680 3700 3720 3740 3760 3780 3800 3820 3840
3860 3880
## [196] 3900 3920 3940 3960 3980 4000 4020 4040 4060 4080 4100 4120 4140
4160 4180
## [211] 4200 4220 4240 4260 4280 4300 4320 4340 4360 4380 4400 4420 4440
4460 4480
## [226] 4500 4520 4540 4560 4580 4600 4620 4640 4660 4680 4700 4720 4740
4760 4780
## [241] 4800 4820 4840 4860 4880 4900 4920 4940
##
## $counts
##   [1]  80 201  95  46  34  19  17  14   7   9   2   8   5   1   2   0   1
3
##  [19]   2   3   3   0   1   0   2   1   1   0   2   0   1   3   0   0   0
0
##  [37]   2   0   0   1   0   0   5   0   0   0   0   2   0   0   0   0   2
0
##  [55]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
##  [73]   0   0   1   0   0   0   0   1   0   0   0   0   0   0   0   0   0
0
##  [91]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [109]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [127]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [145]   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [163]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [181]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
```

```
## [199]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [217]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
## [235]   0   0   0   0   0   0   0   0   0   0   0   0   1
##
## $density
##   [1] 6.908463e-03 1.735751e-02 8.203800e-03 3.972366e-03 2.936097e-03
##   [6] 1.640760e-03 1.468048e-03 1.208981e-03 6.044905e-04 7.772021e-04
##  [11] 1.727116e-04 6.908463e-04 4.317789e-04 8.635579e-05 1.727116e-04
##  [16] 0.000000e+00 8.635579e-05 2.590674e-04 1.727116e-04 2.590674e-04
##  [21] 2.590674e-04 0.000000e+00 8.635579e-05 0.000000e+00 1.727116e-04
##  [26] 8.635579e-05 8.635579e-05 0.000000e+00 1.727116e-04 0.000000e+00
##  [31] 8.635579e-05 2.590674e-04 0.000000e+00 0.000000e+00 0.000000e+00
##  [36] 0.000000e+00 1.727116e-04 0.000000e+00 0.000000e+00 8.635579e-05
##  [41] 0.000000e+00 0.000000e+00 4.317789e-04 0.000000e+00 0.000000e+00
##  [46] 0.000000e+00 0.000000e+00 1.727116e-04 0.000000e+00 0.000000e+00
##  [51] 0.000000e+00 0.000000e+00 1.727116e-04 0.000000e+00 0.000000e+00
##  [56] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [61] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [66] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [71] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 8.635579e-05
##  [76] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 8.635579e-05
##  [81] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [86] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [91] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [96] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [101] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [106] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [111] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [116] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [121] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [126] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [131] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [136] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [141] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [146] 0.000000e+00 0.000000e+00 8.635579e-05 0.000000e+00 0.000000e+00
## [151] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [156] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [161] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [166] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [171] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [176] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [181] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [186] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [191] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [196] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [201] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [206] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [211] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
```

```
## [216] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [221] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [226] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [231] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [236] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [241] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [246] 0.000000e+00 8.635579e-05
##
## $mids
##   [1]   10   30   50   70   90  110  130  150  170  190  210  230  250
270  290
##  [16]  310  330  350  370  390  410  430  450  470  490  510  530  550
570  590
##  [31]  610  630  650  670  690  710  730  750  770  790  810  830  850
870  890
##  [46]  910  930  950  970  990 1010 1030 1050 1070 1090 1110 1130 1150
1170 1190
##  [61] 1210 1230 1250 1270 1290 1310 1330 1350 1370 1390 1410 1430 1450
1470 1490
##  [76] 1510 1530 1550 1570 1590 1610 1630 1650 1670 1690 1710 1730 1750
1770 1790
##  [91] 1810 1830 1850 1870 1890 1910 1930 1950 1970 1990 2010 2030 2050
2070 2090
## [106] 2110 2130 2150 2170 2190 2210 2230 2250 2270 2290 2310 2330 2350
2370 2390
## [121] 2410 2430 2450 2470 2490 2510 2530 2550 2570 2590 2610 2630 2650
2670 2690
## [136] 2710 2730 2750 2770 2790 2810 2830 2850 2870 2890 2910 2930 2950
2970 2990
## [151] 3010 3030 3050 3070 3090 3110 3130 3150 3170 3190 3210 3230 3250
3270 3290
## [166] 3310 3330 3350 3370 3390 3410 3430 3450 3470 3490 3510 3530 3550
3570 3590
## [181] 3610 3630 3650 3670 3690 3710 3730 3750 3770 3790 3810 3830 3850
3870 3890
## [196] 3910 3930 3950 3970 3990 4010 4030 4050 4070 4090 4110 4130 4150
4170 4190
## [211] 4210 4230 4250 4270 4290 4310 4330 4350 4370 4390 4410 4430 4450
4470 4490
## [226] 4510 4530 4550 4570 4590 4610 4630 4650 4670 4690 4710 4730 4750
4770 4790
## [241] 4810 4830 4850 4870 4890 4910 4930
##
## $xname
## [1] "liver$Aspartate_Aminotransferase"
##
## $equidist
## [1] TRUE
##
```

```
## attr(,"class")
## [1] "histogram"

lines(seq(min(liver$Aspartate_Aminotransferase),
max(liver$Aspartate_Aminotransferase), length =
length(liver$Aspartate_Aminotransferase)),
      mix.gam.3[["prob"]][1]*dGA(seq(min(liver$Aspartate_Aminotransferase),
max(liver$Aspartate_Aminotransferase),
      length = length(liver$Aspartate_Aminotransferase)), mu = mu.hat1, sigma
= sigma.hat1),
      lty=1, lwd=3, col="blue")

lines(seq(min(liver$Aspartate_Aminotransferase),
max(liver$Aspartate_Aminotransferase), length =
length(liver$Aspartate_Aminotransferase)),
      mix.gam.3[["prob"]][2]*dGA(seq(min(liver$Aspartate_Aminotransferase),
max(liver$Aspartate_Aminotransferase),
      length = length(liver$Aspartate_Aminotransferase)), mu = mu.hat2, sigma
= sigma.hat2),
      lty = 2, lwd = 3, col = "red")

lines(seq(min(liver$Aspartate_Aminotransferase),
max(liver$Aspartate_Aminotransferase), length =
length(liver$Aspartate_Aminotransferase)),
      mix.gam.3[["prob"]][1]*dGA(seq(min(liver$Aspartate_Aminotransferase),
max(liver$Aspartate_Aminotransferase),
      length = length(liver$Aspartate_Aminotransferase)), mu = mu.hat1, sigma
= sigma.hat1)+
      mix.gam.3[["prob"]][2]*dRG(seq(min(liver$Aspartate_Aminotransferase),
max(liver$Aspartate_Aminotransferase),
      length = length(liver$Aspartate_Aminotransferase)), mu = mu.hat2, sigma
= sigma.hat2),
      lty = 3, lwd = 3, col = "green")
```




```
mix.gm.tb <- data.frame(row.names=c('Gamma Mixture, K=3','Gamma Mixture,
K=2', "Generalized Gamma"),
                        AIC=c(mix.gam.3$aic, mix.gam$aic, AIC(aam.GG)),
                        BIC=c(mix.gam.3$sbc, mix.gam$sbc, aam.GG$sbc))
mix.gm.tb
```

```
##                           AIC      BIC
## Gamma Mixture, K=3 6102.752 6137.642
## Gamma Mixture, K=2 6178.294 6200.101
## Generalized Gamma  6072.943 6086.027
```

We can observe that the AIC value of the mixture of Gamma with k=3 and k=2 has increased, since values are higher than that of the single Gamma distribution. The previous AIC value is `6072.943`, whereas the current value which is higher is `6102.752` and the previous BIC value was `6086.027`, whereas the current value which is higher is `6137.642`. Here we can clearly see that our data fits better in the single Gamma Distribution.

Total Protiens

Lets analyze the features of `Total Protiens` variable as follows:

```
head(liver$Total_Protiens)
```

```
## [1] 6.8 7.5 7.0 6.8 7.3 7.6
```

```
length(liver$Total_Protiens)
```

```
## [1] 579
```

```
table(liver$Total_Protiens)
```

```
##
## 2.7 2.8   3 3.6 3.7 3.8 3.9   4 4.1 4.3 4.4 4.5 4.6 4.7 4.8 4.9   5 5.1
5.2 5.3
##   1   1   1   3   1   2   2   2   2   3   4   4   4   2   3   6  11  10
11  10
## 5.4 5.5 5.6 5.7 5.8 5.9   6 6.1 6.2 6.3 6.4 6.5 6.6 6.7 6.8 6.9   7 7.1
7.2 7.3
##  13  17  18  11  14  14  30  18  24  14  18  14  15  15  28  25  32  22
21  18
## 7.4 7.5 7.6 7.7 7.8 7.9   8 8.1 8.2 8.3 8.4 8.5 8.6 8.7 8.9 9.2 9.5 9.6
##  12  15   9   3   9  14  20   6   8   3   3   4   3   1   1   2   1   1
```

The table() function in R returns the absolute frequencies for each patient-specified value in the data set.

```
unique(liver$Total_Protiens)
```

```
##  [1] 6.8 7.5 7.0 7.3 7.6 6.7 7.4 5.9 8.1 5.8 5.5 6.4 4.3 6.0 5.0 7.2 3.9
5.2 4.9
## [20] 5.6 6.9 6.2 5.1 6.1 6.5 5.7 6.6 6.3 8.0 4.4 5.3 4.6 4.7 5.4 7.1 4.0
3.7 2.7
## [39] 3.0 3.8 7.8 4.5 4.1 4.8 7.9 8.5 7.7 8.2 2.8 9.5 9.6 8.3 8.6 8.4 8.9
8.7 3.6
## [58] 9.2
```

```
length(unique(liver$Total_Protiens))
```

```
## [1] 58
```

```
min(liver$Total_Protiens)
```

```
## [1] 2.7
```

```
max(liver$Total_Protiens)
```

```
## [1] 9.6
```

Here the total observation of Total_Protiens is 579 and is a continuous variable that range lies between 2.7-9.. Now, we will see the summary of the Total_Protiens variable, for further analysis as follows:

```
summary(liver$Total_Protiens)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.700   5.800   6.600   6.482   7.200   9.600
```

```
sd(liver$Total_Protiens)
```

```
## [1] 1.084641
```

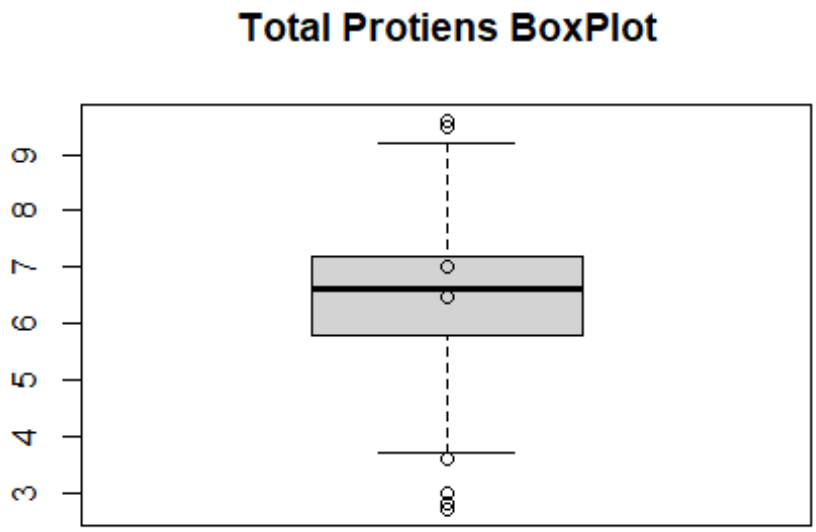
```r
var(liver$Total_Protiens)

## [1] 1.176446

v <- c(liver$Total_Protiens)
mode <- getMode(v)
print(mode)

## [1] 7
```

From the above summary we can observe that the Mean (6.482), Median (6.600) and Mode (7) are not equal so the distribution is asymmetrical. Here Mode > Median > Mean so the distribution could be negative and skewed to the left. Now we also confirm this as follows:

```r
library(moments)
skewness(liver$Total_Protiens)

## [1] -0.2916746
```

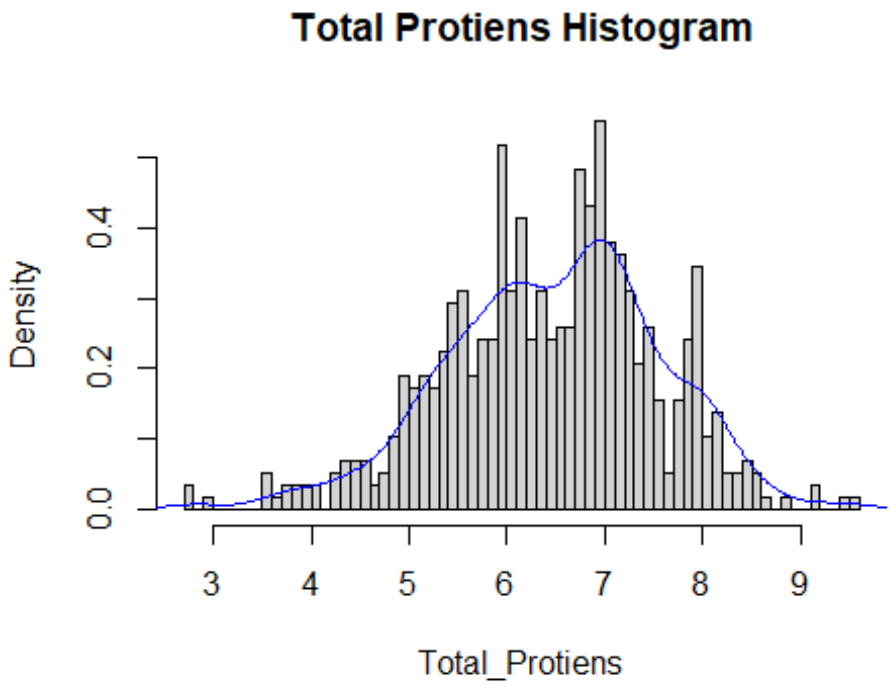
Hence, the skewness is the negative so the distribution is negatively skewed was rightly observed. Also I will plot the Boxplot to comparing the distribution of data across data set as follows:

```r
boxplot(liver$Total_Protiens, main = "Total Protiens BoxPlot")
points(mean(liver$Total_Protiens))
points(mode)
```



Box plots are graphical displays of the five number summary, So they describe the Minimum (lowest mean relative) lower whisker, Quartile 1 (It separates the lowest 25% observation to the rest of the observations), Median (Which divides the lower 50% observation from the upper 50% observations), Quartile 3 (It divides the upper 25% observation to the rest of the observations) and Maximum. In the above Box plot diagram, there are some outliers and the upper whisker shows the highest mean relative. Now, I will plot a histogram a graphical display of data using bars of different heights to measures distribution of continuous data as follows:

```
hist(liver$Total_Protiens, prob = TRUE, breaks = 58, xlab = "Total_Protiens",
main = "Total Protiens Histogram")
lines(density(liver$Total_Protiens), col='blue')
```

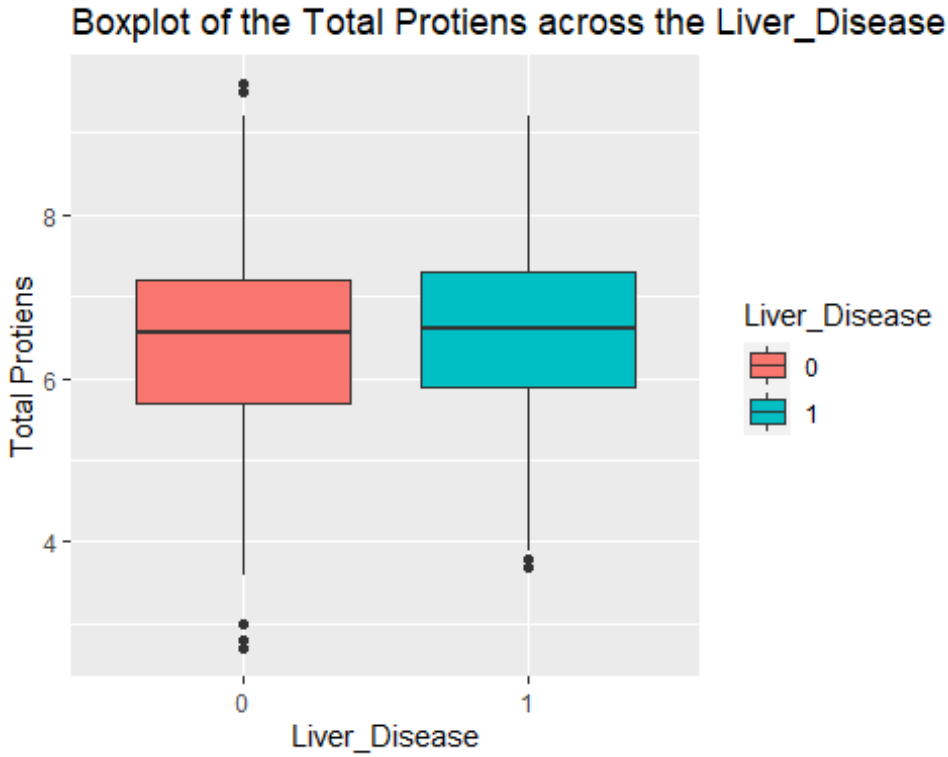


From the above histogram, we can observe that the diagram has many peaks.Using the density lines, I have approximated the histogram graph. Density provides information about the type of distribution under consideration and allows us to determine whether the attribute is distributed normally or not. From the graph above we can see how the `Total_Protiens` variable does not seem to fit to a Normal distribution, there is a peak of the frequency distributions around many values and also a left skewed. We have already observed a negative value of `-0.2916746`, so we will affirm that the distribution is left skewed. To check whether the distribution is `Platykurtic` or not, we will check this by following R method as follows:

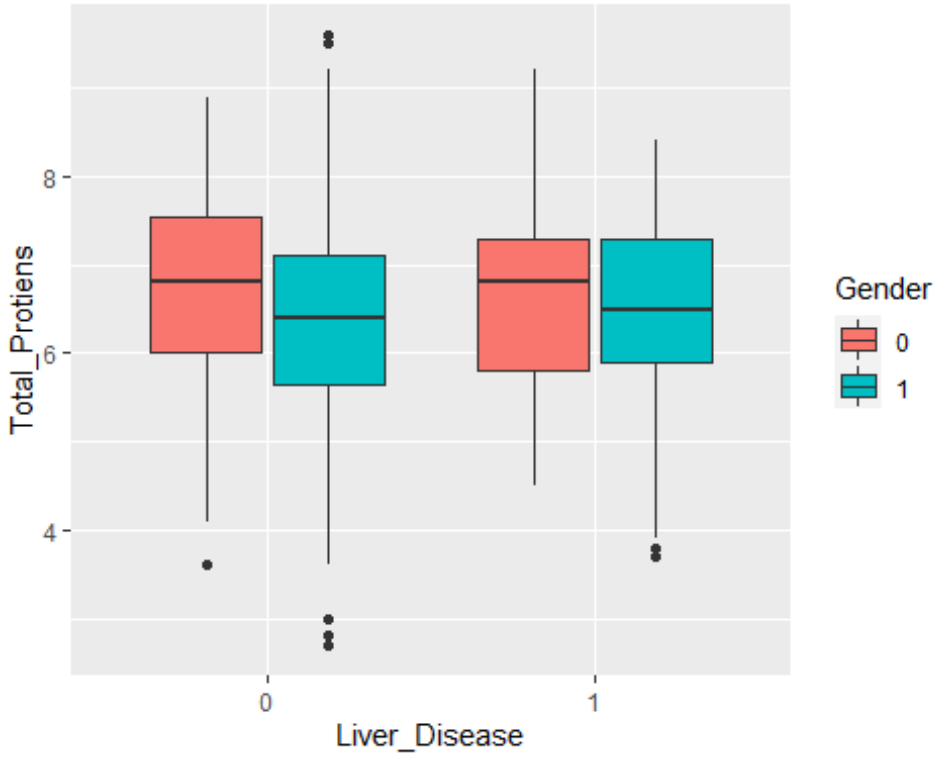
```
kurtosis(liver$Total_Protiens)

## [1] 3.227065
```

The distribution is also Leptokurtic, since the value is greater than 3.

```
ggplot(liver, aes(x = Liver_Disease, y = Total_Protiens, fill =
Liver_Disease)) +
  geom_boxplot() +
  ylab("Total Protiens") +
  ggtitle("Boxplot of the Total Protiens across the Liver_Disease")
```




```
ggplot(liver, aes(Liver_Disease, Total_Protiens)) +
  geom_boxplot(aes(fill = Gender))
```



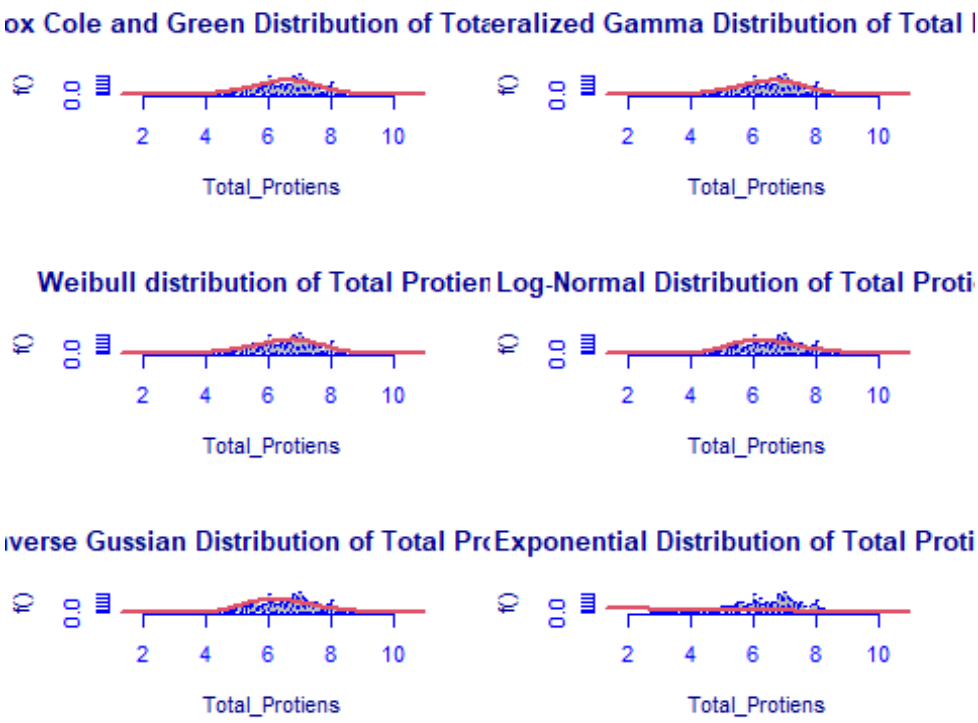
A variable that looks like it has a bell shaped curve. The range for the variable is from 2.7 to 9.6, with a mean of 6.481 and median of 6.6. There does not seem to be much of a difference between the means for the Total Proteins for each response. There is a difference between the responses when you look across the genders. The females have a higher mean for total proteins across the Liver_Disease group.

Total Protien Fit for the Data

Now we will try to fit different models to Total Protiens distribution evaluating the Akaike Information Criterion (AIC) and the Bayesian Information criterion (BIC), The lower are the indexes, the better is the fitting. I will evaluate both the single parameter distributions and mixture distributions as follows:

```
library(MASS)
library(gamlss)


par(mfrow=c(3,2))

tp.BCCG <- histDist(liver$Total_Protiens, family=BCCG, nbins = 58, xlab =
"Total_Protiens", main="Box-Cox Cole and Green Distribution of Total
Protiens")

tp.GG <- histDist(liver$Total_Protiens, family=GG, nbins = 58, xlab =
"Total_Protiens", main="Generalized Gamma Distribution of Total Protiens")
```

```
tp.WEI <- histDist(liver$Total_Protiens, family=WEI, nbins = 58, xlab =
"Total_Protiens", main="Weibull distribution of Total Protiens")

tp.LOGNO <- histDist(liver$Total_Protiens, family=LOGNO, nbins = 58, xlab =
"Total_Protiens", main="Log-Normal Distribution of Total Protiens")

tp.IG <- histDist(liver$Total_Protiens, family=IG, nbins = 58, xlab =
"Total_Protiens", main = "Inverse Gussian Distribution of Total Protiens")

tp.EXP<- histDist(liver$Total_Protiens, family=EXP, nbins = 58, xlab =
"Total_Protiens", main = "Exponential Distribution of Total Protiens")
```




```
library(Matrix)
library(glmnet)
df <- data.frame(Rownames = c("Box-Cox Cole and Green","Generalized Gamma",
"Weibull",
                              "Log-Normal", "Inverse Gussian", "Exponential"),
                 AIC = c(AIC(tp.BCCG), AIC(tp.GG), AIC(tp.WEI),
AIC(tp.LOGNO),
                         AIC(tp.IG), AIC(tp.EXP)),
                 BIC = c(tp.BCCG$sbc, tp.GG$sbc, tp.WEI$sbc,
tp.LOGNO$sbc,
                         tp.IG$sbc, tp.EXP$sbc),
                 df = c(tp.BCCG$df.fit, tp.GG$df.fit, tp.WEI$df.fit,
                        tp.LOGNO$df.fit, tp.IG$df.fit, tp.EXP$df.fit),
                 LogLike = c(logLik(tp.BCCG), logLik(tp.GG),
```

```
logLik(tp.WEI),
                                    logLik(tp.LOGNO), logLik(tp.IG),
logLik(tp.EXP)))
df

##                  Rownames      AIC      BIC df    LogLike
## 1 Box-Cox Cole and Green 1734.355 1747.439  3  -864.1775
## 2      Generalized Gamma 1735.067 1748.151  3  -864.5336
## 3                Weibull 1738.050 1746.773  2  -867.0252
## 4             Log-Normal 1804.629 1813.352  2  -900.3145
## 5        Inverse Gussian 1808.334 1817.057  2  -902.1670
## 6            Exponential 3324.281 3328.642  1 -1661.1404
```

As we can see, the model with the highest log likelihood (-864.1775) and the lowest AIC (1734.355) and BIC (1747.439) is the Box-Cox Cole and Green Distribution. Therefore, based on the greatest likelihood technique, our data fits better in the Box-Cox Cole and Green Distribution. Now, I will also compare two models by means of the Likelihood ratio test as we performed above as follows:

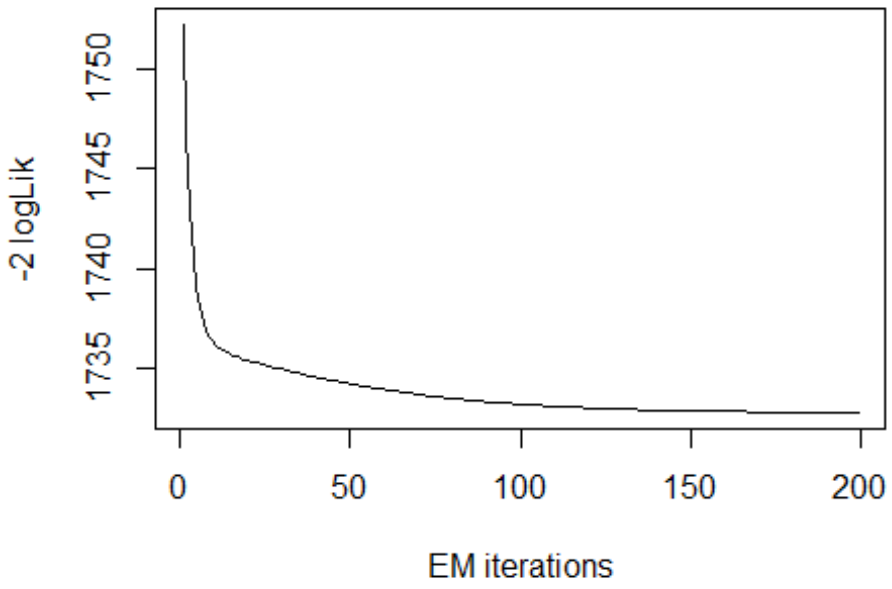
```
library(zoo)
library(lmtest)
lrtest(tp.BCCG, tp.EXP)

## Likelihood ratio test
##
## Model 1: gamlssML(formula = liver$Total_Protiens, family = "BCCG")
## Model 2: gamlssML(formula = liver$Total_Protiens, family = "EXP")
##   #Df   LogLik Df  Chisq Pr(>Chisq)
## 1   3  -864.18
## 2   1 -1661.14 -2 1593.9  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Under the null hypothesis, we compare the Box-Cox Cole and Green distribution to the Exponential distribution under the alternative hypothesis. At any level of significance, we can reject the null hypothesis. Because our distributions fit better in the Box-Cox Cole and Green model, we'll use it.

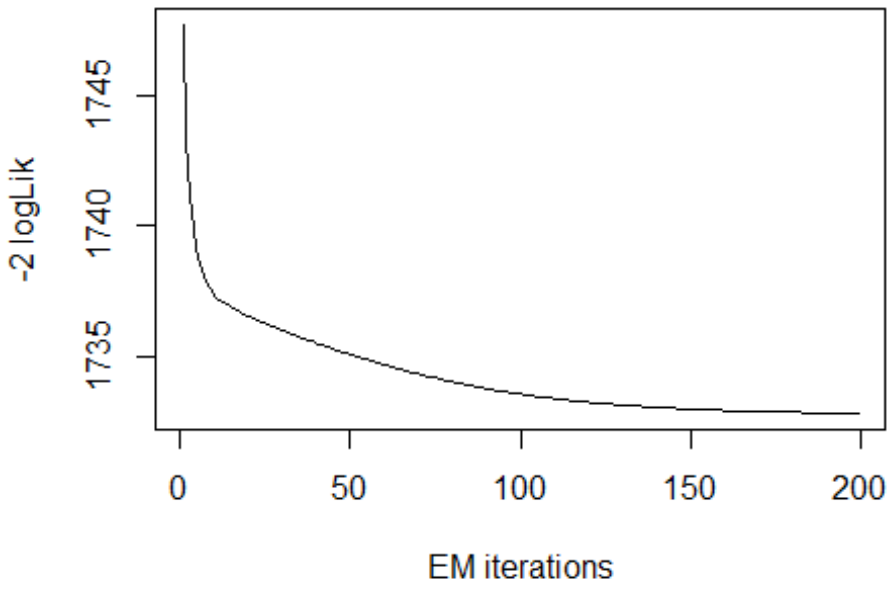
Distributions Mixture

Now, I will also apply the fitting criteria for the distributions with Gamma mixture as follows:

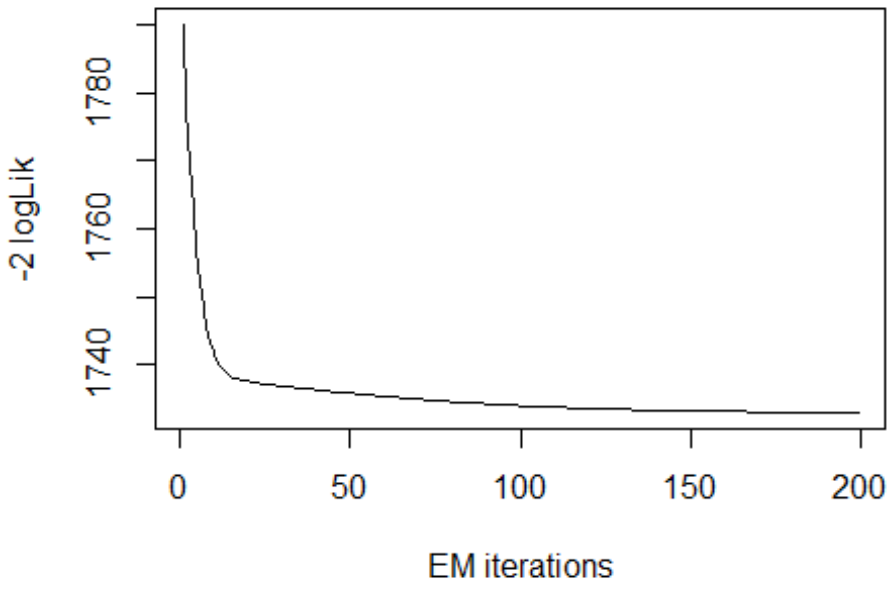
```
library(gamlss.mx)
mix.gam <- gamlssMXfits(n = 5, liver$Total_Protiens~1, family = GA, K = 2,
data = liver)
```



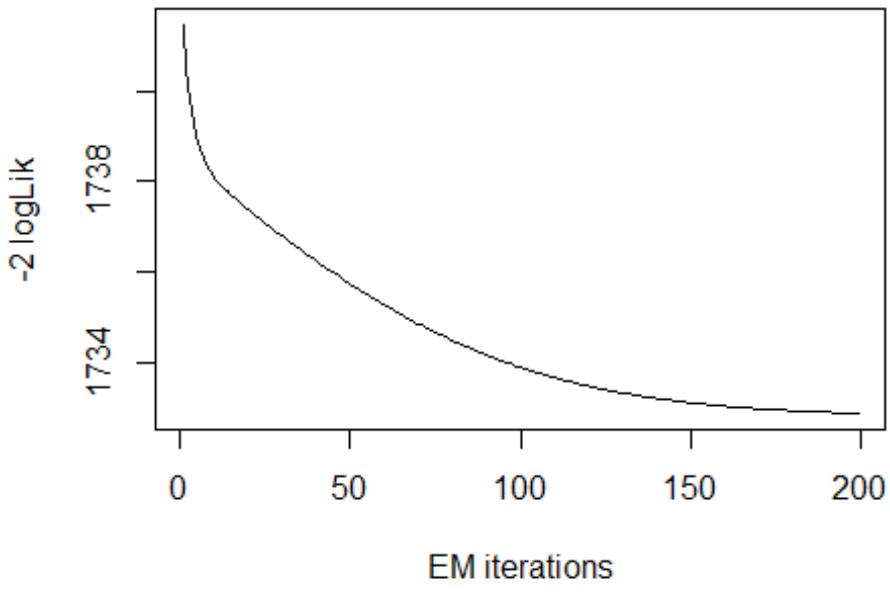

```
## model= 1
```



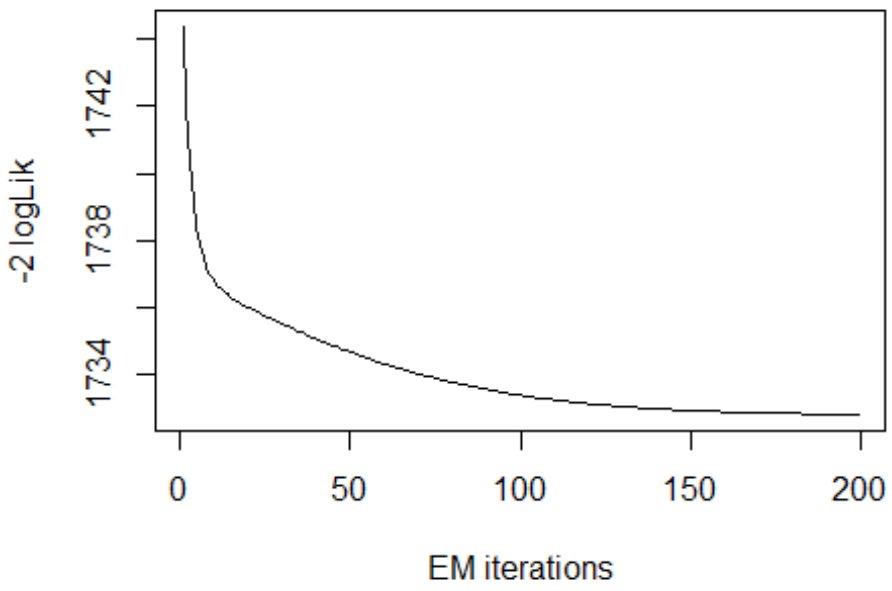

```
## model= 2
```




```
## model= 3
```




```
## model= 4
```




```
## model= 5

print(mix.gam)

##
## Mixing Family:  c("GA", "GA")
##
## Fitting method: EM algorithm
##
## Call:  gamlssMX(formula = liver$Total_Protiens ~ 1, family = GA,
##     K = 2, data = liver)
##
## Mu Coefficients for model: 1
## (Intercept)
##       1.901
## Sigma Coefficients for model: 1
## (Intercept)
##      -1.978
## Mu Coefficients for model: 2
## (Intercept)
##       1.682
## Sigma Coefficients for model: 2
## (Intercept)
##      -1.493
##
## Estimated probabilities: 0.8390968 0.1609032
##
```

```
## Degrees of Freedom for the fit: 5 Residual Deg. of Freedom   574
## Global Deviance:     1732.82
##             AIC:     1742.82
##             SBC:     1764.63
```

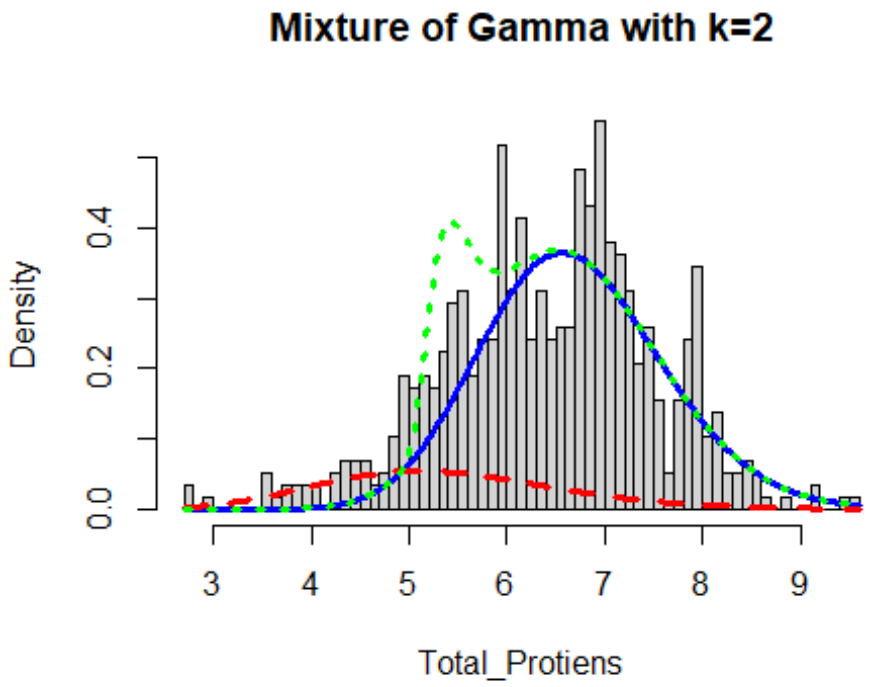
We can observe that the AIC value of the mixture of Gamma has increased, since it is higher than that of the single Gamma distribution. The current `AIC` value is `1742.81`, whereas the previous value was `1734.355` and the current `BIC` value is `1764.62`, whereas the previous value was `1747.439`.

```
logLik(mix.gam)

## 'log Lik.' -866.4106 (df=5)

mix.gam$prob

## [1] 0.8390968 0.1609032

fitted(mix.gam, "mu")[1]

## [1] 6.482017

fitted(mix.gam, "sigma")[2]

## [1] 6.482017

hist(liver$Total_Protiens, breaks = 58, xlab = "Total_Protiens",
main="Mixture of Gamma with k=2", freq = FALSE)

mu.hat1 <- exp(mix.gam[["models"]][[1]][["mu.coefficients"]])

sigma.hat1 <- exp(mix.gam[["models"]][[1]][["sigma.coefficients"]])

mu.hat2 <- exp(mix.gam[["models"]][[2]][["mu.coefficients"]])

sigma.hat2 <- exp(mix.gam[["models"]][[2]][["sigma.coefficients"]])

hist(liver$Total_Protiens, breaks = 58, freq = FALSE, plot = FALSE)

## Warning in hist.default(liver$Total_Protiens, breaks = 58, freq = FALSE, :
## argument 'freq' is not made use of

## $breaks
##  [1] 2.7 2.8 2.9 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0 4.1 4.2 4.3
4.4 4.5
## [20] 4.6 4.7 4.8 4.9 5.0 5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 6.0 6.1 6.2
6.3 6.4
## [39] 6.5 6.6 6.7 6.8 6.9 7.0 7.1 7.2 7.3 7.4 7.5 7.6 7.7 7.8 7.9 8.0 8.1
8.2 8.3
## [58] 8.4 8.5 8.6 8.7 8.8 8.9 9.0 9.1 9.2 9.3 9.4 9.5 9.6
##
## $counts
```

```
##  [1]  2  0  1  0  0  0  0  0  3  1  2  2  2  2  0  3  4  4  4  2  3  6 11
10 11
## [26] 10 13 17 18 11 14 14 30 18 24 14 18 14 15 15 28 25 32 22 21 18 12 15
9  3
## [51]  9 14 20  6  8  3  3  4  3  1  0  1  0  0  2  0  0  1  1
##
## $density
##  [1] 0.03454231 0.00000000 0.01727116 0.00000000 0.00000000 0.00000000
##  [7] 0.00000000 0.00000000 0.05181347 0.01727116 0.03454231 0.03454231
## [13] 0.03454231 0.03454231 0.00000000 0.05181347 0.06908463 0.06908463
## [19] 0.06908463 0.03454231 0.05181347 0.10362694 0.18998273 0.17271157
## [25] 0.18998273 0.17271157 0.22452504 0.29360967 0.31088083 0.18998273
## [31] 0.24179620 0.24179620 0.51813472 0.31088083 0.41450777 0.24179620
## [37] 0.31088083 0.24179620 0.25906736 0.25906736 0.48359240 0.43177893
## [43] 0.55267703 0.37996546 0.36269430 0.31088083 0.20725389 0.25906736
## [49] 0.15544041 0.05181347 0.15544041 0.24179620 0.34542314 0.10362694
## [55] 0.13816926 0.05181347 0.05181347 0.06908463 0.05181347 0.01727116
## [61] 0.00000000 0.01727116 0.00000000 0.00000000 0.03454231 0.00000000
## [67] 0.00000000 0.01727116 0.01727116
##
## $mids
##  [1] 2.75 2.85 2.95 3.05 3.15 3.25 3.35 3.45 3.55 3.65 3.75 3.85 3.95 4.05
4.15
## [16] 4.25 4.35 4.45 4.55 4.65 4.75 4.85 4.95 5.05 5.15 5.25 5.35 5.45 5.55
5.65
## [31] 5.75 5.85 5.95 6.05 6.15 6.25 6.35 6.45 6.55 6.65 6.75 6.85 6.95 7.05
7.15
## [46] 7.25 7.35 7.45 7.55 7.65 7.75 7.85 7.95 8.05 8.15 8.25 8.35 8.45 8.55
8.65
## [61] 8.75 8.85 8.95 9.05 9.15 9.25 9.35 9.45 9.55
##
## $xname
## [1] "liver$Total_Protiens"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

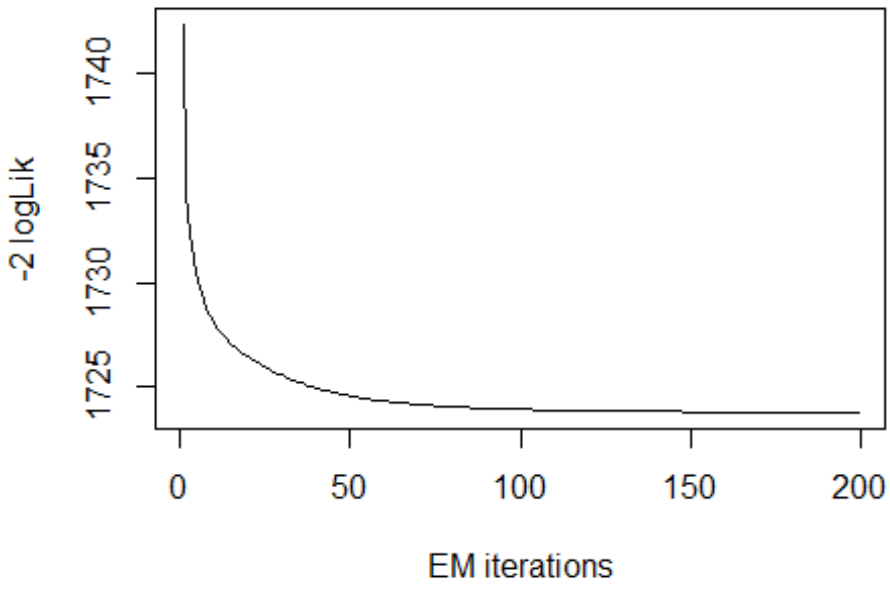
```r
lines(seq(min(liver$Total_Protiens), max(liver$Total_Protiens), length =
length(liver$Total_Protiens)),
      mix.gam[["prob"]][1]*dGA(seq(min(liver$Total_Protiens),
max(liver$Total_Protiens),
      length = length(liver$Total_Protiens)), mu = mu.hat1, sigma =
sigma.hat1),
      lty=1, lwd=3, col="blue")

lines(seq(min(liver$Total_Protiens), max(liver$Total_Protiens), length =
length(liver$Total_Protiens)),
```

```
      mix.gam[["prob"]][2]*dGA(seq(min(liver$Total_Protiens),
max(liver$Total_Protiens),
      length = length(liver$Total_Protiens)), mu = mu.hat2, sigma =
sigma.hat2),
      lty = 2, lwd = 3, col = "red")

lines(seq(min(liver$Total_Protiens), max(liver$Total_Protiens), length =
length(liver$Total_Protiens)),
      mix.gam[["prob"]][1]*dGA(seq(min(liver$Total_Protiens),
max(liver$Total_Protiens),
      length = length(liver$Total_Protiens)), mu = mu.hat1, sigma =
sigma.hat1)+
      mix.gam[["prob"]][2]*dRG(seq(min(liver$Total_Protiens),
max(liver$Total_Protiens),
      length = length(liver$Total_Protiens)), mu = mu.hat2, sigma =
sigma.hat2),
      lty = 3, lwd = 3, col = "green")
```

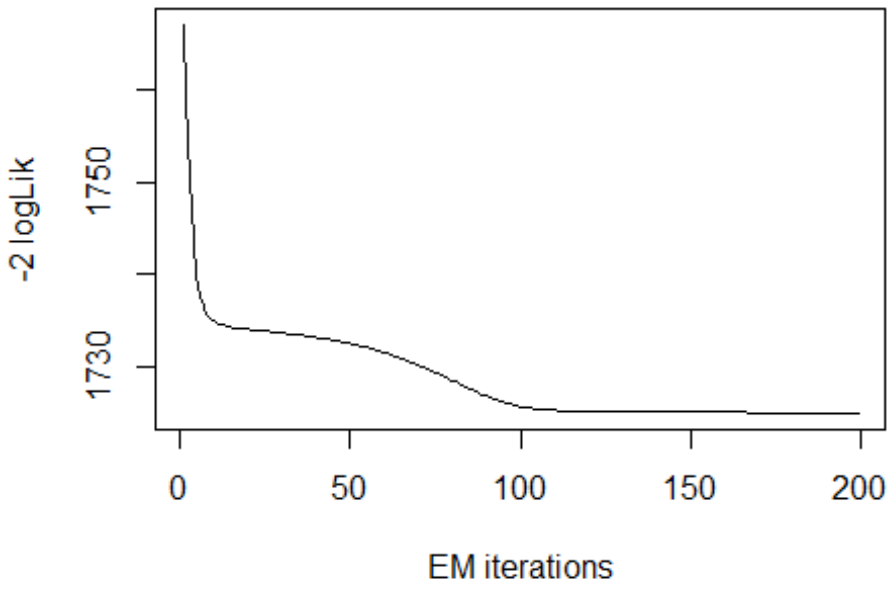


Now, I will again apply the fitting criteria for the distributions with Gamma mixture with the k = 3 as follows:

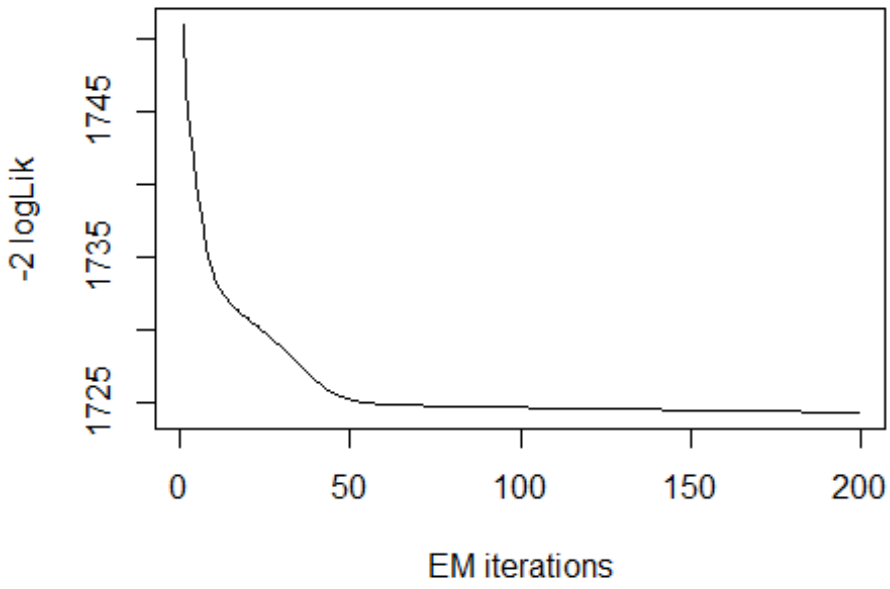
```
mix.gam.3 <- gamlssMXfits(n = 5, liver$Total_Protiens~1, family = GA, K = 3,
data = liver)
```



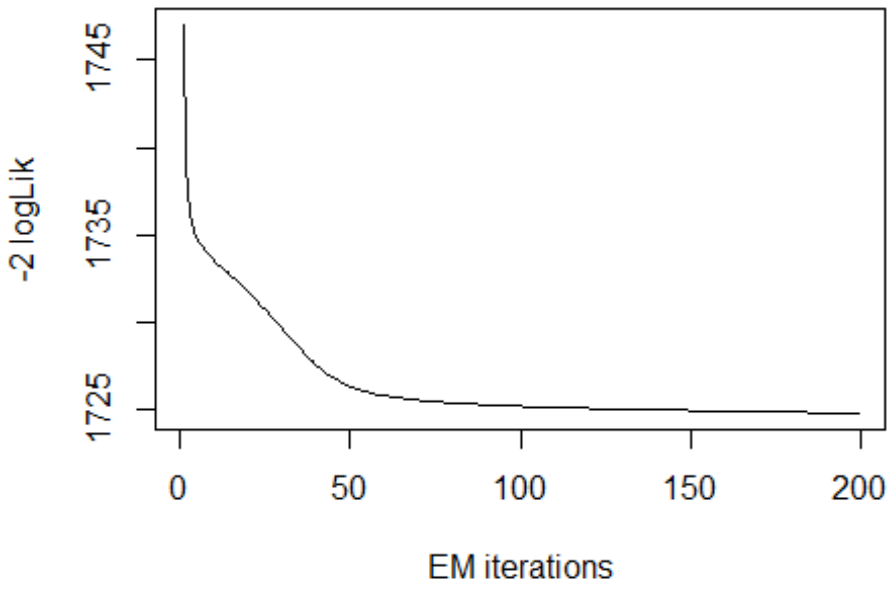

```
## model= 1
```



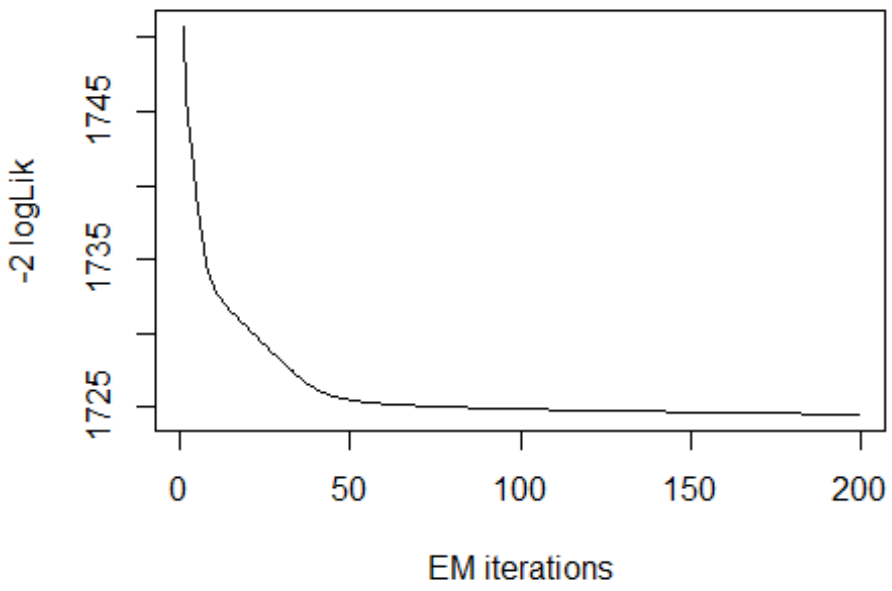

```
## model= 2
```




```
## model= 3
```




```
## model= 4
```




```
## model= 5

print(mix.gam.3)

##
## Mixing Family:  c("GA", "GA", "GA")
##
## Fitting method: EM algorithm
##
## Call:  gamlssMX(formula = liver$Total_Protiens ~ 1, family = GA,
##     K = 3, data = liver)
##
## Mu Coefficients for model: 1
## (Intercept)
##       1.735
## Sigma Coefficients for model: 1
## (Intercept)
##       -2.44
## Mu Coefficients for model: 2
## (Intercept)
##       1.953
## Sigma Coefficients for model: 2
## (Intercept)
##      -2.254
## Mu Coefficients for model: 3
## (Intercept)
##       1.689
```

```
## Sigma Coefficients for model: 3
## (Intercept)
##      -1.436
##
## Estimated probabilities: 0.2289222 0.616795 0.1542828
##
## Degrees of Freedom for the fit: 8 Residual Deg. of Freedom   571
## Global Deviance:     1723.79
##             AIC:     1739.79
##             SBC:     1774.68
```

```
logLik(mix.gam.3)
```

```
## 'log Lik.' -861.8965 (df=8)
```

```
mix.gam.3$prob
```

```
## [1] 0.2289222 0.6167950 0.1542828
```

```
fitted(mix.gam.3, "mu")[1]
```

```
## [1] 6.481988
```

```
fitted(mix.gam.3, "sigma")[2]
```

```
## [1] 6.481988
```

```
hist(liver$Total_Protiens, breaks = 58, xlab = "Total_Protiens",
main="Mixture of Gamma with k=3", freq = FALSE)

mu.hat1 <- exp(mix.gam.3[["models"]][[1]][["mu.coefficients"]])

sigma.hat1 <- exp(mix.gam.3[["models"]][[1]][["sigma.coefficients"]])

mu.hat2 <- exp(mix.gam.3[["models"]][[2]][["mu.coefficients"]])

sigma.hat2 <- exp(mix.gam.3[["models"]][[2]][["sigma.coefficients"]])

hist(liver$Total_Protiens, breaks = 58, freq = FALSE, plot = FALSE)
```

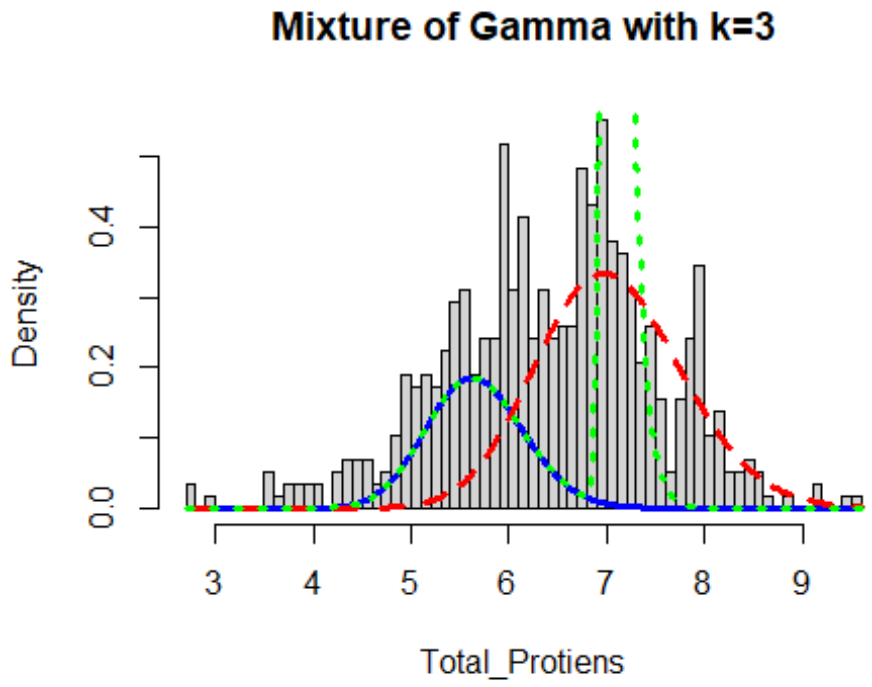
```
## Warning in hist.default(liver$Total_Protiens, breaks = 58, freq = FALSE, :
## argument 'freq' is not made use of
```

```
## $breaks
##  [1] 2.7 2.8 2.9 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0 4.1 4.2 4.3
4.4 4.5
## [20] 4.6 4.7 4.8 4.9 5.0 5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 6.0 6.1 6.2
6.3 6.4
## [39] 6.5 6.6 6.7 6.8 6.9 7.0 7.1 7.2 7.3 7.4 7.5 7.6 7.7 7.8 7.9 8.0 8.1
8.2 8.3
## [58] 8.4 8.5 8.6 8.7 8.8 8.9 9.0 9.1 9.2 9.3 9.4 9.5 9.6
##
```

```
## $counts
##  [1]  2  0  1  0  0  0  0  0  3  1  2  2  2  2  0  3  4  4  4  2  3  6 11
10 11
## [26] 10 13 17 18 11 14 14 30 18 24 14 18 14 15 15 28 25 32 22 21 18 12 15
9  3
## [51]  9 14 20  6  8  3  3  4  3  1  0  1  0  0  2  0  0  1  1
##
## $density
##  [1] 0.03454231 0.00000000 0.01727116 0.00000000 0.00000000 0.00000000
##  [7] 0.00000000 0.00000000 0.05181347 0.01727116 0.03454231 0.03454231
## [13] 0.03454231 0.03454231 0.00000000 0.05181347 0.06908463 0.06908463
## [19] 0.06908463 0.03454231 0.05181347 0.10362694 0.18998273 0.17271157
## [25] 0.18998273 0.17271157 0.22452504 0.29360967 0.31088083 0.18998273
## [31] 0.24179620 0.24179620 0.51813472 0.31088083 0.41450777 0.24179620
## [37] 0.31088083 0.24179620 0.25906736 0.25906736 0.48359240 0.43177893
## [43] 0.55267703 0.37996546 0.36269430 0.31088083 0.20725389 0.25906736
## [49] 0.15544041 0.05181347 0.15544041 0.24179620 0.34542314 0.10362694
## [55] 0.13816926 0.05181347 0.05181347 0.06908463 0.05181347 0.01727116
## [61] 0.00000000 0.01727116 0.00000000 0.00000000 0.03454231 0.00000000
## [67] 0.00000000 0.01727116 0.01727116
##
## $mids
##  [1] 2.75 2.85 2.95 3.05 3.15 3.25 3.35 3.45 3.55 3.65 3.75 3.85 3.95 4.05
4.15
## [16] 4.25 4.35 4.45 4.55 4.65 4.75 4.85 4.95 5.05 5.15 5.25 5.35 5.45 5.55
5.65
## [31] 5.75 5.85 5.95 6.05 6.15 6.25 6.35 6.45 6.55 6.65 6.75 6.85 6.95 7.05
7.15
## [46] 7.25 7.35 7.45 7.55 7.65 7.75 7.85 7.95 8.05 8.15 8.25 8.35 8.45 8.55
8.65
## [61] 8.75 8.85 8.95 9.05 9.15 9.25 9.35 9.45 9.55
##
## $xname
## [1] "liver$Total_Protiens"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```
lines(seq(min(liver$Total_Protiens), max(liver$Total_Protiens), length =
length(liver$Total_Protiens)),
      mix.gam.3[["prob"]][1]*dGA(seq(min(liver$Total_Protiens),
max(liver$Total_Protiens),
      length = length(liver$Total_Protiens)), mu = mu.hat1, sigma =
sigma.hat1),
      lty=1, lwd=3, col="blue")

lines(seq(min(liver$Total_Protiens), max(liver$Total_Protiens), length =
```

```
length(liver$Total_Protiens)),
      mix.gam.3[["prob"]][2]*dGA(seq(min(liver$Total_Protiens),
max(liver$Total_Protiens),
      length = length(liver$Total_Protiens)), mu = mu.hat2, sigma =
sigma.hat2),
      lty = 2, lwd = 3, col = "red")

lines(seq(min(liver$Total_Protiens), max(liver$Total_Protiens), length =
length(liver$Total_Protiens)),
      mix.gam.3[["prob"]][1]*dGA(seq(min(liver$Total_Protiens),
max(liver$Total_Protiens),
      length = length(liver$Total_Protiens)), mu = mu.hat1, sigma =
sigma.hat1)+
      mix.gam.3[["prob"]][2]*dRG(seq(min(liver$Total_Protiens),
max(liver$Total_Protiens),
      length = length(liver$Total_Protiens)), mu = mu.hat2, sigma =
sigma.hat2),
      lty = 3, lwd = 3, col = "green")
```




```
mix.gm.tb <- data.frame(row.names=c('Gamma Mixture, K=3','Gamma Mixture,
K=2', "Generalized Gamma"),
                        AIC=c(mix.gam.3$aic, mix.gam$aic, AIC(tp.GG)),
                        BIC=c(mix.gam.3$sbc, mix.gam$sbc, tp.GG$sbc))
mix.gm.tb

##                            AIC      BIC
## Gamma Mixture, K=3 1739.793 1774.683
```

```
## Gamma Mixture, K=2 1742.821 1764.628
## Generalized Gamma  1735.067 1748.151
```

We can observe that the AIC value of the mixture of Gamma with k=3 and k=2 has increased, since values are higher than that of the single Gamma distribution. The previous AIC value is `1735.067`, whereas the current value which is higher is `1739.865` and the previous BIC value was `1748.151`, whereas the current value which is higher is `1774.755`. Here we can clearly see that our data fits better in the single Gamma Distribution.

Albumin

Lets explore the `Albumin` variable:

```
head(liver$Albumin)

## [1] 3.3 3.2 3.3 3.4 2.4 4.4

length(liver$Albumin)

## [1] 579

table(liver$Albumin)

##
## 0.9   1 1.4 1.5 1.6 1.7 1.8 1.9   2 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9
3 3.1
##   2   1   3   3   8   3  12   7  21  14  12  12  17  24  21  23  18  29
45  27
## 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9   4 4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9
5 5.5
##  26  21  21  23  18  21  15  24  37  16  12  14   8   6   4   3   1   4
1   2
```

The table() function in R returns the absolute frequencies for each patient-specified value in the data set.

```
unique(liver$Albumin)

##  [1] 3.3 3.2 3.4 2.4 4.4 3.5 3.6 4.1 2.7 3.0 2.3 3.1 2.6 1.6 3.9 4.0 1.9
1.5 2.9
## [20] 2.0 2.2 2.8 1.8 2.5 2.1 3.7 3.8 4.3 1.7 4.2 4.5 0.9 1.4 4.7 5.5 4.9
4.6 5.0
## [39] 4.8 1.0

length(unique(liver$Albumin))

## [1] 40

min(liver$Albumin)

## [1] 0.9

max(liver$Albumin)
```

```
## [1] 5.5
```

Here the total observation of Albumin is 579 and is a continuous variable that range lies between 0.9-5.5. Now, we will see the summary of the Albumin variable, for further analysis as follows:

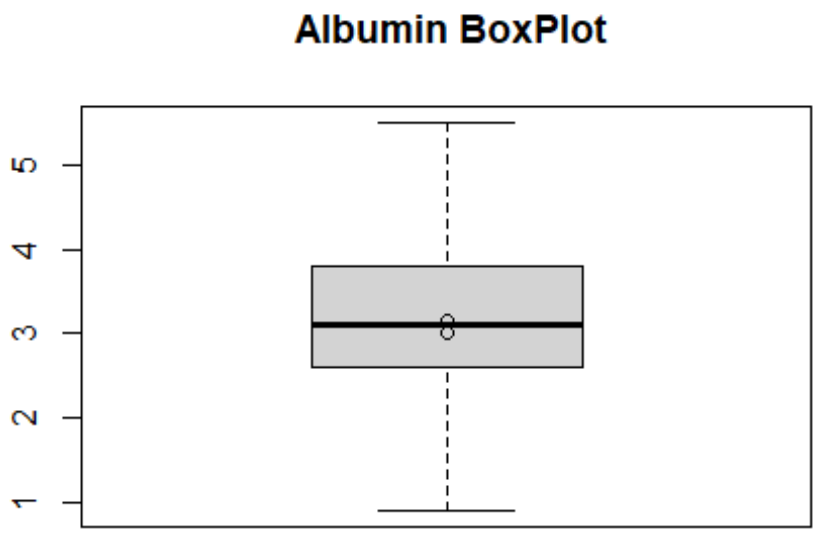
```
summary(liver$Albumin)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.900   2.600   3.100   3.139   3.800   5.500

sd(liver$Albumin)

## [1] 0.7944347

var(liver$Albumin)

## [1] 0.6311265

v <- c(liver$Albumin)
mode <- getMode(v)
print(mode)

## [1] 3
```

From the above summary we can observe that the Mean (3.139), Median (3.100) and Mode (3) are not equal so the distribution is asymmetrical. Here Mean > Median > Mode so the distribution could be positive and skewed to the right. Now we also confirm this as follows:

```
library(moments)
skewness(liver$Albumin)

## [1] -0.04839028
```

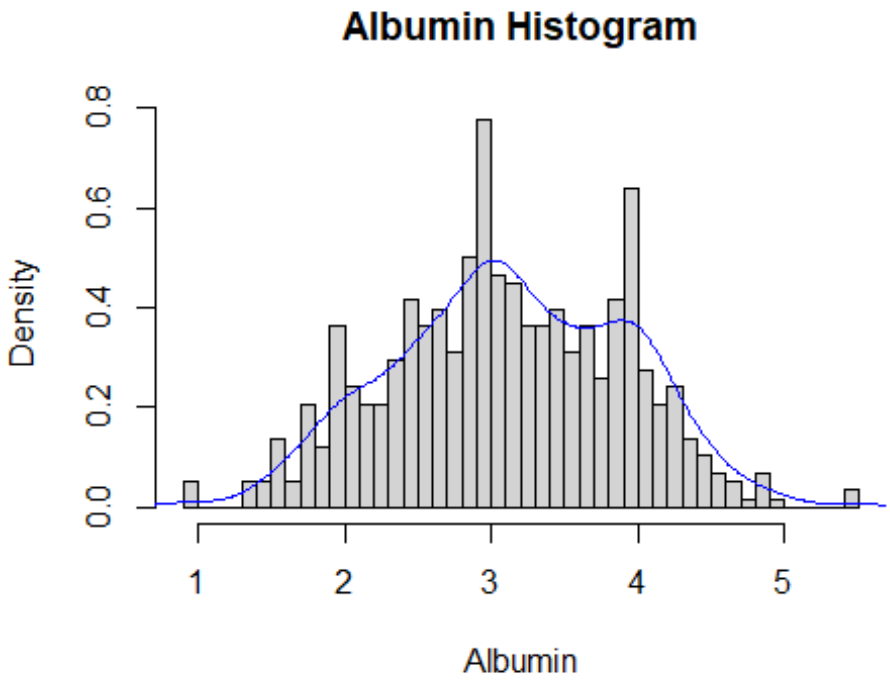
Hence, the skewness is the negative so the distribution is negatively skewed. Also I will plot the Boxplot to comparing the distribution of data across dataset as follows:

```
boxplot(liver$Albumin, main = " Albumin BoxPlot")
points(mean(liver$Albumin))
points(mode)
```



Box plots are graphical displays of the five number summary, So they describe the Minimum (lowest mean relative) lower whisker, Quartile 1 (It separates the lowest 25% observation to the rest of the observations), Median (Which divides the lower 50% observation from the upper 50% observations), Quartile 3 (It divides the upper 25% observation to the rest of the observations) and Maximum. In the above Box plot diagram, there are no outliers and the upper whisker shows the maximum. Now, I will plot a histogram a graphical display of data using bars of different heights to measures distribution of continuous data as follows:

```
hist(liver$Albumin, prob = TRUE, breaks = 40, xlab = "Albumin", main =
"Albumin Histogram")
lines(density(liver$Albumin), col='blue')
```

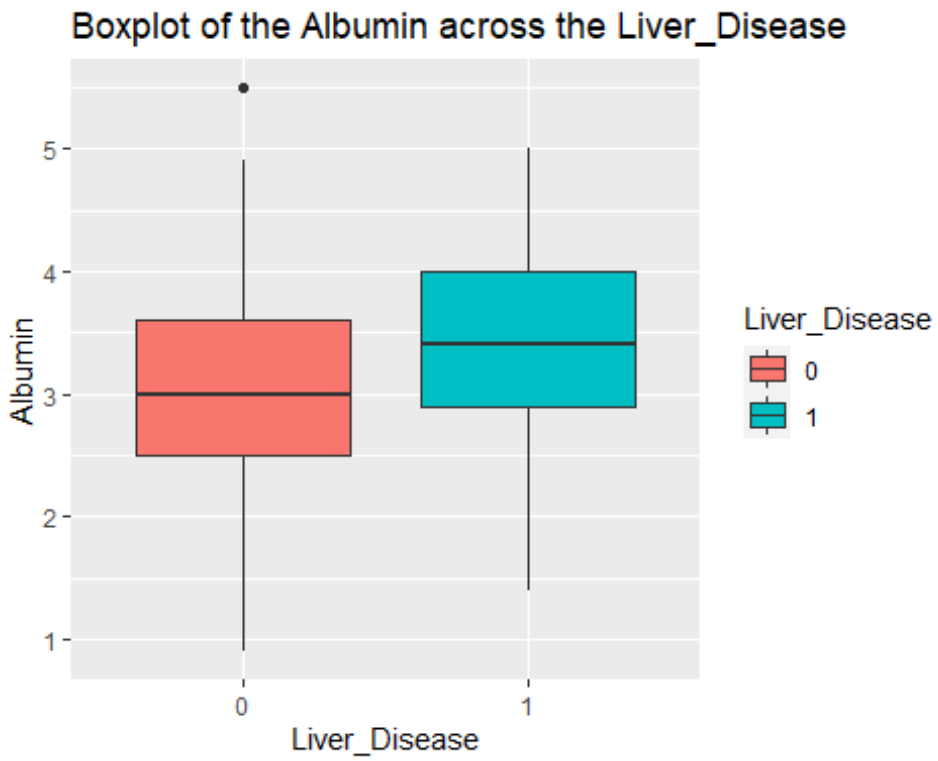


From the above histogram, we can observe that the diagram has many peaks.Using the density lines, I have approximated the histogram graph. Density provides information about the type of distribution under consideration and allows us to determine whether the attribute is distributed normally or not. From the graph above we can see how the `Albumin` variable does not seem to fit to a Normal distribution, there is a peak of the frequency distributions around many values and also a left skewed. We have already observed a negative value of `-0.04839028`, so we will affirm that the distribution is left skewed. To check whether the distribution is `Platykurtic` or not, we will check this by following R method as follows:

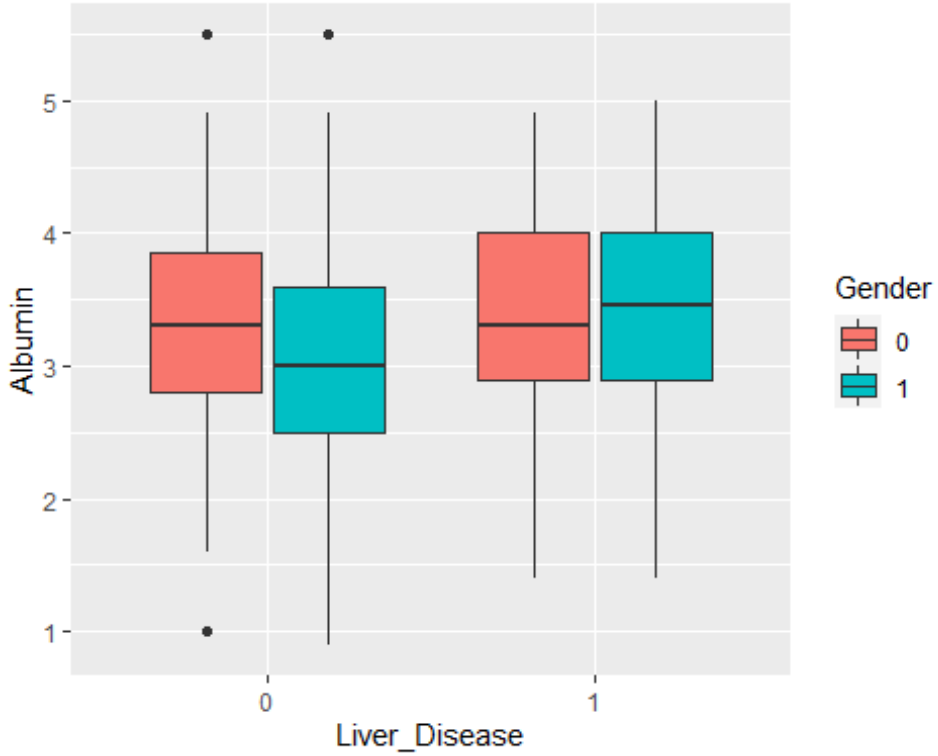
```
kurtosis(liver$Albumin)

## [1] 2.602133
```

The distribution is also `Platykurtic`, since the value is less than 3.

```
ggplot(liver, aes(x = Liver_Disease, y = Albumin, fill = Liver_Disease)) +
  geom_boxplot() +
  ylab("Albumin") +
  ggtitle("Boxplot of the Albumin across the Liver_Disease")
```




```
ggplot(liver, aes(Liver_Disease, Albumin)) +
  geom_boxplot(aes(fill = Gender))
```

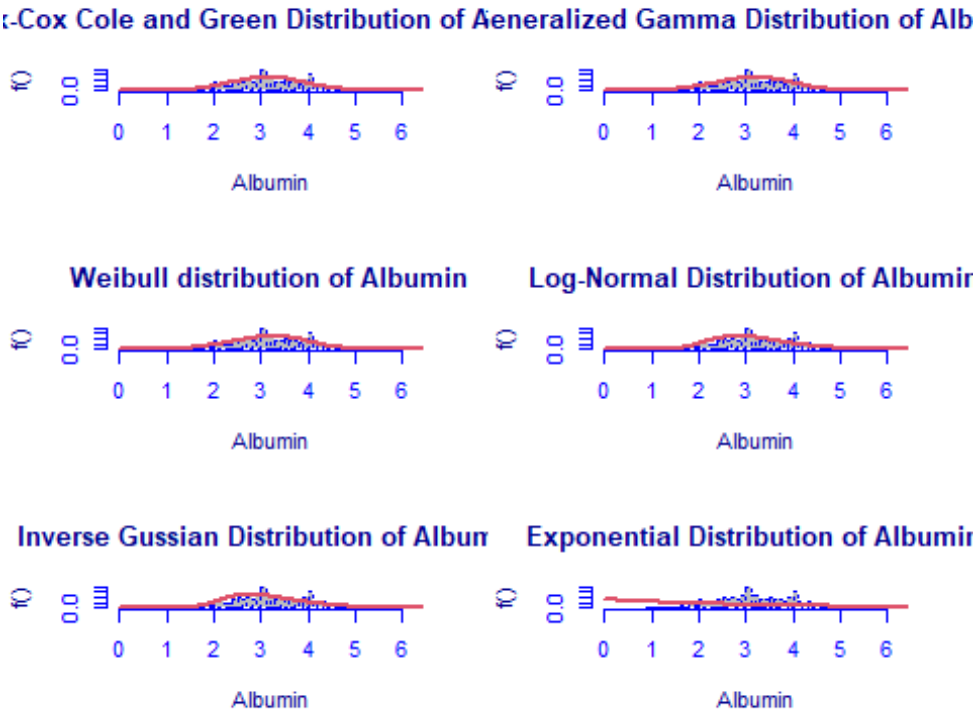


Range from 0.9 to 5.5, mean of 3.139, and median of 3.100. Bell shaped density for the Albumin values. The mean is higher for Liver_Disease = 1 and the range is larger when the Liver_Disease = 0. There is a clear difference in the means for the Albumin values across the genders and Liver_Disease. The mean is higher for females when the Liver_Disease = 0 and is higher for males when the Liver_Disease = 1.

Albumin Fit for the Data

Now we will try to fit different models to Albumin distribution evaluating the Akaike Information Criterion (AIC) and the Bayesian Information criterion (BIC), The lower are the indexes, the better is the fitting. I will evaluate both the single parameter distributions and mixture distributions as follows:

```
library(MASS)
library(gamlss)


par(mfrow=c(3,2))

a.BCCG <- histDist(liver$Albumin, family=BCCG, nbins = 40, xlab = "Albumin",
main="Box-Cox Cole and Green Distribution of Albumin")

a.GG <- histDist(liver$Albumin, family=GG, nbins = 40, xlab = "Albumin",
main="Generalized Gamma Distribution of Albumin")

a.WEI <- histDist(liver$Albumin, family=WEI, nbins = 40, xlab = "Albumin",
main="Weibull distribution of Albumin")

a.LOGNO <- histDist(liver$Albumin, family=LOGNO, nbins = 40, xlab =
"Albumin", main="Log-Normal Distribution of Albumin")

a.IG <- histDist(liver$Albumin, family=IG, nbins = 40, xlab = "Albumin", main
= "Inverse Gussian Distribution of Albumin")

a.EXP<- histDist(liver$Albumin, family=EXP, nbins = 40, xlab = "Albumin",
main = "Exponential Distribution of Albumin")
```




```
library(Matrix)
library(glmnet)
df <- data.frame(Rownames = c("Box-Cox Cole and Green","Generalized Gamma",
"Weibull",
                              "Log-Normal", "Inverse Gussian", "Exponential"),
                 AIC = c(AIC(a.BCCG), AIC(a.GG), AIC(a.WEI),
AIC(a.LOGNO),
                         AIC(a.IG), AIC(a.EXP)),
                 BIC = c(a.BCCG$sbc, a.GG$sbc, a.WEI$sbc, a.LOGNO$sbc,
                         a.IG$sbc, a.EXP$sbc),
                 df = c(a.BCCG$df.fit, a.GG$df.fit, a.WEI$df.fit,
                        a.LOGNO$df.fit, a.IG$df.fit, a.EXP$df.fit),
                 LogLike = c(logLik(a.BCCG), logLik(a.GG), logLik(a.WEI),
                             logLik(a.LOGNO), logLik(a.IG),
logLik(a.EXP)))
df
```

```
##                 Rownames      AIC      BIC df    LogLike
## 1 Box-Cox Cole and Green 1381.580 1394.663  3  -687.7898
## 2      Generalized Gamma 1377.845 1390.929  3  -685.9226
## 3                Weibull 1377.253 1385.976  2  -686.6266
## 4             Log-Normal 1442.531 1451.253  2  -719.2653
## 5        Inverse Gussian 1448.224 1456.947  2  -722.1121
## 6            Exponential 2484.462 2488.823  1 -1241.2311
```

As we can see, the model with the highest log likelihood (-685.9226) and the lowest AIC (1377.845) and BIC (1390.929) is the Generalized Gamma Distribution. Therefore, based

on the greatest likelihood technique, our data fits better in the Generalized Gamma Distribution. Now, I will also compare two models by means of the Likelihood ratio test as we performed above as follows:

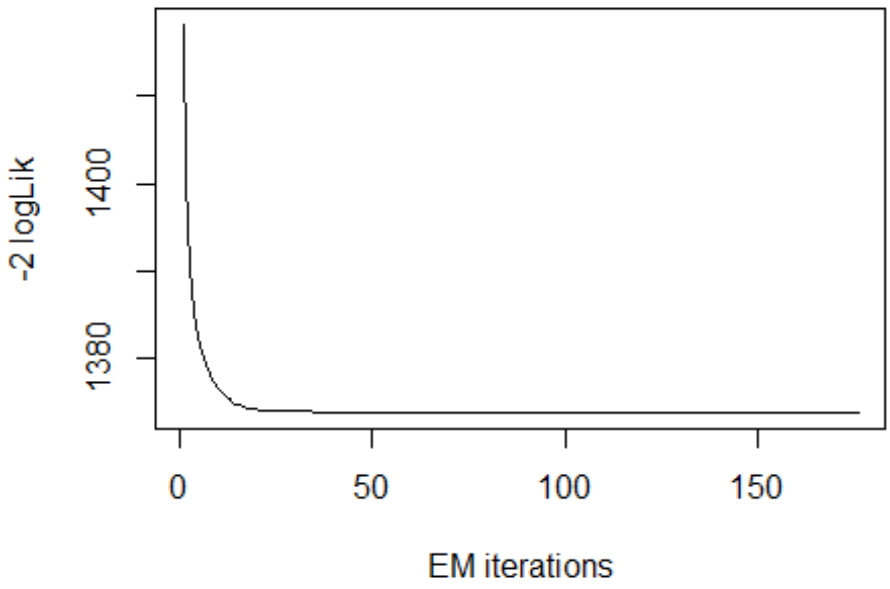
```
library(zoo)
library(lmtest)
lrtest(a.GG, a.EXP)

## Likelihood ratio test
##
## Model 1: gamlssML(formula = liver$Albumin, family = "GG")
## Model 2: gamlssML(formula = liver$Albumin, family = "EXP")
##   #Df   LogLik Df  Chisq Pr(>Chisq)
## 1   3  -685.92
## 2   1 -1241.23 -2 1110.6  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Under the null hypothesis, we compare the Generalized Gamma distribution to the Exponential distribution under the alternative hypothesis. At any level of significance, we can reject the null hypothesis. Because our distributions fit better in the Generalized Gamma model, we'll use it.

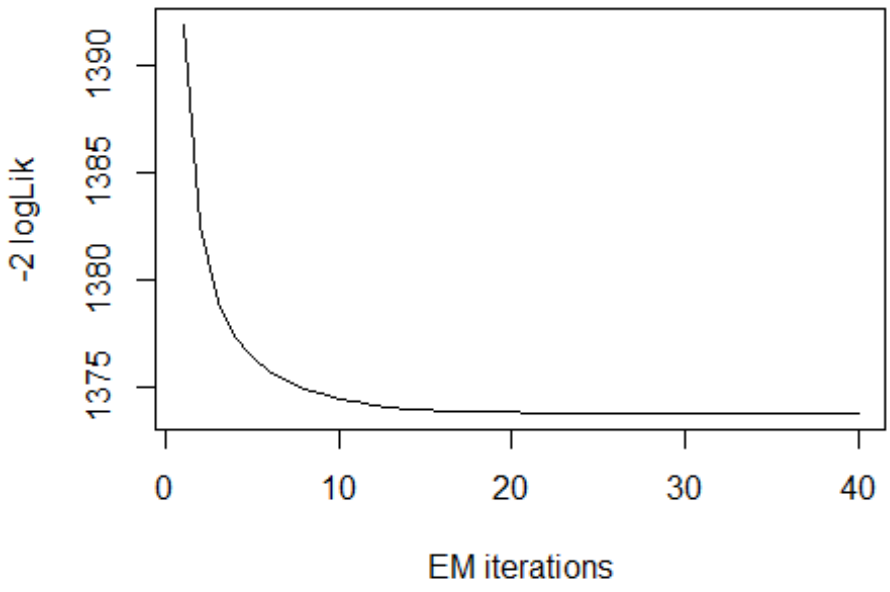
Distributions Mixture

Now, I will also apply the fitting criteria for the distributions with Gamma mixture as follows:

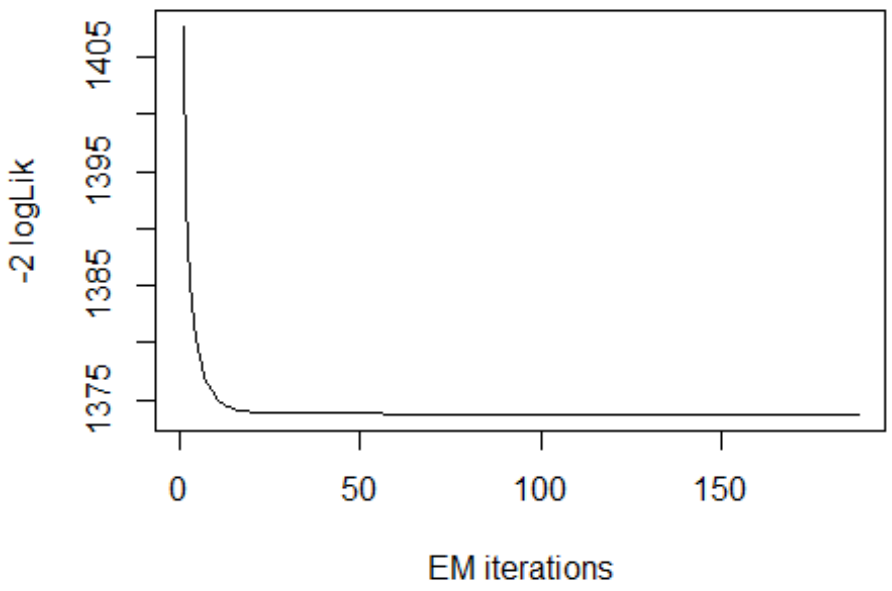
```
library(gamlss.mx)
mix.gam <- gamlssMXfits(n = 5, liver$Albumin~1, family = GA, K = 2, data =
liver)
```



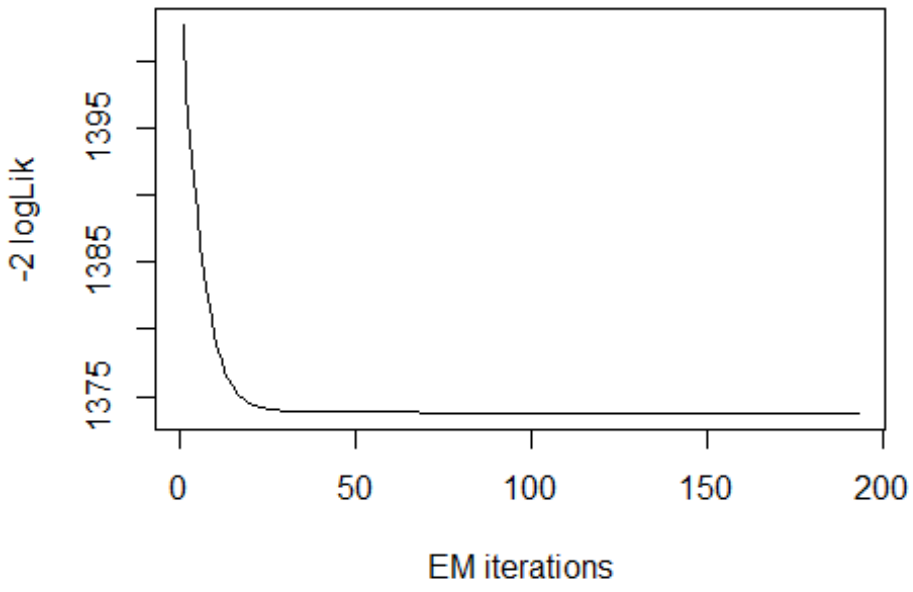

```
## model= 1
```



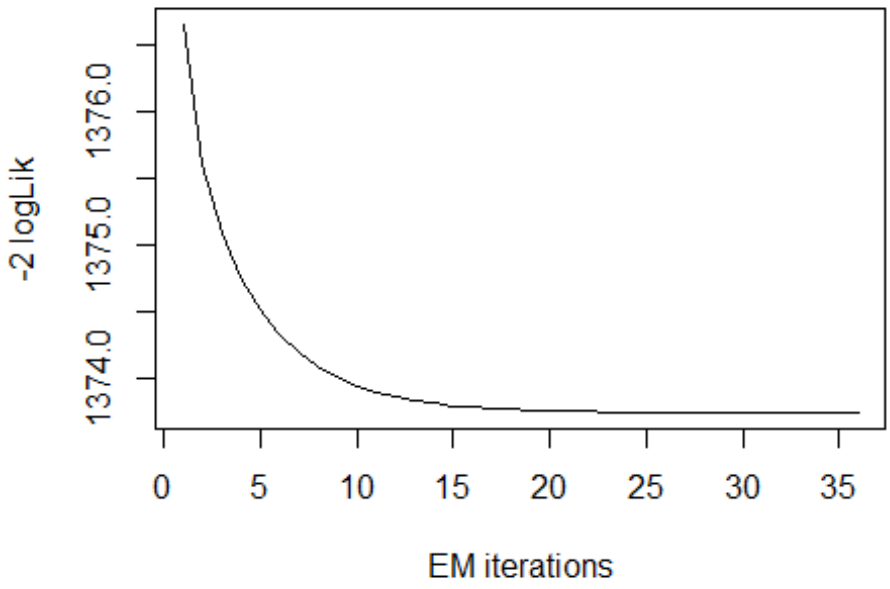

```
## model= 2
```




```
## model= 3
```




```
## model= 4
```




```
## model= 5

print(mix.gam)

##
## Mixing Family:  c("GA", "GA")
##
## Fitting method: EM algorithm
##
## Call:  gamlssMX(formula = liver$Albumin ~ 1, family = GA,
##     K = 2, data = liver)
##
## Mu Coefficients for model: 1
## (Intercept)
##       1.007
## Sigma Coefficients for model: 1
## (Intercept)
##      -1.305
## Mu Coefficients for model: 2
## (Intercept)
##       1.286
## Sigma Coefficients for model: 2
## (Intercept)
##      -1.848
##
## Estimated probabilities: 0.5431728 0.4568272
##
```

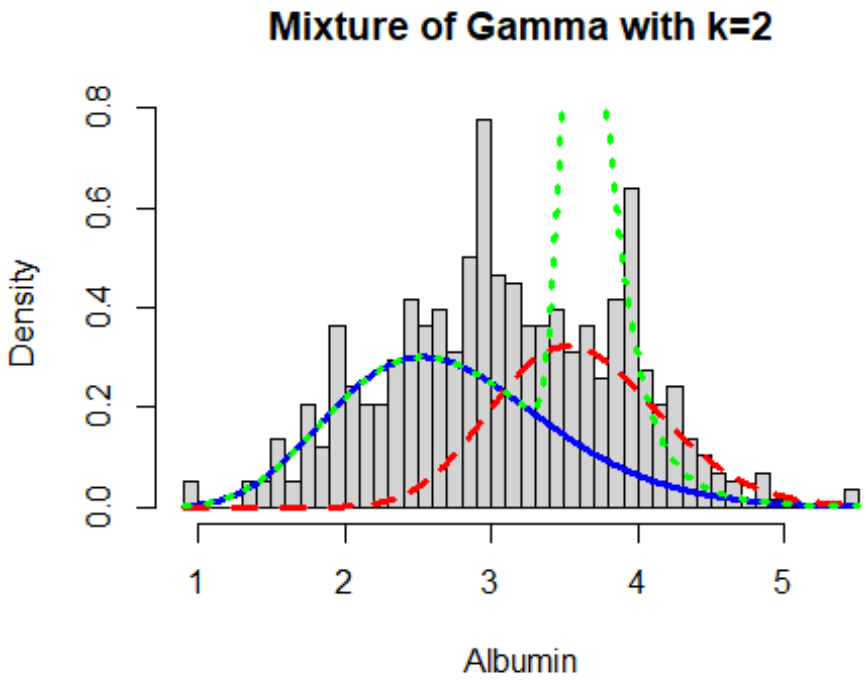
```
## Degrees of Freedom for the fit: 5 Residual Deg. of Freedom   574
## Global Deviance:     1373.75
##             AIC:     1383.75
##             SBC:     1405.55
```

We can observe that the AIC value of the mixture of Gamma has improved, since it is higher than that of the single Gamma distribution. The current `AIC` value is `1383.75`, whereas the previous value was `1377.845`and the current `BIC` value is `1405.55`, whereas the previous value was `1390.929`.

```
logLik(mix.gam)

## 'log Lik.' -686.8726 (df=5)

mix.gam$prob

## [1] 0.5431728 0.4568272

fitted(mix.gam, "mu")[1]

## [1] 3.138593

fitted(mix.gam, "sigma")[2]

## [1] 3.138593

hist(liver$Albumin, breaks = 40, xlab = "Albumin", main="Mixture of Gamma
with k=2", freq = FALSE)

mu.hat1 <- exp(mix.gam[["models"]][[1]][["mu.coefficients"]])

sigma.hat1 <- exp(mix.gam[["models"]][[1]][["sigma.coefficients"]])

mu.hat2 <- exp(mix.gam[["models"]][[2]][["mu.coefficients"]])

sigma.hat2 <- exp(mix.gam[["models"]][[2]][["sigma.coefficients"]])

hist(liver$Albumin, breaks = 40, freq = FALSE, plot = FALSE)

## Warning in hist.default(liver$Albumin, breaks = 40, freq = FALSE, plot =
FALSE):
## argument 'freq' is not made use of

## $breaks
##  [1] 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5
2.6 2.7
## [20] 2.8 2.9 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0 4.1 4.2 4.3 4.4
4.5 4.6
## [39] 4.7 4.8 4.9 5.0 5.1 5.2 5.3 5.4 5.5
##
## $counts
##  [1]  3  0  0  0  3  3  8  3 12  7 21 14 12 12 17 24 21 23 18 29 45 27 26
```

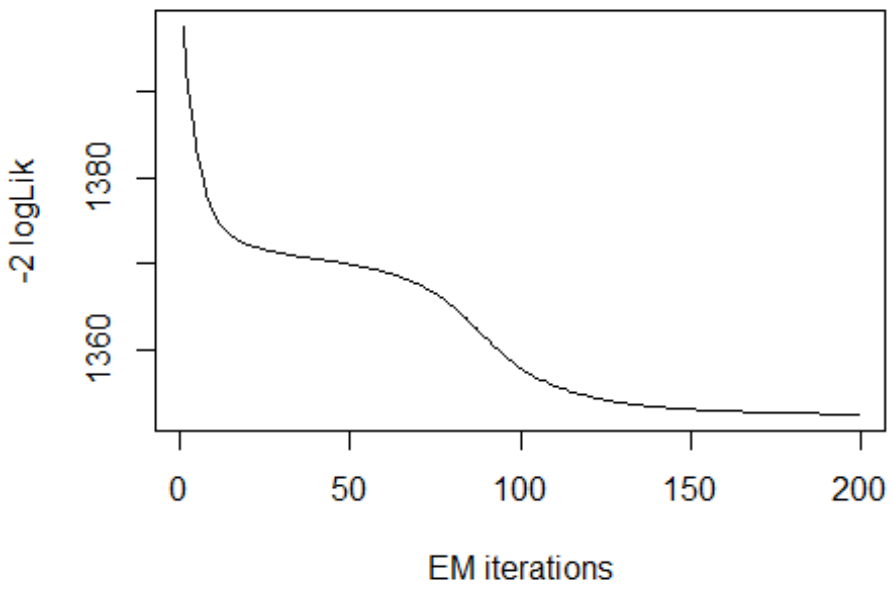
```
21 21
## [26] 23 18 21 15 24 37 16 12 14  8  6  4  3  1  4  1  0  0  0  0  2
##
## $density
##  [1] 0.05181347 0.00000000 0.00000000 0.00000000 0.05181347 0.05181347
##  [7] 0.13816926 0.05181347 0.20725389 0.12089810 0.36269430 0.24179620
## [13] 0.20725389 0.20725389 0.29360967 0.41450777 0.36269430 0.39723661
## [19] 0.31088083 0.50086356 0.77720207 0.46632124 0.44905009 0.36269430
## [25] 0.36269430 0.39723661 0.31088083 0.36269430 0.25906736 0.41450777
## [31] 0.63903282 0.27633851 0.20725389 0.24179620 0.13816926 0.10362694
## [37] 0.06908463 0.05181347 0.01727116 0.06908463 0.01727116 0.00000000
## [43] 0.00000000 0.00000000 0.00000000 0.03454231
##
## $mids
##  [1] 0.95 1.05 1.15 1.25 1.35 1.45 1.55 1.65 1.75 1.85 1.95 2.05 2.15 2.25
2.35
## [16] 2.45 2.55 2.65 2.75 2.85 2.95 3.05 3.15 3.25 3.35 3.45 3.55 3.65 3.75
3.85
## [31] 3.95 4.05 4.15 4.25 4.35 4.45 4.55 4.65 4.75 4.85 4.95 5.05 5.15 5.25
5.35
## [46] 5.45
##
## $xname
## [1] "liver$Albumin"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```
lines(seq(min(liver$Albumin), max(liver$Albumin), length =
length(liver$Albumin)),
      mix.gam[["prob"]][1]*dGA(seq(min(liver$Albumin), max(liver$Albumin),
      length = length(liver$Albumin)), mu = mu.hat1, sigma = sigma.hat1),
      lty=1, lwd=3, col="blue")

lines(seq(min(liver$Albumin), max(liver$Albumin), length =
length(liver$Albumin)),
      mix.gam[["prob"]][2]*dGA(seq(min(liver$Albumin), max(liver$Albumin),
      length = length(liver$Albumin)), mu = mu.hat2, sigma = sigma.hat2),
      lty = 2, lwd = 3, col = "red")

lines(seq(min(liver$Albumin), max(liver$Albumin), length =
length(liver$Albumin)),
      mix.gam[["prob"]][1]*dGA(seq(min(liver$Albumin), max(liver$Albumin),
      length = length(liver$Albumin)), mu = mu.hat1, sigma = sigma.hat1)+
      mix.gam[["prob"]][2]*dRG(seq(min(liver$Albumin), max(liver$Albumin),
      length = length(liver$Albumin)), mu = mu.hat2, sigma = sigma.hat2),
      lty = 3, lwd = 3, col = "green")
```

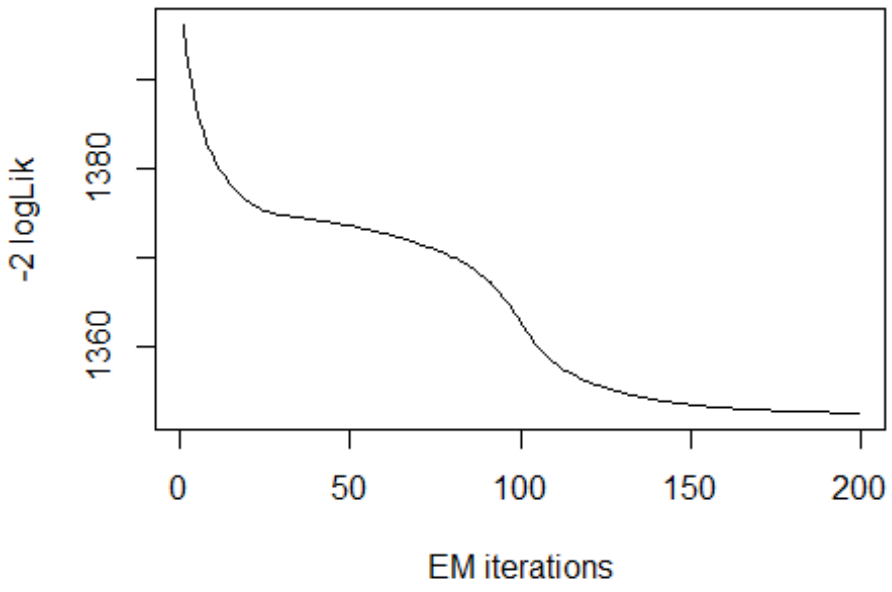


Now, I will again apply the fitting criteria for the distributions with Gamma mixture with the k = 3 as follows:

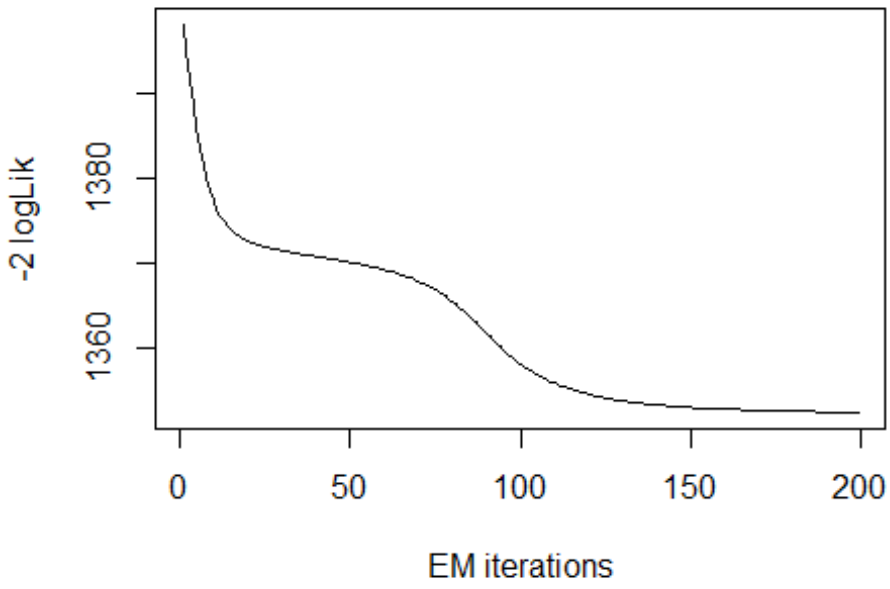
```
mix.gam.3 <- gamlssMXfits(n = 5, liver$Albumin~1, family = GA, K = 3, data =
liver)
```



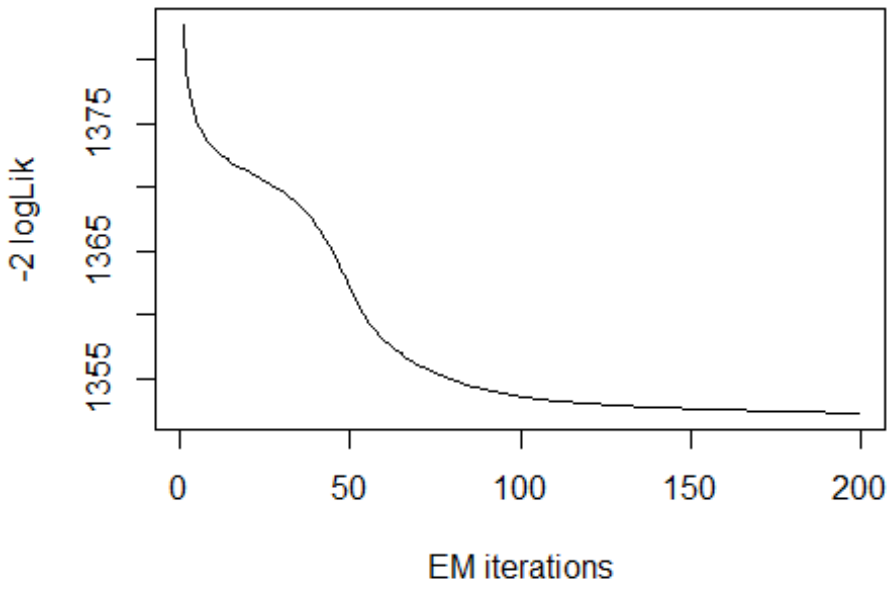

```
## model= 1
```



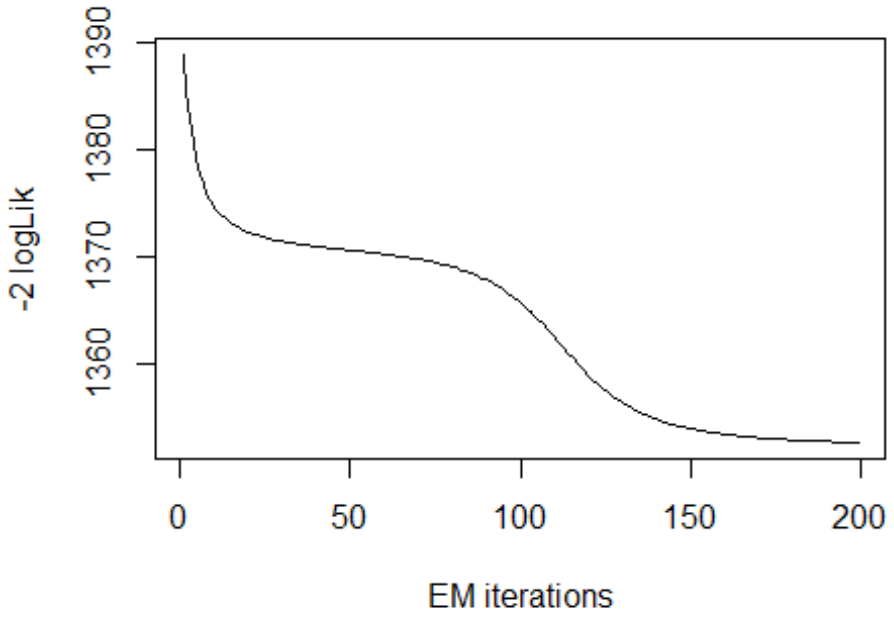

```
## model= 2
```




```
## model= 3
```




```
## model= 4
```




```
## model= 5

print(mix.gam.3)

##
## Mixing Family:  c("GA", "GA", "GA")
##
## Fitting method: EM algorithm
##
## Call:  gamlssMX(formula = liver$Albumin ~ 1, family = GA,
##     K = 3, data = liver)
##
## Mu Coefficients for model: 1
## (Intercept)
##       1.379
## Sigma Coefficients for model: 1
## (Intercept)
##      -2.594
## Mu Coefficients for model: 2
## (Intercept)
##       1.123
## Sigma Coefficients for model: 2
## (Intercept)
##      -2.659
## Mu Coefficients for model: 3
## (Intercept)
##       1.047
```

```
## Sigma Coefficients for model: 3
## (Intercept)
##      -1.263
##
## Estimated probabilities: 0.2266227 0.1564675 0.6169098
##
## Degrees of Freedom for the fit: 8 Residual Deg. of Freedom   571
## Global Deviance:     1352.27
##             AIC:     1368.27
##             SBC:     1403.17
```

```
logLik(mix.gam.3)
```

```
## 'log Lik.' -676.1375 (df=8)
```

```
mix.gam.3$prob
```

```
## [1] 0.2266227 0.1564675 0.6169098
```

```
fitted(mix.gam.3, "mu")[1]
```

```
## [1] 3.138605
```

```
fitted(mix.gam.3, "sigma")[2]
```

```
## [1] 3.138605
```

```
hist(liver$Albumin, breaks = 40, xlab = "Albumin", main="Mixture of Gamma
with k=3", freq = FALSE)
```

```
mu.hat1 <- exp(mix.gam.3[["models"]][[1]][["mu.coefficients"]])
```

```
sigma.hat1 <- exp(mix.gam.3[["models"]][[1]][["sigma.coefficients"]])
```

```
mu.hat2 <- exp(mix.gam.3[["models"]][[2]][["mu.coefficients"]])
```

```
sigma.hat2 <- exp(mix.gam.3[["models"]][[2]][["sigma.coefficients"]])
```

```
hist(liver$Albumin, breaks = 40, freq = FALSE, plot = FALSE)
```

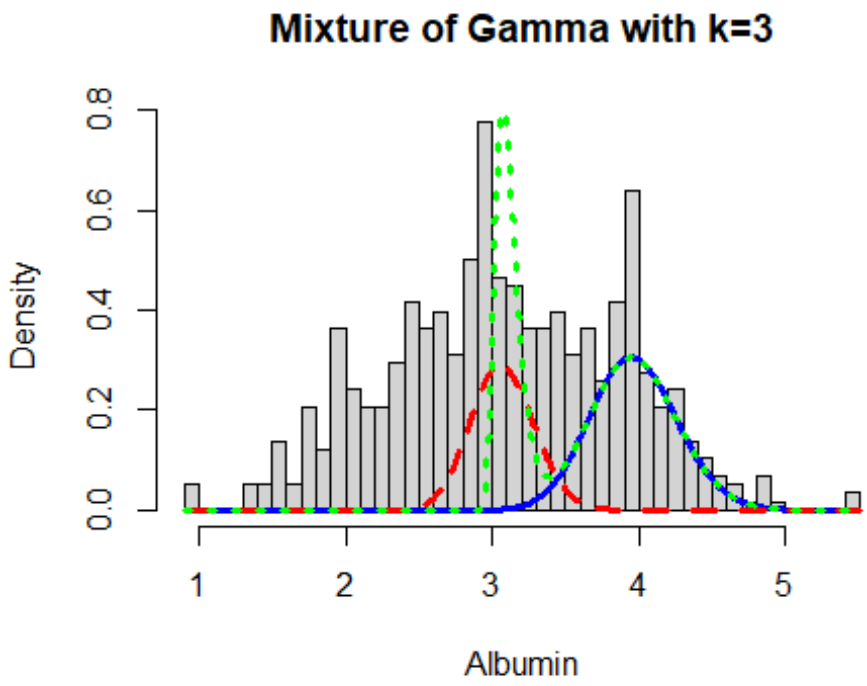
```
## Warning in hist.default(liver$Albumin, breaks = 40, freq = FALSE, plot =
FALSE):
## argument 'freq' is not made use of
```

```
## $breaks
##  [1] 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5
2.6 2.7
## [20] 2.8 2.9 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0 4.1 4.2 4.3 4.4
4.5 4.6
## [39] 4.7 4.8 4.9 5.0 5.1 5.2 5.3 5.4 5.5
##
## $counts
```

```
##  [1]  3  0  0  0  3  3  8  3 12  7 21 14 12 12 17 24 21 23 18 29 45 27 26
21 21
## [26] 23 18 21 15 24 37 16 12 14  8  6  4  3  1  4  1  0  0  0  0  2
##
## $density
##  [1] 0.05181347 0.00000000 0.00000000 0.00000000 0.05181347 0.05181347
##  [7] 0.13816926 0.05181347 0.20725389 0.12089810 0.36269430 0.24179620
## [13] 0.20725389 0.20725389 0.29360967 0.41450777 0.36269430 0.39723661
## [19] 0.31088083 0.50086356 0.77720207 0.46632124 0.44905009 0.36269430
## [25] 0.36269430 0.39723661 0.31088083 0.36269430 0.25906736 0.41450777
## [31] 0.63903282 0.27633851 0.20725389 0.24179620 0.13816926 0.10362694
## [37] 0.06908463 0.05181347 0.01727116 0.06908463 0.01727116 0.00000000
## [43] 0.00000000 0.00000000 0.00000000 0.03454231
##
## $mids
##  [1] 0.95 1.05 1.15 1.25 1.35 1.45 1.55 1.65 1.75 1.85 1.95 2.05 2.15 2.25
2.35
## [16] 2.45 2.55 2.65 2.75 2.85 2.95 3.05 3.15 3.25 3.35 3.45 3.55 3.65 3.75
3.85
## [31] 3.95 4.05 4.15 4.25 4.35 4.45 4.55 4.65 4.75 4.85 4.95 5.05 5.15 5.25
5.35
## [46] 5.45
##
## $xname
## [1] "liver$Albumin"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```r
lines(seq(min(liver$Albumin), max(liver$Albumin), length =
length(liver$Albumin)),
      mix.gam.3[["prob"]][1]*dGA(seq(min(liver$Albumin), max(liver$Albumin),
      length = length(liver$Albumin)), mu = mu.hat1, sigma = sigma.hat1),
      lty=1, lwd=3, col="blue")

lines(seq(min(liver$Albumin), max(liver$Albumin), length =
length(liver$Albumin)),
      mix.gam.3[["prob"]][2]*dGA(seq(min(liver$Albumin), max(liver$Albumin),
      length = length(liver$Albumin)), mu = mu.hat2, sigma = sigma.hat2),
      lty = 2, lwd = 3, col = "red")

lines(seq(min(liver$Albumin), max(liver$Albumin), length =
length(liver$Albumin)),
      mix.gam.3[["prob"]][1]*dGA(seq(min(liver$Albumin), max(liver$Albumin),
      length = length(liver$Albumin)), mu = mu.hat1, sigma = sigma.hat1)+
      mix.gam.3[["prob"]][2]*dRG(seq(min(liver$Albumin), max(liver$Albumin),
```

```
      length = length(liver$Albumin)), mu = mu.hat2, sigma = sigma.hat2),
      lty = 3, lwd = 3, col = "green")
```




```
mix.gm.tb <- data.frame(row.names=c('Gamma Mixture, K=3','Gamma Mixture,
K=2', "Generalized Gamma"),
                        AIC=c(mix.gam.3$aic, mix.gam$aic, AIC(a.GG)),
                        BIC=c(mix.gam.3$sbc, mix.gam$sbc, a.GG$sbc))
mix.gm.tb
```

```
##                           AIC      BIC
## Gamma Mixture, K=3 1368.275 1403.165
## Gamma Mixture, K=2 1383.745 1405.552
## Generalized Gamma  1377.845 1390.929
```

We can observe that the AIC value of the mixture of Gamma with k=3 has increased, since values are higher than that of the single Gamma distribution and Gamma Mixture k=2 . The previous AIC value is `1377.845`, whereas the current value which is lower is `1368.254`. Here we can clearly see that our data fits better in the Gamma Mixture with k=3 Distribution.

Albumin and Globulin Ratio

Lets explore the `Albumin and Globulin Ratio` variable:

```
head(liver$Albumin_and_Globulin_Ratio)
```

```
## [1] 0.90 0.74 0.89 1.00 0.40 1.30
```

```r
length(liver$Albumin_and_Globulin_Ratio)
```

```
## [1] 579
```

```r
table(liver$Albumin_and_Globulin_Ratio)
```

```
##
##  0.3 0.35 0.37 0.39  0.4 0.45 0.46 0.47 0.48  0.5 0.52 0.53 0.55 0.58  0.6
0.61
##    4    1    1    1   14    1    1    2    1   29    2    1    1    1   31
1
## 0.62 0.64 0.67 0.68 0.69  0.7 0.71 0.74 0.75 0.76 0.78  0.8 0.87 0.88 0.89
0.9
##    1    1    1    1    1   53    1    1    4    2    1   65    1    1    1
59
## 0.92 0.93 0.95 0.96 0.97    1 1.02 1.03 1.06 1.09  1.1 1.11 1.12 1.16 1.18
1.2
##    2    2    2    3    1  106    1    1    2    1   46    1    1    2    2
35
## 1.25 1.27  1.3 1.34 1.36 1.38 1.39  1.4  1.5 1.51 1.55 1.58  1.6 1.66  1.7
1.72
##    1    1   25    2    1    3    1   17   10    1    1    2    5    1    4
1
##  1.8 1.85  1.9  2.5  2.8
##    3    2    1    2    1
```

The table() function in R returns the absolute frequencies for each patient-specified value in the data set.

```r
unique(liver$Albumin_and_Globulin_Ratio)
```

```
##  [1] 0.90 0.74 0.89 1.00 0.40 1.30 1.10 1.20 0.80 0.60 0.87 0.70 0.92 0.55
0.50
## [16] 1.85 0.95 1.40 1.18 0.61 1.34 1.39 1.60 1.58 1.25 0.78 0.76 1.55 0.71
0.62
## [31] 0.67 0.75 1.16 1.50 1.66 0.96 1.38 0.52 0.47 0.93 0.48 0.58 0.69 1.27
1.12
## [46] 1.06 0.53 1.03 0.68 1.90 1.70 1.80 0.30 0.97 0.35 1.51 0.64 0.45 1.36
0.88
## [61] 1.09 1.11 1.72 2.80 0.46 0.39 1.02 2.50 0.37
```

```r
length(unique(liver$Albumin_and_Globulin_Ratio))
```

```
## [1] 69
```

```r
min(liver$Albumin_and_Globulin_Ratio)
```

```
## [1] 0.3
```

```r
max(liver$Albumin_and_Globulin_Ratio)
```

```
## [1] 2.8
```

Here the total observation of Albumin_and_Globulin_Ratio is 579 and is a continuous variable that range lies between 0.3-2.8. Now, we will see the summary of the Albumin_and_Globulin_Ratio variable, for further analysis as follows:

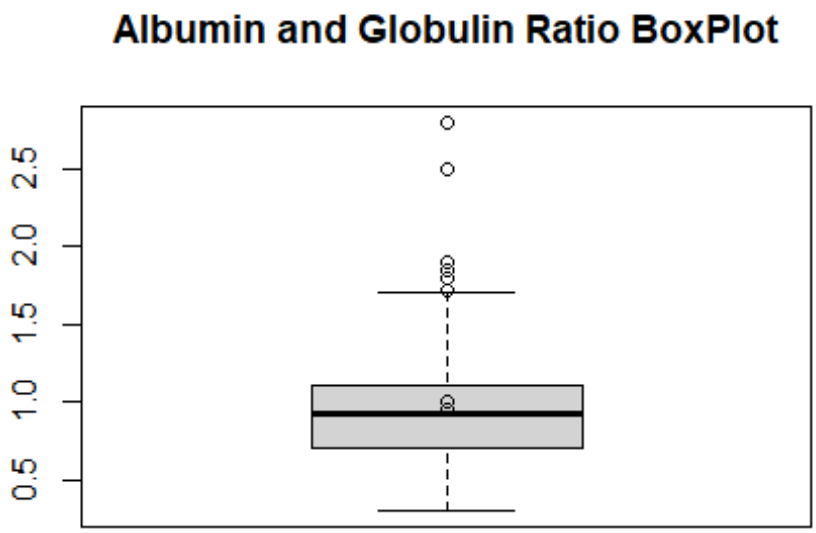
```
summary(liver$Albumin_and_Globulin_Ratio)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3000  0.7000  0.9300  0.9471  1.1000  2.8000

sd(liver$Albumin_and_Globulin_Ratio)

## [1] 0.3195921

var(liver$Albumin_and_Globulin_Ratio)

## [1] 0.1021391

v <- c(liver$Albumin_and_Globulin_Ratio)
mode <- getMode(v)
print(mode)

## [1] 1
```

From the above summary we can observe that the Mean (0.9471), Median (0.9300) and Mode (1) are not equal so the distribution is asymmetrical. Here Mean > Median > Mode so the distribution could be positive and skewed to the right. Now we also confirm this as follows:

```
library(moments)
skewness(liver$Albumin_and_Globulin_Ratio)

## [1] 0.9897269
```

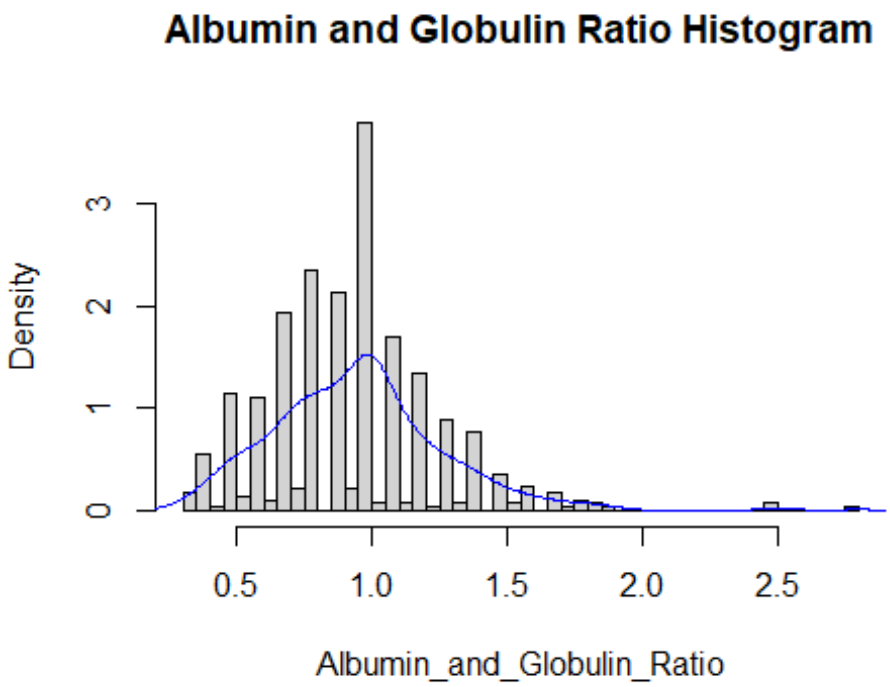
Hence, the skewness is the positive so the distribution is positively skewed was rightly observed. Also I will plot the Boxplot to comparing the distribution of data across dataset as follows:

```
boxplot(liver$Albumin_and_Globulin_Ratio, main = "Albumin and Globulin Ratio
BoxPlot")
points(mean(liver$Albumin_and_Globulin_Ratio))
points(mode)
```



Box plots are graphical displays of the five number summary, So they describe the Minimum (lowest mean relative) lower whisker, Quartile 1 (It separates the lowest 25% observation to the rest of the observations), Median (Which divides the lower 50% observation from the upper 50% observations), Quartile 3 (It divides the upper 25% observation to the rest of the observations) and Maximum. In the above Boxplot diagram, there are some outliers and the upper whisker shows the highest mean relative. Now, I will plot a histogram a graphical display of data using bars of different heights to measures distribution of continuous data as follows:

```
hist(liver$Albumin_and_Globulin_Ratio, prob = TRUE, breaks = 69, xlab =
"Albumin_and_Globulin_Ratio", main = "Albumin and Globulin Ratio Histogram")
lines(density(liver$Albumin_and_Globulin_Ratio), col='blue')
```



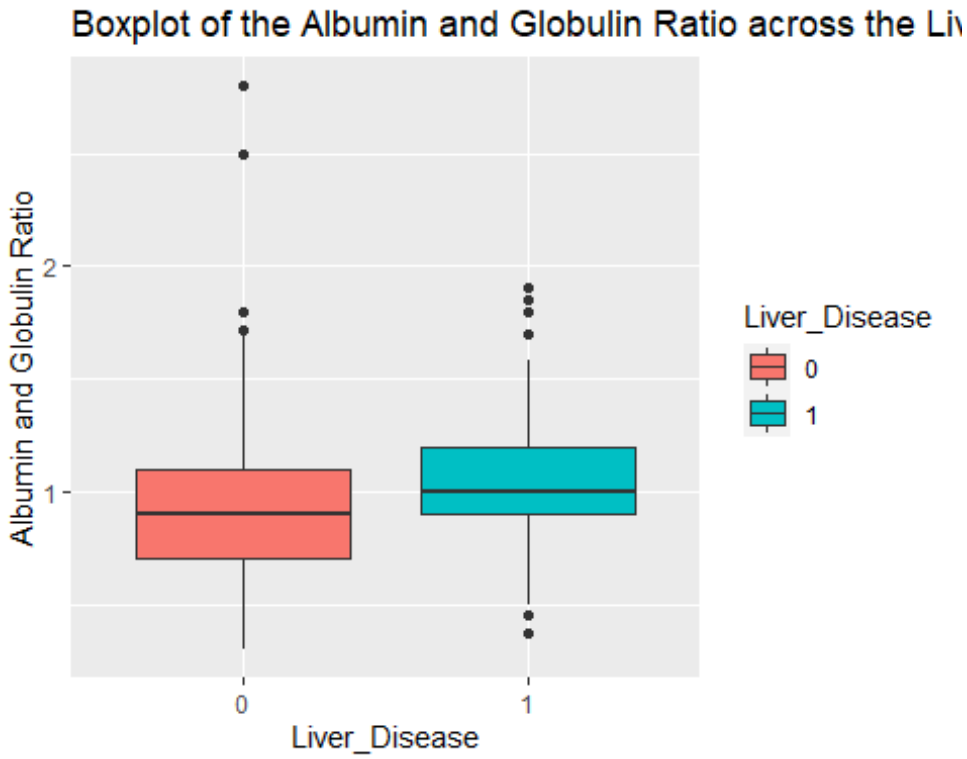
From the above histogram, we can observe that the diagram has many peaks.Using the density lines, I have approximated the histogram graph. Density provides information about the type of distribution under consideration and allows us to determine whether the attribute is distributed normally or not. From the graph above we can see how the `Albumin_and_Globulin_Ratio` variable does not seem to fit to a Normal distribution, there is a peak of the frequency distributions around many values and also a right skewed. We have already observed a positive value of `0.9897269`, so we will affirm that the distribution is right skewed. To check whether the distribution is `Platykurtic` or not, we will check this by following R method as follows:

```
kurtosis(liver$Albumin_and_Globulin_Ratio)
```

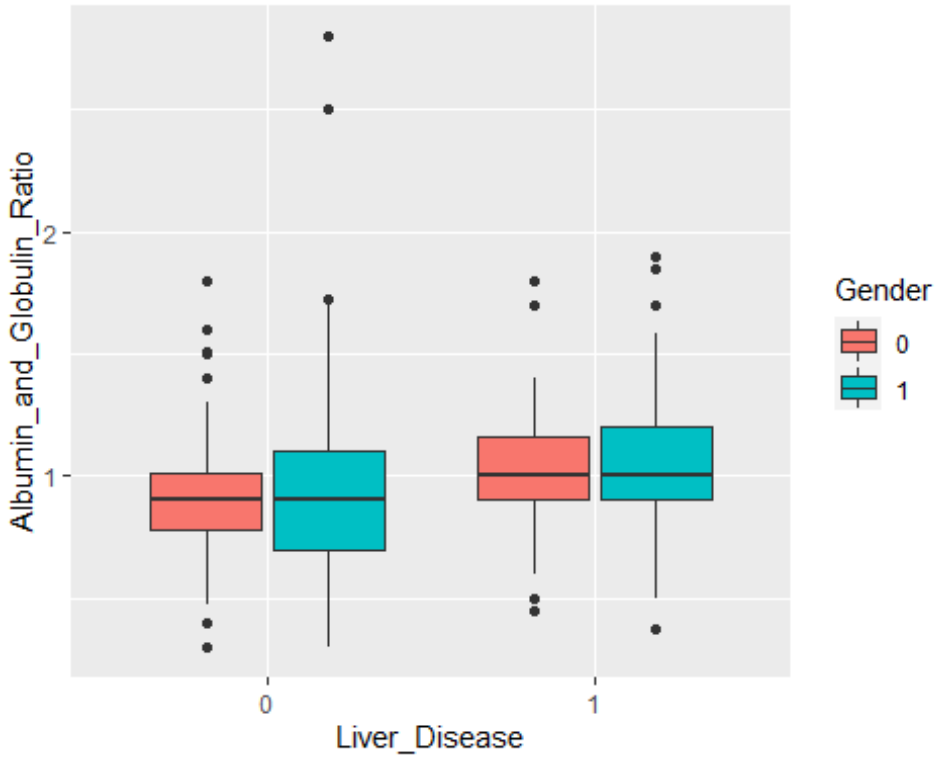
```
## [1] 6.243282
```

The distribution is also `Leptokurtic`, since the value is greater than 3.

```
ggplot(liver, aes(x = Liver_Disease, y = Albumin_and_Globulin_Ratio,
                  fill = Liver_Disease)) +
  geom_boxplot() +
  ylab("Albumin and Globulin Ratio") +
  ggtitle("Boxplot of the Albumin and Globulin Ratio across the
Liver_Disease")
```




```
ggplot(liver, aes(Liver_Disease, Albumin_and_Globulin_Ratio)) +
  geom_boxplot(aes(fill = Gender))
```

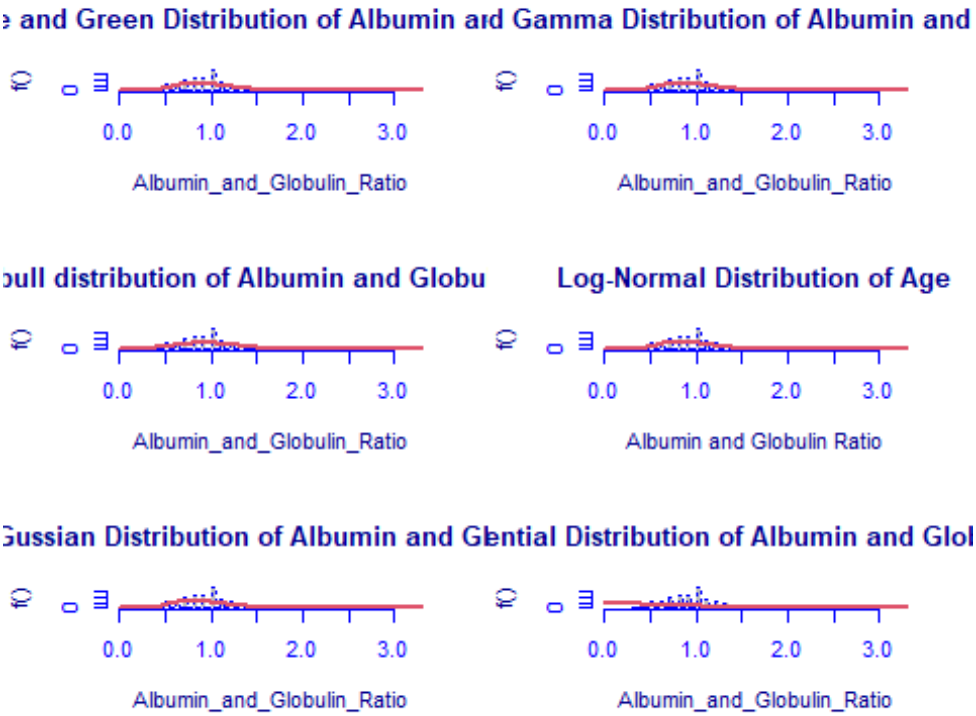


The mean for the ratio is 0.947 and is very close to the median of 0.93. The ratio has somewhat of a bell shaped curve. The mean for the ratio is higher in the Liver_Disease = 1 response, with the range being larger in the Liver_Disease = 0 group. There is no difference with the Liver_Disease when you compare the genders. The ranges vary though.

Albumin and Globulin Ratio Fit for the Data

Now we will try to fit different models to Albumin_and_Globulin_Ratio distribution evaluating the Akaike Information Criterion (AIC) and the Bayesian Information criterion (BIC), The lower are the indexes, the better is the fitting. I will evaluate both the single parameter distributions and mixture distributions as follows:

```
library(MASS)
library(gamlss)


par(mfrow=c(3,2))

agr.BCCG <- histDist(liver$Albumin_and_Globulin_Ratio, family=BCCG, nbins =
69, xlab = "Albumin_and_Globulin_Ratio", main="Box-Cox Cole and Green
Distribution of Albumin and Globulin Ratio")

agr.GG <- histDist(liver$Albumin_and_Globulin_Ratio, family=GG, nbins = 69,
xlab = "Albumin_and_Globulin_Ratio", main="Generalized Gamma Distribution of
Albumin and Globulin Ratio")

agr.WEI <- histDist(liver$Albumin_and_Globulin_Ratio, family=WEI, nbins = 69,
xlab = "Albumin_and_Globulin_Ratio", main="Weibull distribution of Albumin
and Globulin Ratio")

agr.LOGNO <- histDist(liver$Albumin_and_Globulin_Ratio, family=LOGNO, nbins =
69, xlab = "Albumin and Globulin Ratio", main="Log-Normal Distribution of
Age")

agr.IG <- histDist(liver$Albumin_and_Globulin_Ratio, family=IG, nbins = 69,
xlab = "Albumin_and_Globulin_Ratio", main = "Inverse Gussian Distribution of
Albumin and Globulin Ratio")

agr.EXP<- histDist(liver$Albumin_and_Globulin_Ratio, family=EXP, nbins = 69,
xlab = "Albumin_and_Globulin_Ratio", main = "Exponential Distribution of
Albumin and Globulin Ratio")
```




```
library(Matrix)
library(glmnet)
df <- data.frame(Rownames = c("Box-Cox Cole and Green","Generalized Gamma",
"Weibull",
                              "Log-Normal", "Inverse Gussian", "Exponential"),
                 AIC = c(AIC(agr.BCCG), AIC(agr.GG), AIC(agr.WEI),
AIC(agr.LOGNO),
                         AIC(agr.IG), AIC(agr.EXP)),
                 BIC = c(agr.BCCG$sbc, agr.GG$sbc, agr.WEI$sbc,
agr.LOGNO$sbc,
                         agr.IG$sbc, agr.EXP$sbc),
                 df = c(agr.BCCG$df.fit, agr.GG$df.fit, agr.WEI$df.fit,
                        agr.LOGNO$df.fit, agr.IG$df.fit, agr.EXP$df.fit),
                 LogLike = c(logLik(agr.BCCG), logLik(agr.GG),
logLik(agr.WEI),
                             logLik(agr.LOGNO), logLik(agr.IG),
logLik(agr.EXP)))
df
```

```
##                 Rownames       AIC       BIC df   LogLike
## 1 Box-Cox Cole and Green  274.4812  287.5651  3 -134.2406
## 2      Generalized Gamma  275.0929  288.1769  3 -134.5465
## 3                Weibull  331.1582  339.8808  2 -163.5791
## 4             Log-Normal  286.9738  295.6965  2 -141.4869
## 5        Inverse Gussian  291.8929  300.6155  2 -143.9464
## 6            Exponential 1097.0179 1101.3792  1 -547.5089
```

As we can see, the model with the highest log likelihood (-134.2406) and the lowest AIC (274.4812) and BIC (287.5651) is the Box-Cox Cole and Green Distribution. Therefore, based on the greatest likelihood technique, our data fits better in the Box-Cox Cole and Green Distribution. Now, I will also compare two models by means of the Likelihood ratio test as we performed above as follows:

```
library(zoo)
library(lmtest)
lrtest(agr.BCCG, agr.EXP)

## Likelihood ratio test
##
## Model 1: gamlssML(formula = liver$Albumin_and_Globulin_Ratio, family =
"BCCG")
## Model 2: gamlssML(formula = liver$Albumin_and_Globulin_Ratio, family =
"EXP")
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   3 -134.24
## 2   1 -547.51 -2 826.54  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Under the null hypothesis, we compare the Box-Cox Cole and Green distribution to the Exponential distribution under the alternative hypothesis. At any level of significance, we can reject the null hypothesis. Because our distributions fit better in the Box-Cox Cole and Green distribution model, we'll use it.

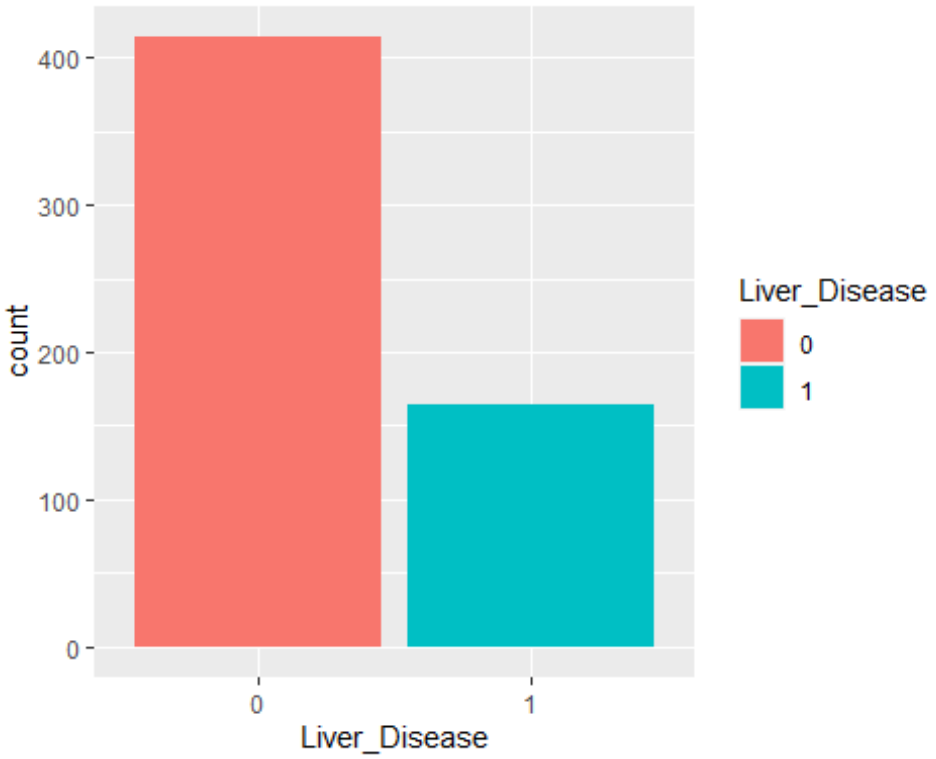
Liver Disease

Lets analyze the features of `Liver Disease` variable as follows:

```
length(liver$Liver_Disease)

## [1] 579

table(liver$Liver_Disease)

##
##   0   1
## 414 165
```

The table() function in R returns the absolute frequencies for each patient-specified value in the data set.Here we can see the `liver$Liver_Disease` variable is a categorical variable that can take two values `0` which means no disease, `1` which means have liver disease.

```
library(ggplot2)
ggplot(liver, aes(x = Liver_Disease, fill = Liver_Disease)) +
geom_bar()
```

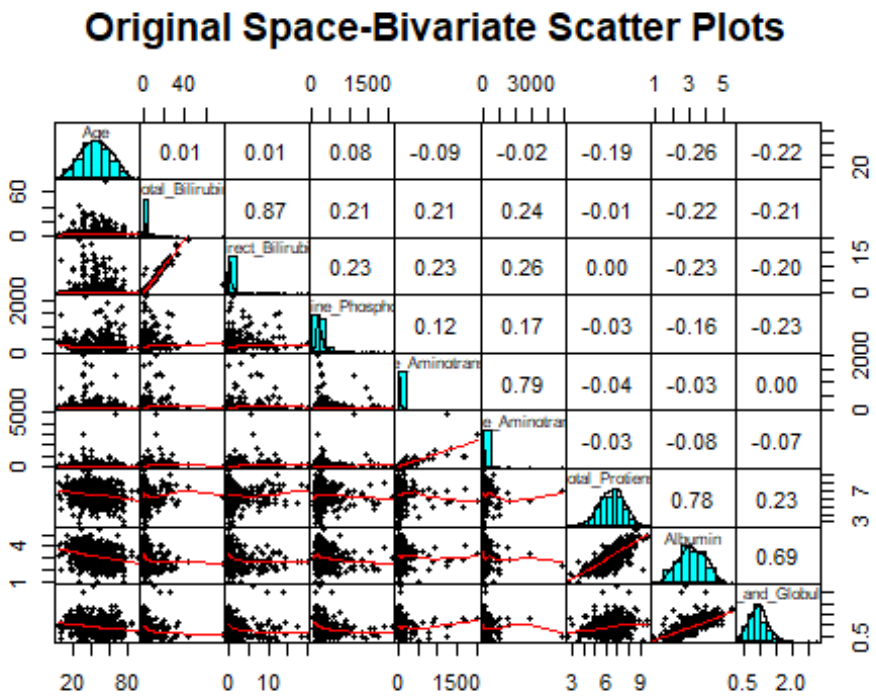


As the bar plot shows our data set contains data belonging to 414 patients have no Liver Disease and 165 patients having a Liver Disease. So from both plot and table() we can see that our data is not balanced and the study includes more no disease than having disease.

Principal Component Analysis

Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set. Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and make analyzing data much easier and faster for machine learning algorithms without extraneous variables to process. Before proceeding with the PCA, it is necessary to evaluate whether there is correlation between the numerical variables. To accomplish this, we'll need a subset of the dataset that contains all continuous values.

```
liver_sub <- liver[ -c(2,11) ]

library(psych)

## Warning: package 'psych' was built under R version 4.0.5

pairs.panels(liver_sub, main= "Original Space-Bivariate Scatter Plots",
             ellipses = FALSE, gap = 0)
```



This is a preliminary analysis of the data in the original space, which aim is to understand if it would be useful to run a Principal Component Analysis and a Cluster one on this dataset.

In fact, in the upper triangle of the matrix there are the coefficients of correlation between variables, which are used to understand if PCA is useful or not, while in the lower triangle there are the scatterplots of data and on the main diagonal there is the non-parametric density of the data, both used to understand if Component analysis could be useful or not. Specifically, if we look at the plot, we can see that there is a high correlation between some variables, for example between `Age` and `Total_Bilirubin`, which is `0.87` have a strongest positive correlation, `Alamine_Aminotransferase` and `Aspartate_Aminotransferase`, which is `0.79` having a strongest positive correlation, `Total_Protiens` and `Albumin` having a strongest positive correlation of `0.78` and also `Albumin` and `Albumin_and_Globulin_Ratio` having a correlation of `0.69`. This means that a PCA in this dataset could really be useful, in fact we could create a linear combination between the variables and express them through a single variable. As regard to the diagonal panels, they are used to understand if the single variable is useful for clustering: if the density line is bi-model, the relative variable could be useful for clustering, otherwise not. In this case, each variable separately considered is not useful to see clusters, but, if we look at the pairwise of variables, it is useful. In fact, from the scatterplots of the data in the lower triangle we can see that there are clusters, because the data are grouped along the diagonal instead of remaining scattered in space.

Prepare the Data

In order to evaluate the difference between the variables, the mean and the variance are computed for each variable.

```
apply(liver_sub, 2, mean)
```

```
##                         Age            Total_Bilirubin
##                  44.7823834                  3.3153713
##           Direct_Bilirubin       Alkaline_Phosphotase
##                  1.4941278                291.3661485
##   Alamine_Aminotransferase Aspartate_Aminotransferase
##                  81.1260794                110.4145078
##             Total_Protiens                    Albumin
##                  6.4816926                  3.1385147
## Albumin_and_Globulin_Ratio
##                  0.9470639
```

```
apply(liver_sub, 2, var)
```

```
##                         Age            Total_Bilirubin
##               2.631463e+02               3.878445e+01
##           Direct_Bilirubin       Alkaline_Phosphotase
##               7.932664e+00               5.932238e+04
##   Alamine_Aminotransferase Aspartate_Aminotransferase
##               3.355595e+04               8.401304e+04
##             Total_Protiens                    Albumin
##               1.176446e+00               6.311265e-01
## Albumin_and_Globulin_Ratio
##               1.021391e-01
```

There is a great difference in the variables. It is preferable to normalize a variable in order to have a zero mean and uniform variance when working with homogeneous variables.

```
scaled_liver <- apply(liver_sub, 2, scale)
head(scaled_liver)
```

```
##             Age Total_Bilirubin Direct_Bilirubin Alkaline_Phosphotase
## [1,] 1.24632497     -0.41995671       -0.4949862           -0.4284995
## [2,] 1.06138848      1.21788279        1.4222880            1.6736358
## [3,] 1.06138848      0.63982179        0.9252169            0.8155376
## [4,] 0.81480651     -0.37178496       -0.3884709           -0.4490282
## [5,] 1.67784343      0.09387529        0.1796103           -0.3956537
## [6,] 0.07506058     -0.24332696       -0.2819557           -0.3422792
##      Alamine_Aminotransferase Aspartate_Aminotransferase Total_Protiens
## [1,]              -0.35552499                -0.31883559      0.2934680
## [2,]              -0.09349172                -0.03593068      0.9388428
## [3,]              -0.11532783                -0.14633260      0.4778608
## [4,]              -0.36644304                -0.31193547      0.2934680
## [5,]              -0.29547570                -0.17738313      0.7544500
## [6,]              -0.33914791                -0.33263583      1.0310392
##          Albumin Albumin_and_Globulin_Ratio
## [1,]  0.20327072                -0.1472624
## [2,]  0.07739506                -0.6479006
## [3,]  0.20327072                -0.1785523
## [4,]  0.32914639                 0.1656364
## [5,] -0.92961029                -1.7117566
## [6,]  1.58790307                 1.1043330
```

Computing PCs

In order to find the Principal Components, the Eigen decomposition is applied to the covariance matrix of the standardized data.

```
liver_cov <- cov(scaled_liver)
liver_eigen<-eigen(liver_cov)
liver_eigen$value
```

```
## [1] 2.75391157 2.02686232 1.36548027 0.95799983 0.84466046 0.66638940
0.20344650
## [8] 0.12581520 0.05543444
```

The eigen vectors of the PCs are displayed as an example:

```
phi <- liver_eigen$vectors[,1:3]
phi <- -phi
row.names(phi) <- c("Age", "Total_Bilirubin", "Direct_Bilirubin",
"Alkaline_Phosphotase", "Alamine_Aminotransferase",
"Aspartate_Aminotransferase", "Total_Protiens", "Albumin",
"Albumin_and_Globulin_Ratio")
colnames(phi) <- c("PC1", "PC2", "PC3")
phi
```

```
##                                     PC1         PC2         PC3
## Age                           0.1401403  0.28574605 -0.01471618
## Total_Bilirubin               0.4134106 -0.25254722  0.45274491
## Direct_Bilirubin              0.4195284 -0.26370362  0.44042614
## Alkaline_Phosphotase          0.2530276 -0.05438827  0.09926171
## Alamine_Aminotransferase      0.2671770 -0.41670205 -0.50458906
## Aspartate_Aminotransferase    0.3005978 -0.39339220 -0.48188882
## Total_Protiens               -0.2775673 -0.41943858  0.29778888
## Albumin                      -0.4400519 -0.43099512  0.11470573
## Albumin_and_Globulin_Ratio   -0.3701262 -0.30329504 -0.04811273
```

By examining the loading we note that first loading vector phi 1 puts most of its weight on Direct_Bilirubin (0.419) and much less weight on Albumin (-0.440). The second loading vector phi 2 puts most of its weight on Age (0.285) and much less weight on Albumin (-0.430). The third loading vector phi 3 puts most of its weight on Total_Bilirubin (0.452) and much less weight on Alamine_Aminotransferase (-0.504).

Principal Component Scores

```
PC1 <- scaled_liver %*% phi[,1]
PC2 <- scaled_liver %*% phi[,2]
PC3 <- scaled_liver %*% phi[,3]
PC <- data.frame(ID = row.names(liver), PC1, PC2, PC3)
head(PC)
```

```
##   ID        PC1        PC2         PC3
## 1  1 -0.6222677  0.7235652 -0.01818250
## 2  2  1.5817709 -0.6479197  1.71242776
## 3  3  0.7769593 -0.2749009  1.06541563
## 4  4 -0.7953810  0.4138056  0.05641998
## 5  5  0.9501497  1.2261700  0.49260715
## 6  6 -1.8792566 -1.0037263  0.49803940
```

```
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

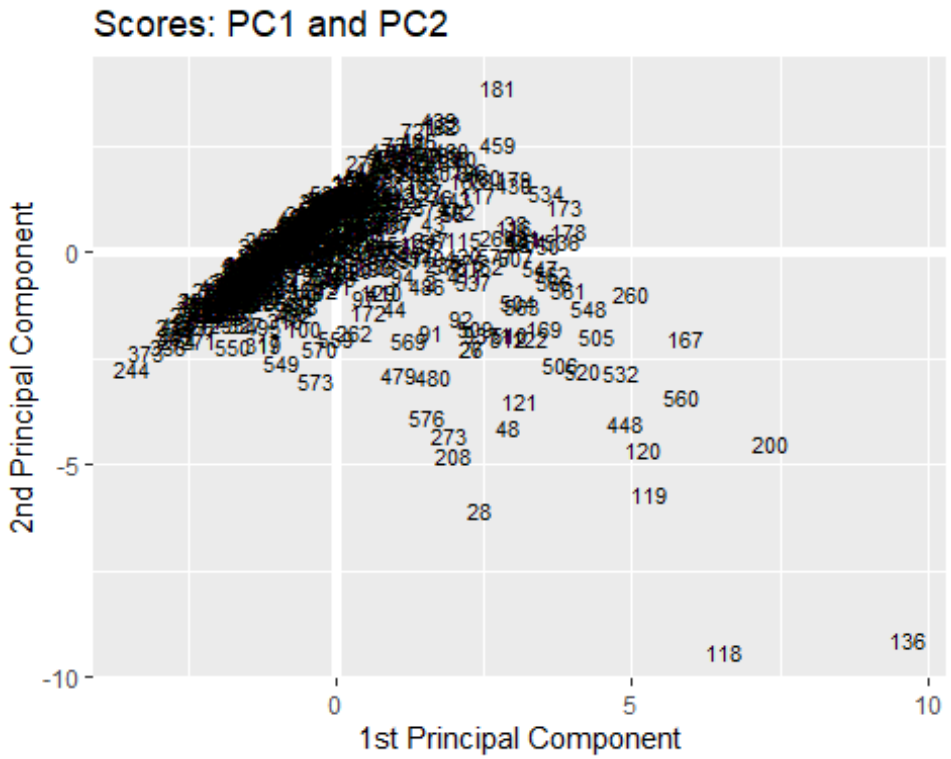
```
## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

```
library(modelr)
```

```
## Warning: package 'modelr' was built under R version 4.0.5
```

```
ggplot(PC, aes(PC1, PC2)) +
modelr::geom_ref_line(h = 0) +
modelr::geom_ref_line(v = 0) +
geom_text(aes(label = ID), size = 3) +
xlab("1st Principal Component") +
```

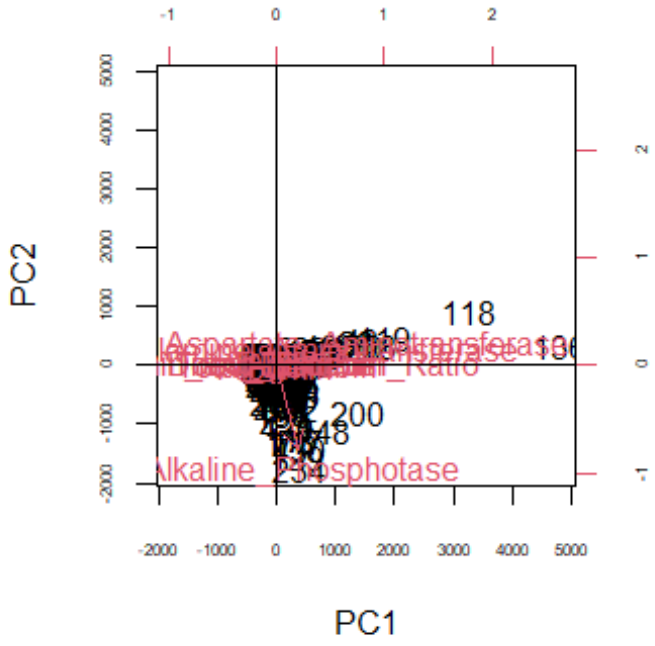
```
ylab("2nd Principal Component") +
ggtitle("Scores: PC1 and PC2")
```



Biplot

It is possible to visualize the scores and the original variable (represented by arrows) in the space spanned by the first two principal components. we set center= True to shift the variable into zero center as follows:

```
set.seed(123)
liver_pc <- prcomp(liver_sub, center = TRUE, scale. = FALSE)
biplot(liver_pc, cex.axis = 0.5, scale=0)

## Warning in arrows(0, 0, y[, 1L] * 0.8, y[, 2L] * 0.8, col = col[2L],
length =
## arrow.len): zero-length arrow is of indeterminate angle and so skipped

## Warning in arrows(0, 0, y[, 1L] * 0.8, y[, 2L] * 0.8, col = col[2L],
length =
## arrow.len): zero-length arrow is of indeterminate angle and so skipped

## Warning in arrows(0, 0, y[, 1L] * 0.8, y[, 2L] * 0.8, col = col[2L],
length =
## arrow.len): zero-length arrow is of indeterminate angle and so skipped

abline(h=0)
abline(v=0)
```



The angle between the arrows gives information on the correlation between the two variables.

```
cor(liver_sub)
```

```
##                                    Age Total_Bilirubin Direct_Bilirubin
## Age                         1.000000000     0.011000374     6.784303e-03
## Total_Bilirubin             0.011000374     1.000000000     8.744810e-01
## Direct_Bilirubin            0.006784303     0.874480969     1.000000e+00
## Alkaline_Phosphotase        0.078878350     0.205739173     2.340076e-01
## Alamine_Aminotransferase   -0.087799162     0.213375493     2.331801e-01
## Aspartate_Aminotransferase -0.020498946     0.237323055     2.570224e-01
## Total_Protiens             -0.186248122    -0.007905923     3.270877e-05
## Albumin                    -0.264210935    -0.222086570    -2.284092e-01
## Albumin_and_Globulin_Ratio -0.216408346    -0.206267186    -2.001247e-01
##                            Alkaline_Phosphotase Alamine_Aminotransferase
## Age                                  0.07887835              -0.08779916
## Total_Bilirubin                      0.20573917               0.21337549
## Direct_Bilirubin                     0.23400757               0.23318008
## Alkaline_Phosphotase                 1.00000000               0.12477671
## Alamine_Aminotransferase             0.12477671               1.00000000
## Aspartate_Aminotransferase           0.16657999               0.79186215
## Total_Protiens                      -0.02706202              -0.04243210
## Albumin                             -0.16341865              -0.02865750
## Albumin_and_Globulin_Ratio          -0.23416650              -0.00237499
##                            Aspartate_Aminotransferase Total_Protiens
## Age                                        -0.02049895  -1.862481e-01
```

```
## Total_Bilirubin                          0.23732305  -7.905923e-03
## Direct_Bilirubin                         0.25702239   3.270877e-05
## Alkaline_Phosphotase                     0.16657999  -2.706202e-02
## Alamine_Aminotransferase                 0.79186215  -4.243210e-02
## Aspartate_Aminotransferase               1.00000000  -2.575101e-02
## Total_Protiens                          -0.02575101   1.000000e+00
## Albumin                                 -0.08491457   7.831122e-01
## Albumin_and_Globulin_Ratio              -0.07003983   2.348872e-01
##                               Albumin Albumin_and_Globulin_Ratio
## Age                        -0.26421094                -0.21640835
## Total_Bilirubin            -0.22208657                -0.20626719
## Direct_Bilirubin           -0.22840915                -0.20012469
## Alkaline_Phosphotase       -0.16341865                -0.23416650
## Alamine_Aminotransferase   -0.02865750                -0.00237499
## Aspartate_Aminotransferase -0.08491457                -0.07003983
## Total_Protiens              0.78311217                 0.23488718
## Albumin                     1.00000000                 0.68963234
## Albumin_and_Globulin_Ratio  0.68963234                 1.00000000
```

To select the number of principal components, three heuristic methods are proposed as follows:

Cumulative Proportion of Variance Explained (CPVE)

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

```
(PVE <- liver_eigen$values/sum(liver_eigen$values))
```

```
## [1] 0.305990174 0.225206925 0.151720030 0.106444426 0.093851162
0.074043266
## [7] 0.022605167 0.013979467 0.006159383
```

- The First PC explains 30.59% of the variability.
- The Second PC explains 22.52% of the variability.
- The Third PC explains 15.17% of the variability.
- The Fourth PC explains 10.64% of the variability.
- The Fifth PC explains 9.38% of the variability.
- The Sixth PC explains 7.40% of the variability.
- The Seven PC explains 2.26% of the variability.
- The Eight PC explains 1.39% of the variability.
- The Ninth PC explains 0.61% of the variability.

```
cumsum(PVE)
```

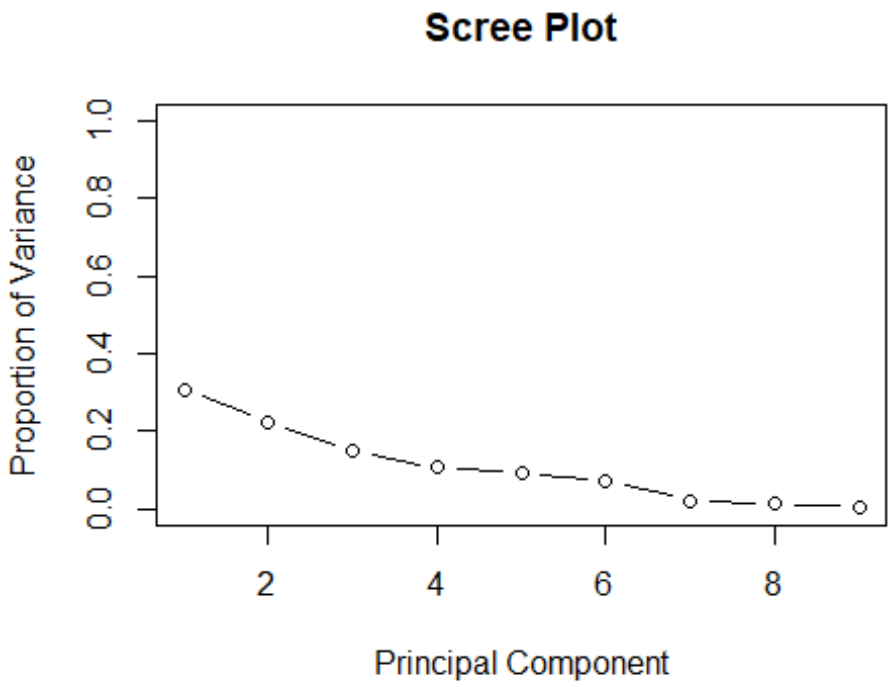
```
## [1] 0.3059902 0.5311971 0.6829171 0.7893616 0.8832127 0.9572560 0.9798612
## [8] 0.9938406 1.0000000
```

According to this method, the first five principal components are retained because together they explain the 88% of the total variance.

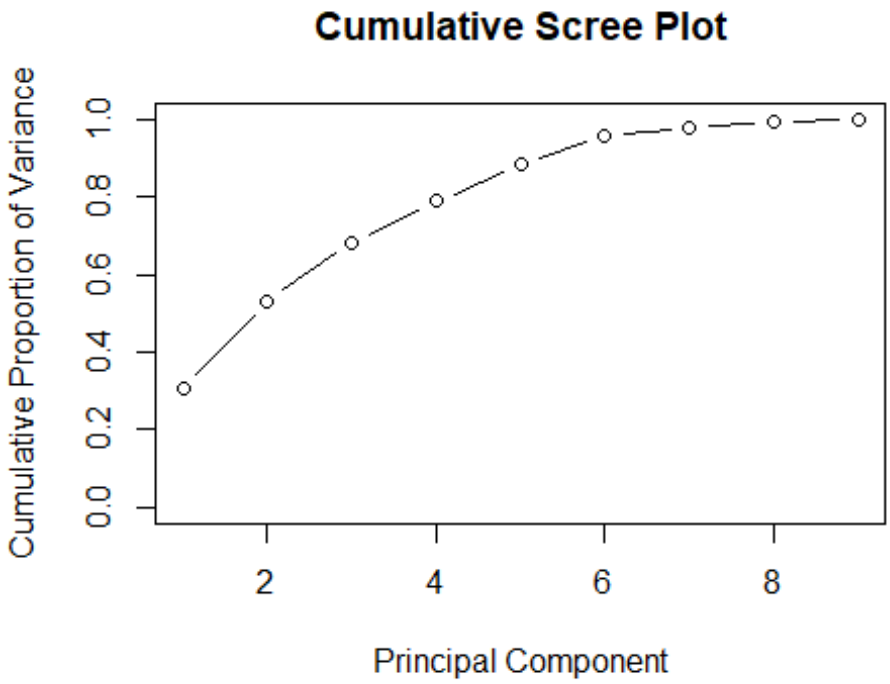
Scree Plot

The scree plot shows the value of q that matches the value of m when the curve falls flat.

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




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



In the "Proportion of variance Explained" plot, the elbow point in not so clear and it may be at q=2 or q=4 or q=6. However, according to this method, it seems reasonable to retain the first six principal components.

Kaiser's Rule

For standardized data, the principal components with a variance greater than one are chosen according to Kaiser's rule.

```
liver_eigen$values

## [1] 2.75391157 2.02686232 1.36548027 0.95799983 0.84466046 0.66638940
0.20344650
## [8] 0.12581520 0.05543444
```

The rule of the Kaiser indicates that first three principal component should be maintained.

PCA Result

I have achieved various results based on various methods. The 'CPVE' rule recommends that the first five components are retained, while the Scree plot provides result to retain the first six components but the kaiser's rule implies that the first three components are maintained. I chose the results of Scree plot since more PCs will be obtained.

Cluster Analysis

The purpose of the clustering is to locate homogeneous subgroups in the liver dataset and to accomplish this analysis, various methods can be useful. The analysis composed of many different steps. In the first step, a certain sort of distance is computed among pairs and the distance matrix is established. This is because in this kind of analysis, the concept of dissimilarity is important, since the unit most "similar" will be put into the same cluster, while there must be a large dissimilarity between the other clusters.

Hopkins statistic

```
liver_scale <- scale(liver_sub)

library(clustertend)

## Warning: package 'clustertend' was built under R version 4.0.5

hopkins(liver_scale, n = nrow(liver_scale)-1)

## $H
## [1] 0.09085019
```

The statistical value of Hopkins is nearly 0. The outcome is clustered data, assuming that the uniform distribution is the configuration without cluster.

Euclidean Distance

Let us compute the Euclidean distance as follows:

```
dist.eucl <- dist(liver_scale, method = "euclidean")
eucl <- round(as.matrix(dist.eucl)[1:9, 1:9], 2)
row.names(eucl) <- c("Age", "Total_Bilirubin", "Direct_Bilirubin",
"Alkaline_Phosphotase", "Alamine_Aminotransferase",
"Aspartate_Aminotransferase", "Total_Protiens", "Albumin",
"Albumin_and_Globulin_Ratio")
colnames(eucl) <- c("Age", "Total_Bilirubin", "Direct_Bilirubin",
"Alkaline_Phosphotase", "Alamine_Aminotransferase",
"Aspartate_Aminotransferase", "Total_Protiens", "Albumin",
"Albumin_and_Globulin_Ratio")
eucl

##                             Age Total_Bilirubin Direct_Bilirubin
## Age                        0.00            3.41             2.20
## Total_Bilirubin            3.41            0.00             1.33
## Direct_Bilirubin           2.20            1.33             0.00
## Alkaline_Phosphotase       0.56            3.42             2.16
## Alamine_Aminotransferase   2.21            3.12             2.54
## Aspartate_Aminotransferase 2.34            3.95             2.92
## Total_Protiens             2.45            4.15             3.17
## Albumin                    2.34            4.07             3.03
## Albumin_and_Globulin_Ratio 3.31            4.62             3.76
```

```
##                            Alkaline_Phosphotase Alamine_Aminotransferase
## Age                                        0.56                     2.21
## Total_Bilirubin                            3.42                     3.12
## Direct_Bilirubin                           2.16                     2.54
## Alkaline_Phosphotase                       0.00                     2.58
## Alamine_Aminotransferase                   2.58                     0.00
## Aspartate_Aminotransferase                 1.90                     4.16
## Total_Protiens                             1.99                     3.78
## Albumin                                    1.84                     3.88
## Albumin_and_Globulin_Ratio                 2.81                     4.79
##                            Aspartate_Aminotransferase Total_Protiens
Albumin
## Age                                              2.34           2.45
2.34
## Total_Bilirubin                                  3.95           4.15
4.07
## Direct_Bilirubin                                 2.92           3.17
3.03
## Alkaline_Phosphotase                             1.90           1.99
1.84
## Alamine_Aminotransferase                         4.16           3.78
3.88
## Aspartate_Aminotransferase                       0.00           2.02
1.80
## Total_Protiens                                   2.02           0.00
0.51
## Albumin                                          1.80           0.51
0.00
## Albumin_and_Globulin_Ratio                       1.87           1.20
1.21
##                            Albumin_and_Globulin_Ratio
## Age                                              3.31
## Total_Bilirubin                                  4.62
## Direct_Bilirubin                                 3.76
## Alkaline_Phosphotase                             2.81
## Alamine_Aminotransferase                         4.79
## Aspartate_Aminotransferase                       1.87
## Total_Protiens                                   1.20
## Albumin                                          1.21
## Albumin_and_Globulin_Ratio                       0.00
```

In this symmetric matrix, each value represents the distance between units. The values on the diagonal represent the distance between units and themselves (which is zero).

Manhattan Distance

Let us compute the Manhattan distance as follows:

```
dist.man <- dist(liver_scale, method = "manhattan")
man <- round(as.matrix(dist.man)[1:9,1:9],2)
```

```r
row.names(man) <- c("Age", "Total_Bilirubin", "Direct_Bilirubin",
"Alkaline_Phosphotase", "Alamine_Aminotransferase",
"Aspartate_Aminotransferase", "Total_Protiens", "Albumin",
"Albumin_and_Globulin_Ratio")
colnames(man) <- c("Age", "Total_Bilirubin", "Direct_Bilirubin",
"Alkaline_Phosphotase", "Alamine_Aminotransferase",
"Aspartate_Aminotransferase", "Total_Protiens", "Albumin",
"Albumin_and_Globulin_Ratio")
man
```

```
##                             Age Total_Bilirubin Direct_Bilirubin
## Age                        0.00            7.66             4.54
## Total_Bilirubin            7.66            0.00             3.12
## Direct_Bilirubin           4.54            3.12             0.00
## Alkaline_Phosphotase       1.06            8.03             4.91
## Alamine_Aminotransferase   5.01            7.65             6.27
## Aspartate_Aminotransferase 5.05           10.06             7.87
## Total_Protiens             3.38           10.16             7.04
## Albumin                    3.51           10.47             7.35
## Albumin_and_Globulin_Ratio 5.66           11.44             9.06
##                            Alkaline_Phosphotase Alamine_Aminotransferase
## Age                                        1.06                     5.01
## Total_Bilirubin                            8.03                     7.65
## Direct_Bilirubin                           4.91                     6.27
## Alkaline_Phosphotase                       0.00                     5.75
## Alamine_Aminotransferase                   5.75                     0.00
## Aspartate_Aminotransferase                 4.06                     8.26
## Total_Protiens                             2.52                     7.89
## Albumin                                    2.61                     8.26
## Albumin_and_Globulin_Ratio                 4.77                     9.41
##                            Aspartate_Aminotransferase Total_Protiens
Albumin
## Age                                              5.05           3.38
3.51
## Total_Bilirubin                                 10.06          10.16
10.47
## Direct_Bilirubin                                 7.87           7.04
7.35
## Alkaline_Phosphotase                             4.06           2.52
2.61
## Alamine_Aminotransferase                         8.26           7.89
8.26
## Aspartate_Aminotransferase                       0.00           4.42
3.86
## Total_Protiens                                   4.42           0.00
1.15
## Albumin                                          3.86           1.15
0.00
## Albumin_and_Globulin_Ratio                       3.01           2.59
2.40
```

```
##                             Albumin_and_Globulin_Ratio
## Age                                               5.66
## Total_Bilirubin                                  11.44
## Direct_Bilirubin                                  9.06
## Alkaline_Phosphotase                              4.77
## Alamine_Aminotransferase                          9.41
## Aspartate_Aminotransferase                        3.01
## Total_Protiens                                    2.59
## Albumin                                           2.40
## Albumin_and_Globulin_Ratio                        0.00
```

In this symmetric matrix, each value represents the distance between units. The values on the diagonal represent the distance between units and themselves (which is zero).

Visualization

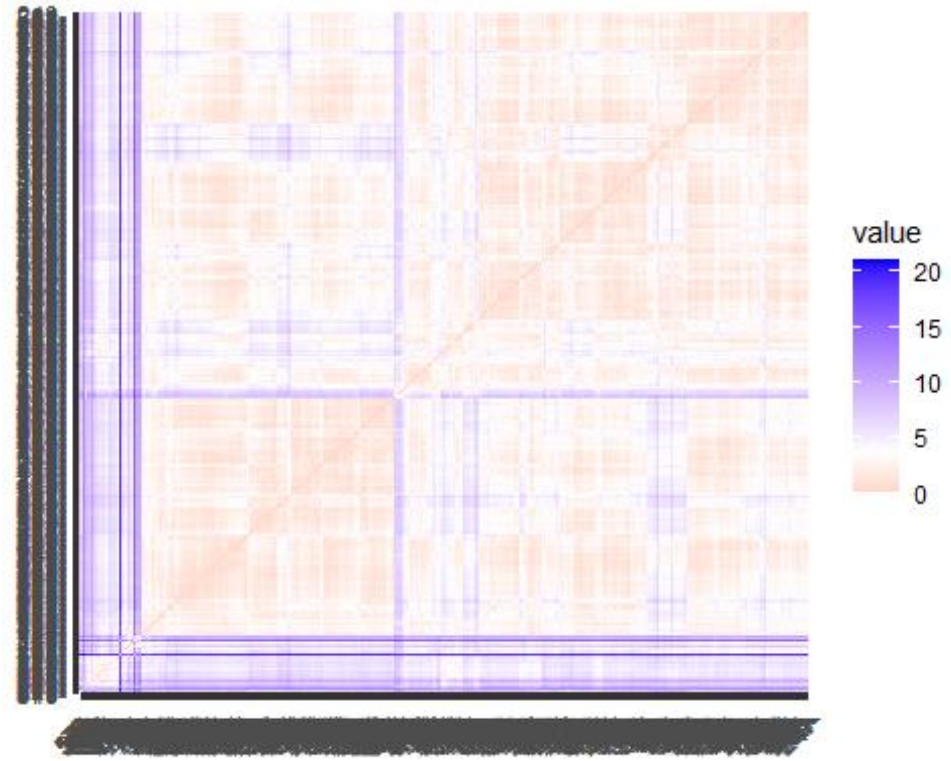
Euclidean

```
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.0.5
```

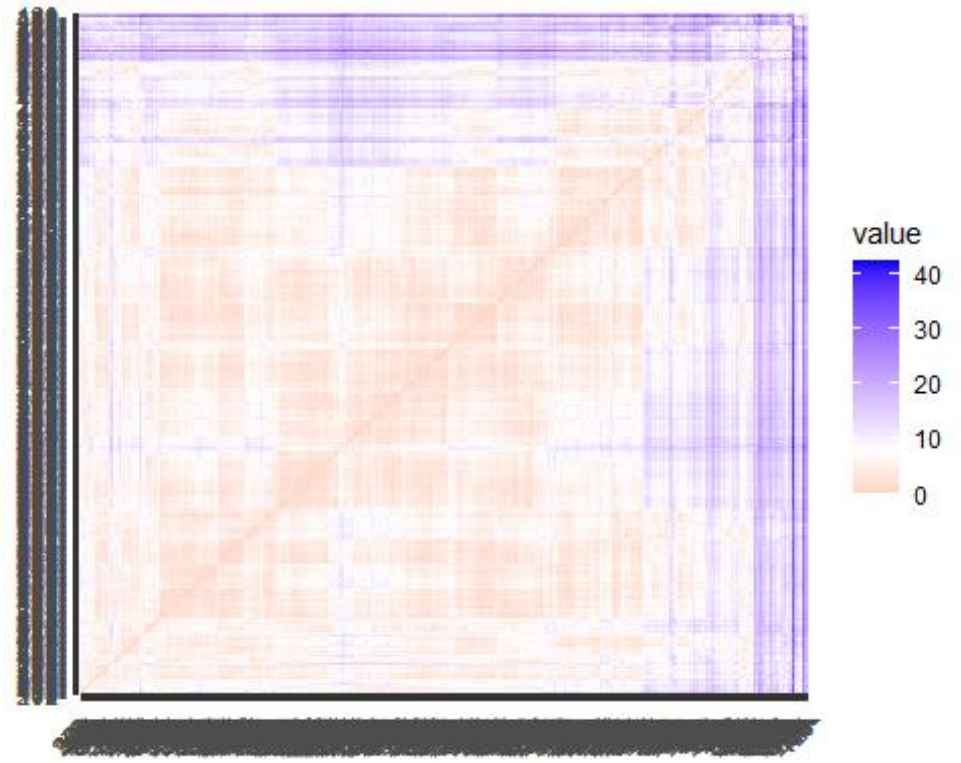
```
## Welcome! Want to learn more? See two factoextra-related books at
https://goo.gl/ve3WBa
```

```
fviz_dist(dist.eucl)
```



Manhattan

```
fviz_dist(dist.man)
```



These are the distance matrices, based on the Euclidean distance. The level of its colors is proportional to the value of the dissimilarity between observations: red stands for high similarity, blue indicates low similarity. Hence, in the diagonal the maximum of similarity is reached- namely the minimum of dissimilarity, in fact there are the pairs made up of each unit with itself, pairs with dissimilarity equal to zero.

Maximum Number of Clusters

Using both Euclidean and Manhattan distance, according to different clustering methods, the maximum number of clusters will be computed as follows:

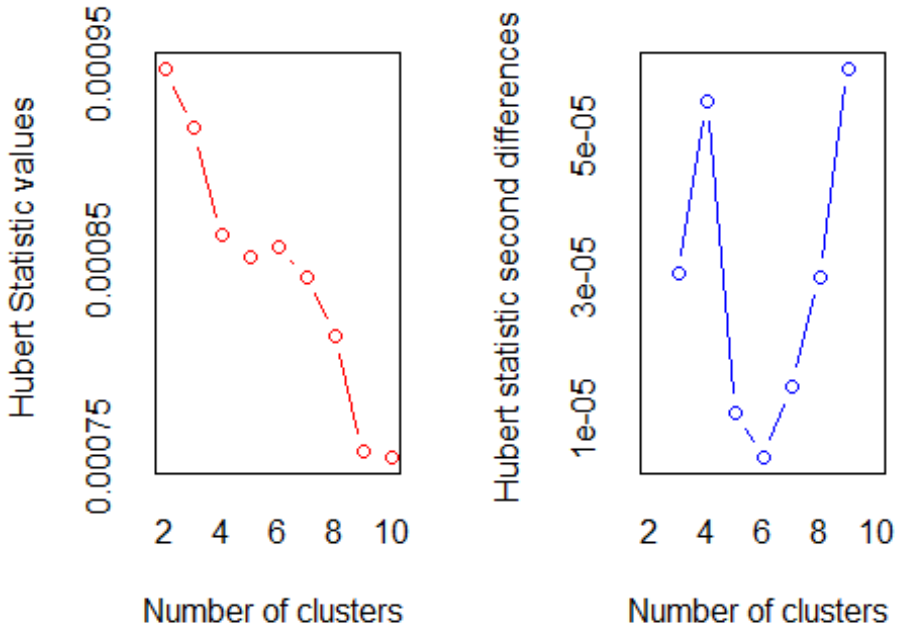
Hierarchical Method

Average Linkage Method & Euclidean Distance

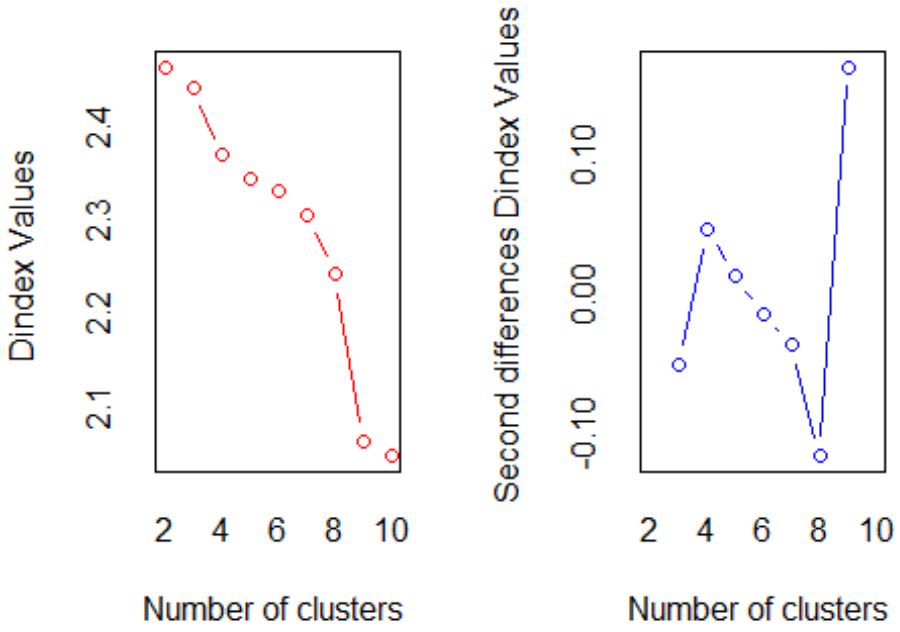
Average linkage method and Euclidean distance means the linkage methods work by calculating the distances or similarities between all objects. Then the closest pair of clusters are combined into a single cluster, reducing the number of clusters remaining. The process is then repeated until there is only a single cluster left.

```
library(NbClust)
nb <- NbClust(liver_scale, distance = "euclidean", min.nc = 2, max.nc = 10,
method = "average")
```

```
## [1] "Frey index : No clustering structure in this data set"
```



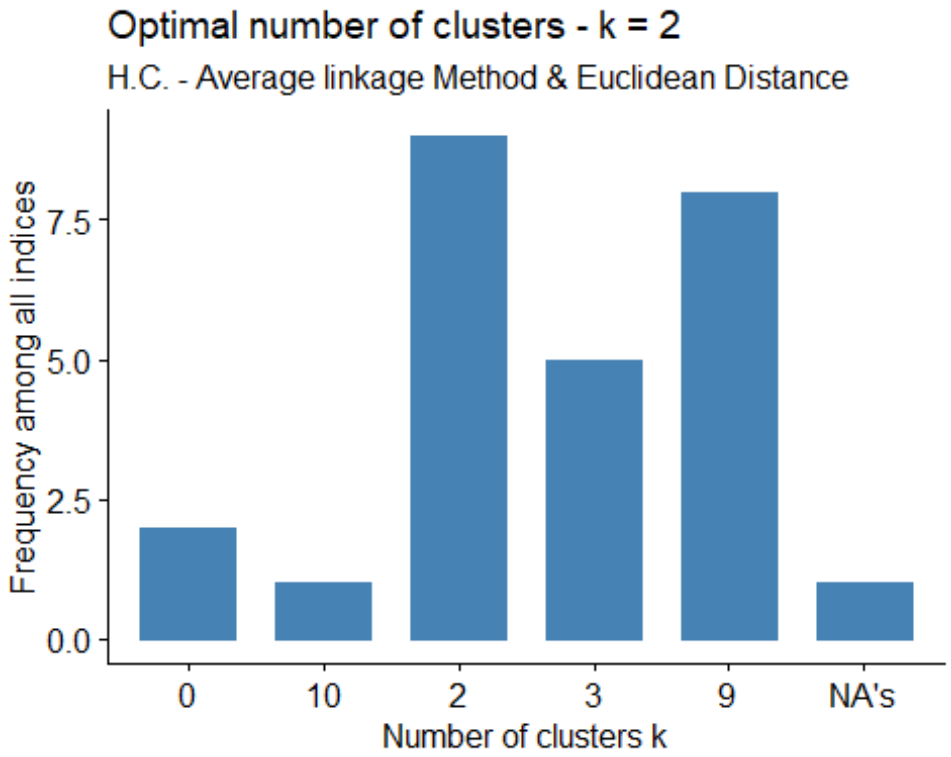

```
## *** : The Hubert index is a graphical method of determining the number of
clusters.
##                 In the plot of Hubert index, we seek a significant knee
that corresponds to a
##                 significant increase of the value of the measure i.e the
significant peak in Hubert
##       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



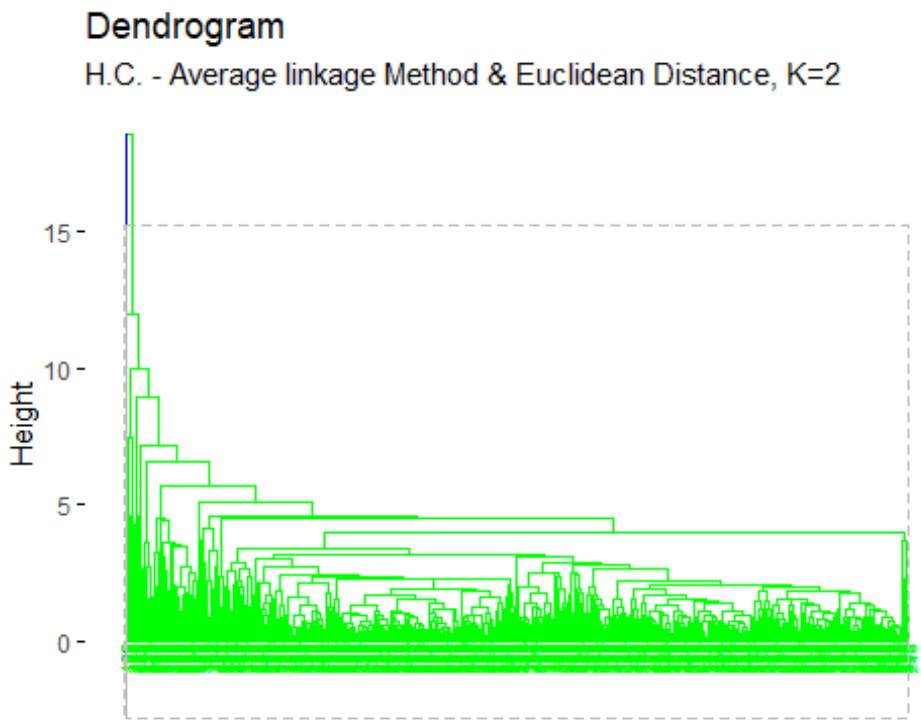

```
## *** : The D index is a graphical method of determining the number of
clusters.
##                 In the plot of D index, we seek a significant knee (the
significant peak in Dindex
##                 second differences plot) that corresponds to a significant
increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 9 proposed 2 as the best number of clusters
## * 5 proposed 3 as the best number of clusters
## * 8 proposed 9 as the best number of clusters
## * 1 proposed 10 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************
```

```
library(factoextra)
fviz_nbclust(nb) +
labs(subtitle = "H.C. - Average linkage Method & Euclidean Distance",
     cex.sub= 0.5)
```

```
## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : the condition has length > 1 and only the first
element
## will be used

## Warning in if (class(best_nc) == "matrix") .viz_NbClust(x, print.summary,
: the
## condition has length > 1 and only the first element will be used

## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : the condition has length > 1 and only the first
element
## will be used

## Warning in if (class(best_nc) == "matrix") {: the condition has length > 1
and
## only the first element will be used

## Among all indices:
## ===================
## * 2 proposed  0 as the best number of clusters
## * 9 proposed  2 as the best number of clusters
## * 5 proposed  3 as the best number of clusters
## * 8 proposed  9 as the best number of clusters
## * 1 proposed  10 as the best number of clusters
## * 1 proposed  NA's as the best number of clusters
##
## Conclusion
## =========================
## * According to the majority rule, the best number of clusters is  2 .
```




```
hc <- hclust(dist.eucl, method = "average")
grp <- cutree(hc, k=2)
table(grp)

## grp
##   1   2
## 578   1

head(grp)

## 1 2 3 4 5 6
## 1 1 1 1 1 1

fviz_dend(hc, k = 2, cex = 0.5, k_colors = c("blue", "green"),
          color_labels_by_k = TRUE, rect = TRUE) +
labs(title = "Dendrogram", subtitle = "H.C. - Average linkage Method &
Euclidean Distance, K=2", cex.subtitle= 0.5)
```

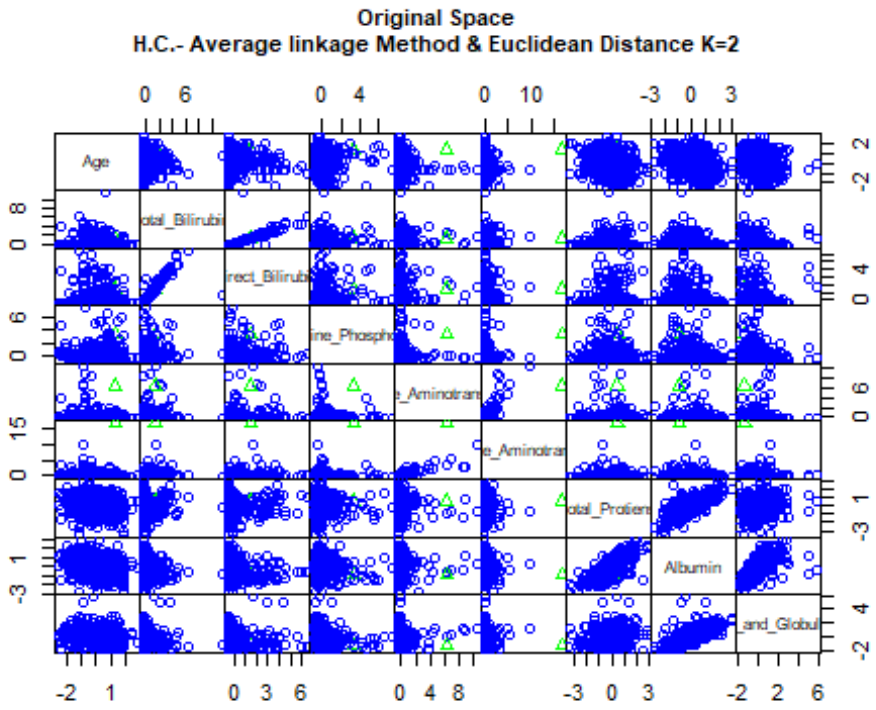



```
cor(dist.eucl, cophenetic(hc))
```

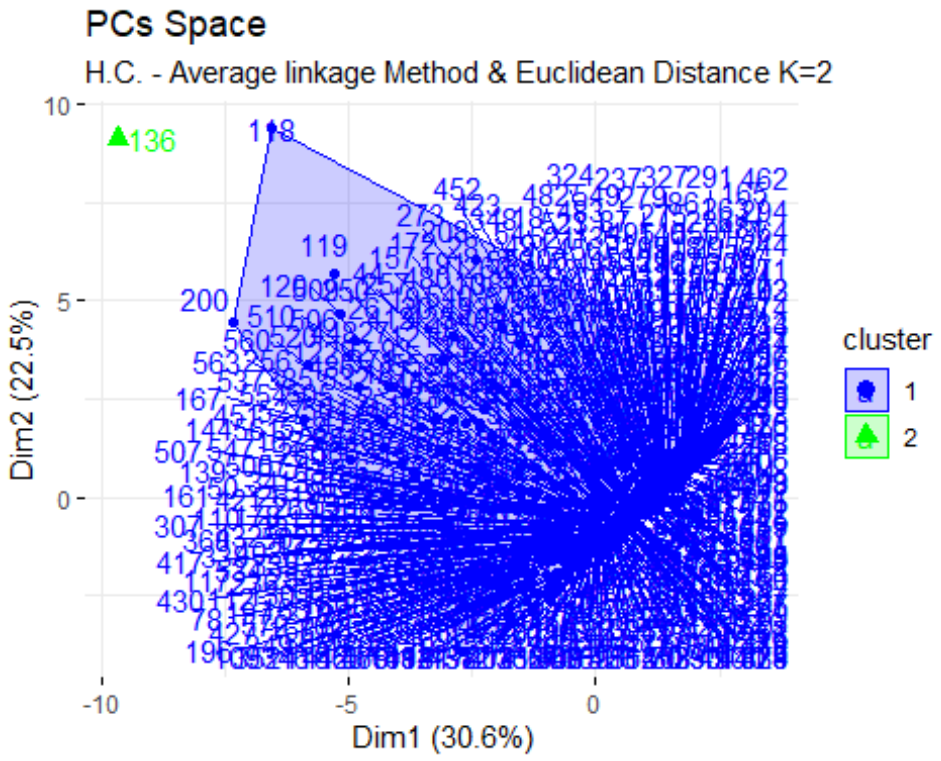
```
## [1] 0.8818356
```

According to the result of "NbClust", the best number of clusters, applying hierarchical clustering using average linkage method and euclidean distance is 2.

```
pairs(liver_scale, gap=0, pch=grp, cex.main= 0.7,
      main="Original Space\nH.C.- Average linkage Method & Euclidean Distance
K=2",
      col=c("blue", "green")[grp])
```




```
options(ggrepel.max.overlaps = Inf)

fviz_cluster(list(data = liver_scale, cluster = grp), palette = c("blue",
"green"),
             ellipse.type = "convex", main="PCs Space", repel = TRUE,
             show.clust.aver = FALSE, ggtheme = theme_minimal()) +
  labs(subtitle = "H.C. - Average linkage Method & Euclidean Distance K=2",
             cex.sub= 0.5)
```

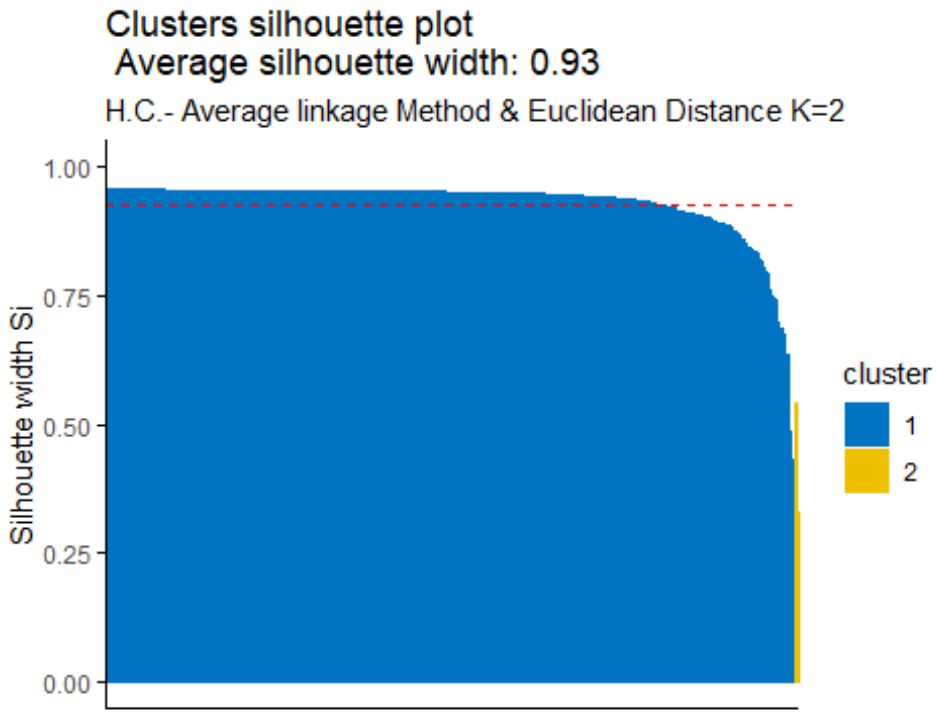


To evaluate the goodness of clustering algorithm results, internal and external validation measures will be analyzed as follows:

Internal Validation Measures

Silhouette Width

```
hclust<- eclust(liver_sub, k=2, "hclust", hc_method  = "average", nboot = 50,
                hc_metric = "euclidean")
silinfo <- hclust$silinfo
silinfo$avg.width

## [1] 0.9252641

fviz_silhouette(hclust, palette = "jco", ggtheme = theme_classic()) +
  labs(subtitle = "H.C.- Average linkage Method & Euclidean Distance K=2",
  cex.sub= 0.5)

##   cluster size ave.sil.width
## 1       1  577          0.93
## 2       2    2          0.44
```




```
silinfo$clus.avg.widths
```

```
## [1] 0.9269589 0.4363246
```

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

```
## [1] cluster   neighbor  sil_width
## <0 rows> (or 0-length row.names)
```

The value of average silhouette width indicates that in average the units are well enough clustered. As in particular, in cluster one the units are on average the same silhouette value with respect to the silhouette width but second cluster which has the lower silhouette value with respect to the silhouette width.

Dunn Index

```
library(fpc)
```

```
## Warning: package 'fpc' was built under R version 4.0.5
```

```
stats <- cluster.stats(dist(liver_scale), hclust$cluster)
stats$dunn
```

```
## [1] 0.3913763
```

Units are not clustered sufficiently according to the Dunn index.

External Validation Measures

Confusion Index

According to the Confusion matrix, the number of clusters is more than nominal values. The clusters found are 2 while the nominal variable can take 2 possible values.

```
table(liver$Liver_Disease, hclust$cluster)

##
##       1   2
##   0 412   2
##   1 165   0
```

A large number of patients who doesn't have the liver disease (n = 412) has been classified in cluster 1 while cluster 2 have only 2 values. The same happened for whom have the liver disease (n = 165) classified in cluster 1 while cluster 2 has 0 values.

Correct Rand Index

```
liver.disease <- as.numeric(liver$Liver_Disease)
stats<- cluster.stats(d = dist(liver_scale), liver.disease, hclust$cluster)
stats$corrected.rand

## [1] -0.004118416
```

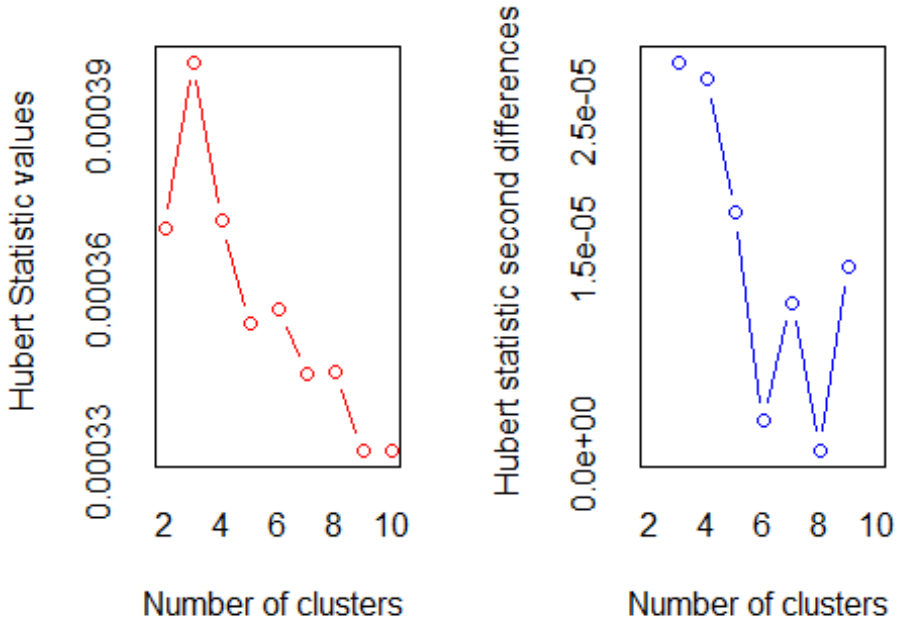
According to the Correct Rand Index, there is no agreement between the numerical value and the cluster solution. From -1 to +1, the agreement is very close to 0.

Meila's VI Index

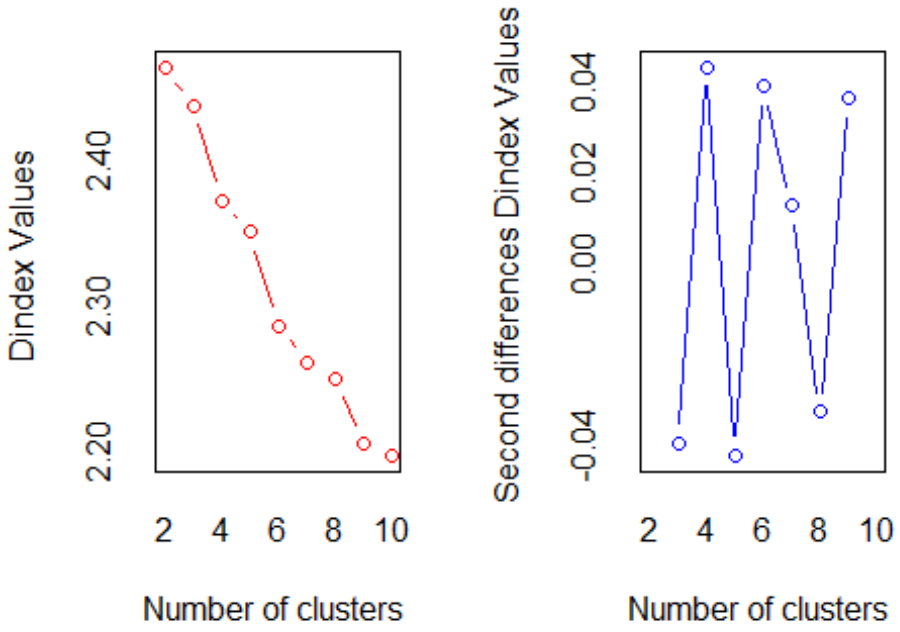
```
stats$vi

## [1] 0.6182953
```

Average Linkage Method & Manhattan Distance

```
nb <- NbClust(liver_scale, distance = "manhattan", min.nc = 2, max.nc = 10,
method = "average")

## Warning in pf(beale, pp, df2): NaNs produced

## [1] "Frey index : No clustering structure in this data set"
```



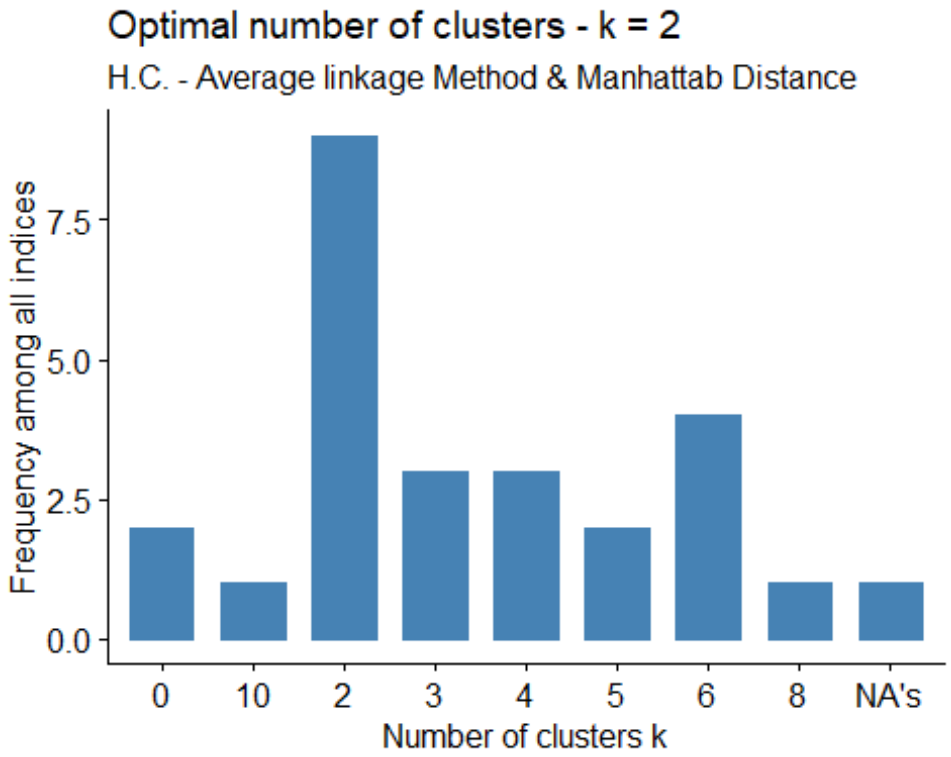

```
## *** : The Hubert index is a graphical method of determining the number of
clusters.
##                 In the plot of Hubert index, we seek a significant knee
that corresponds to a
##                 significant increase of the value of the measure i.e the
significant peak in Hubert
##       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



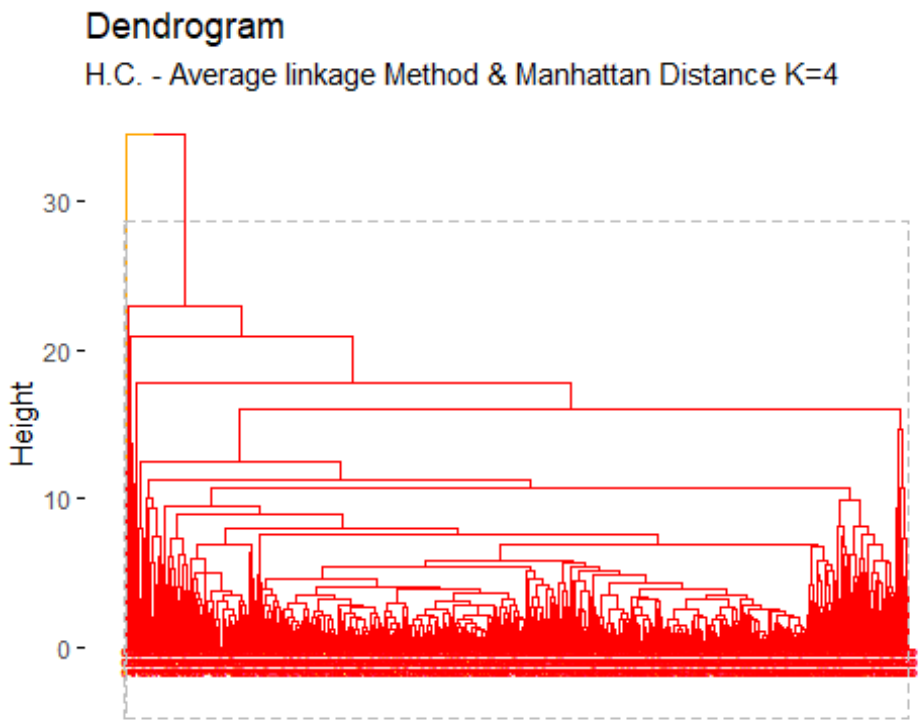

```
## *** : The D index is a graphical method of determining the number of
clusters.
##                 In the plot of D index, we seek a significant knee (the
significant peak in Dindex
##                 second differences plot) that corresponds to a significant
increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 9 proposed 2 as the best number of clusters
## * 3 proposed 3 as the best number of clusters
## * 3 proposed 4 as the best number of clusters
## * 2 proposed 5 as the best number of clusters
## * 4 proposed 6 as the best number of clusters
## * 1 proposed 8 as the best number of clusters
## * 1 proposed 10 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************
```

```r
fviz_nbclust(nb) +
  labs(subtitle = "H.C. - Average linkage Method & Manhattab Distance",
       cex.sub= 0.5)
```

```
## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : the condition has length > 1 and only the first
element
## will be used

## Warning in if (class(best_nc) == "matrix") .viz_NbClust(x, print.summary,
: the
## condition has length > 1 and only the first element will be used

## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : the condition has length > 1 and only the first
element
## will be used

## Warning in if (class(best_nc) == "matrix") {: the condition has length > 1
and
## only the first element will be used

## Among all indices:
## ===================
## * 2 proposed  0 as the best number of clusters
## * 9 proposed  2 as the best number of clusters
## * 3 proposed  3 as the best number of clusters
## * 3 proposed  4 as the best number of clusters
## * 2 proposed  5 as the best number of clusters
## * 4 proposed  6 as the best number of clusters
## * 1 proposed  8 as the best number of clusters
## * 1 proposed  10 as the best number of clusters
## * 1 proposed  NA's as the best number of clusters
##
## Conclusion
## =========================
## * According to the majority rule, the best number of clusters is  2 .
```




```
dist.man <- dist(liver_scale, method = "manhattan")
hc <- hclust(dist.man, method = "average")
grp <- cutree(hc, k=2)
table(grp)

## grp
##   1   2
## 578   1

fviz_dend(hc, k = 2, cex = 0.5, k_colors = c("orange", "red"),
          color_labels_by_k = TRUE, rect = TRUE) +
labs(title = "Dendrogram",
     subtitle = "H.C. - Average linkage Method & Manhattan Distance K=4",
     cex.subtitle= 0.5)
```

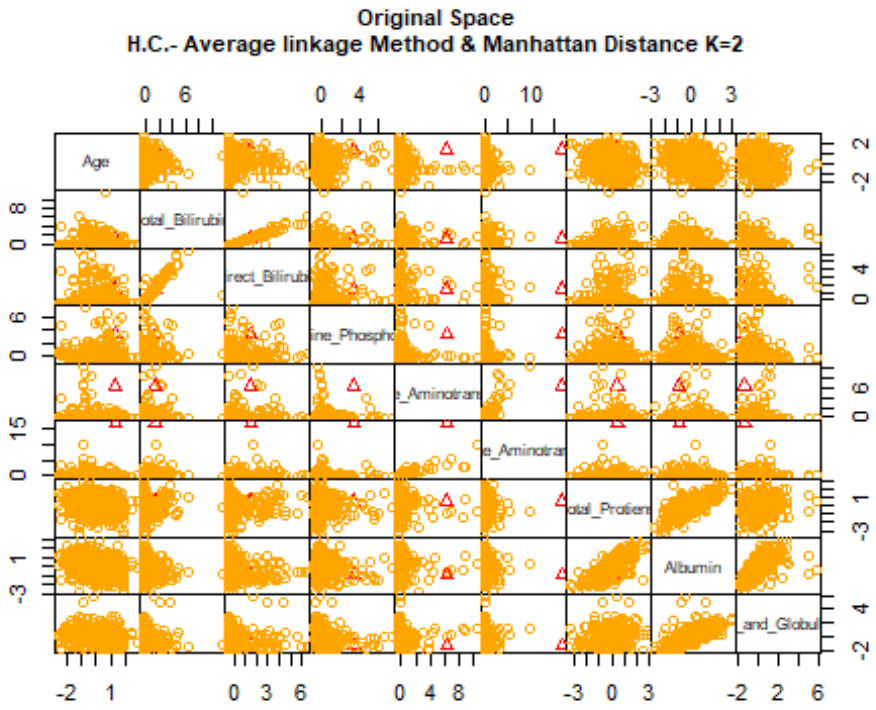



```
cor(dist.man, cophenetic(hc))
```

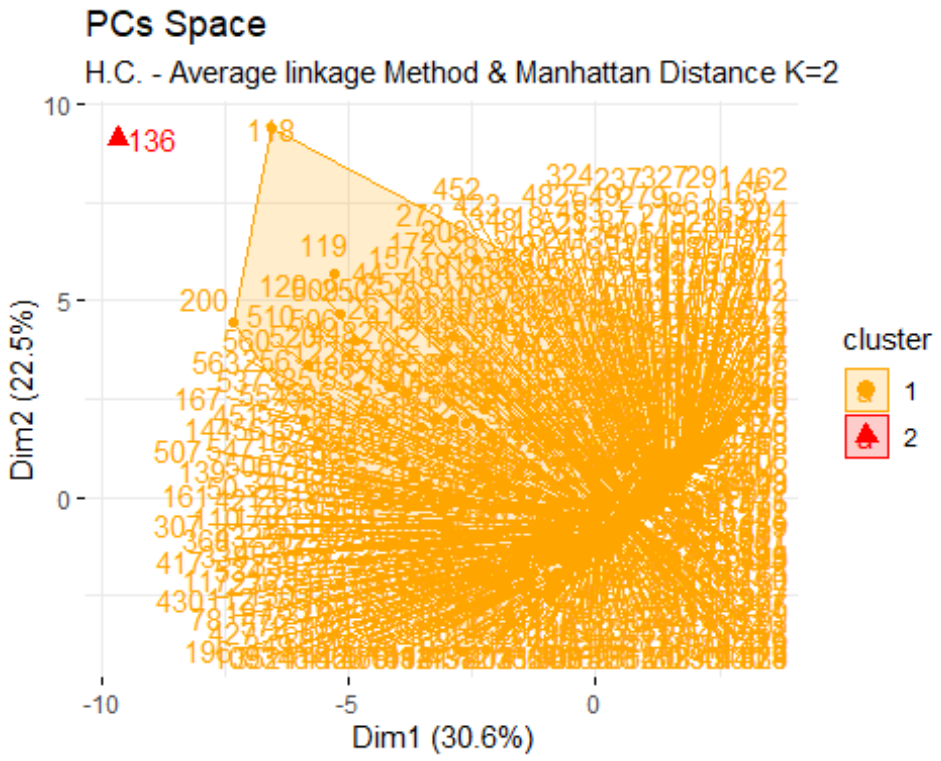
```
## [1] 0.8639619
```

According to the result of "NbClust", the best number of clusters, applying hierarchical clustering using average linkage method and manhattan distance is 2.

```
pairs(liver_scale, gap=0, pch=grp, cex.main= 0.7,
      main="Original Space\nH.C.- Average linkage Method & Manhattan Distance
K=2",
      col=c("orange", "red")[grp])
```




```
options(ggrepel.max.overlaps = Inf)

fviz_cluster(list(data = liver_scale, cluster = grp),
palette = c("orange", "red"), ellipse.type = "convex",
main="PCs Space", repel = TRUE, show.clust.aver = FALSE,
ggtheme = theme_minimal()) +
labs(subtitle = "H.C. - Average linkage Method & Manhattan Distance K=2",
cex.sub= 0.5)
```

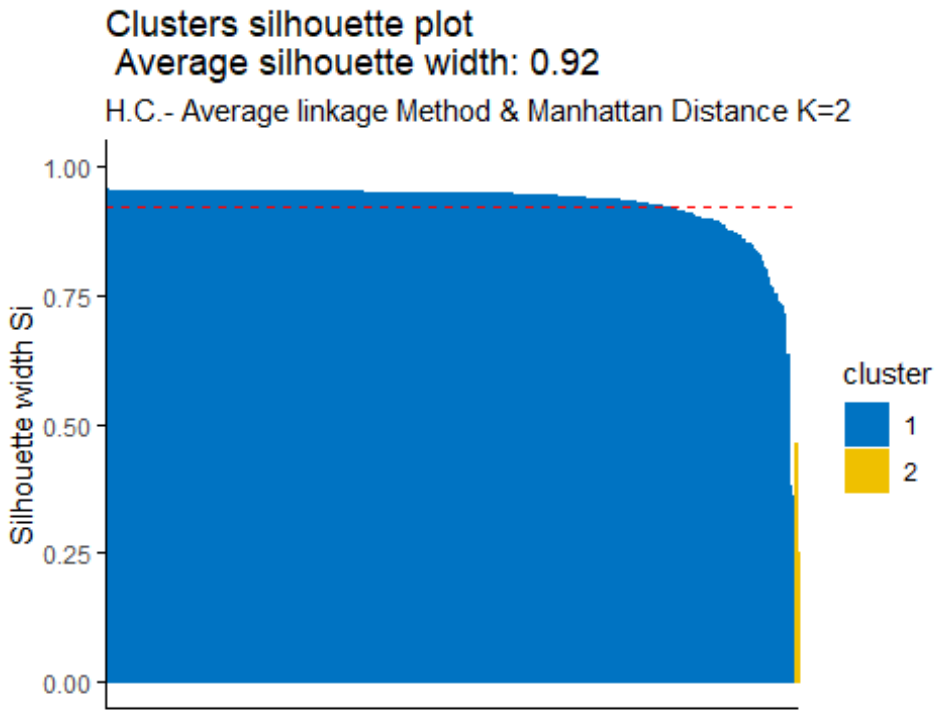


To evaluate the goodness of clustering algorithm results, internal and external validation measures will be analyzed as follows:

Internal Validation Measures

Silhouette width

```
hclust<- eclust(liver_sub, k=2, "hclust", hc_method = "average", nboot = 50,
                hc_metric = "manhattan")
silinfo <- hclust$silinfo
silinfo$avg.width

## [1] 0.9228996

fviz_silhouette(hclust, palette = "jco", ggtheme = theme_classic()) +
labs(subtitle = "H.C.- Average linkage Method & Manhattan Distance K=2",
cex.sub= 0.5)

##   cluster size ave.sil.width
## 1       1  577          0.92
## 2       2    2          0.36
```




```
silinfo$clus.avg.widths
```

```
## [1] 0.9248572 0.3581227
```

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

```
## [1] cluster   neighbor  sil_width
## <0 rows> (or 0-length row.names)
```

The value of average silhouette width indicates that in average the units are well enough clustered. In particular, in cluster 1 (blue cluster) the units having the same silhouette value with respect to the silhouette width, in cluster 2 (the yellow one) the units are on average having the lower silhouette value with respect to silhouette width.

Dunn Index

```
library(fpc)
stats <- cluster.stats(dist(liver_scale), hclust$cluster)
stats$dunn
```

```
## [1] 0.3913763
```

According to the Dunn index, the units are not clustered well enough.

External Validation Measures

Confusion Matrix

According to the Confusion matrix, the number of clusters is equal to nominal values.

```
table(liver$Liver_Disease, hclust$cluster)
```

```
##
##       1   2
##   0 412   2
##   1 165   0
```

For the liver disease, data has been classified mostly in cluster 1, 577 units in cluster 1, and 2 in cluster 2. For the patients having liver disease classified mostly in cluster 1 while cluster 2 has 0 values. Safe to say data are not well balanced in both cluster.

Correct Rand Index

```
liver.disease <- as.numeric(liver$Liver_Disease)
stats<- cluster.stats(d = dist(liver_scale), liver.disease, hclust$cluster)
stats$corrected.rand
```

```
## [1] -0.004118416
```

According to the Correct Rand Index, there is no agreement between the numerical values and the cluster solution. From -1 to +1, the agreement is very close to 0.

Meila's VI Index

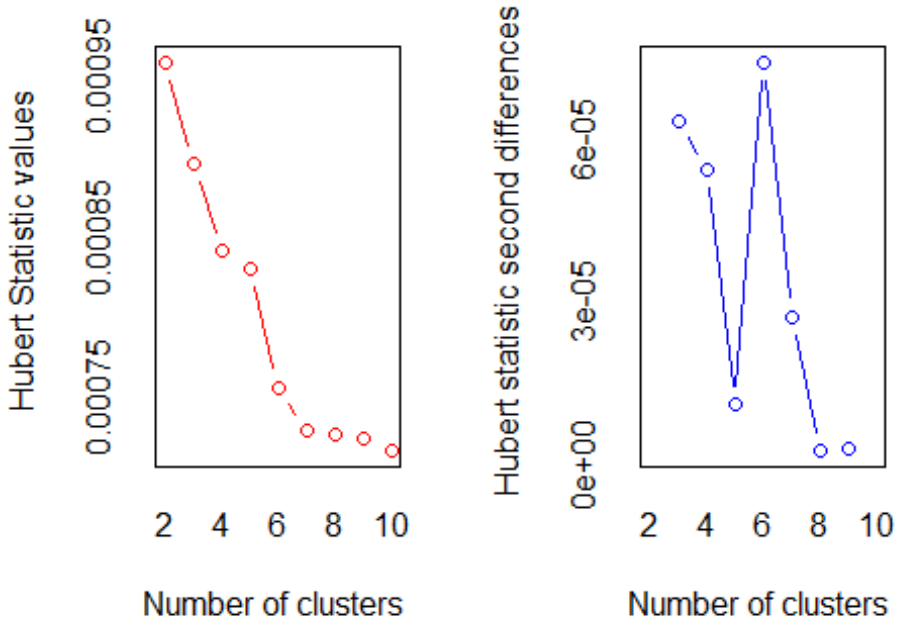
```
stats$vi
```

```
## [1] 0.6182953
```

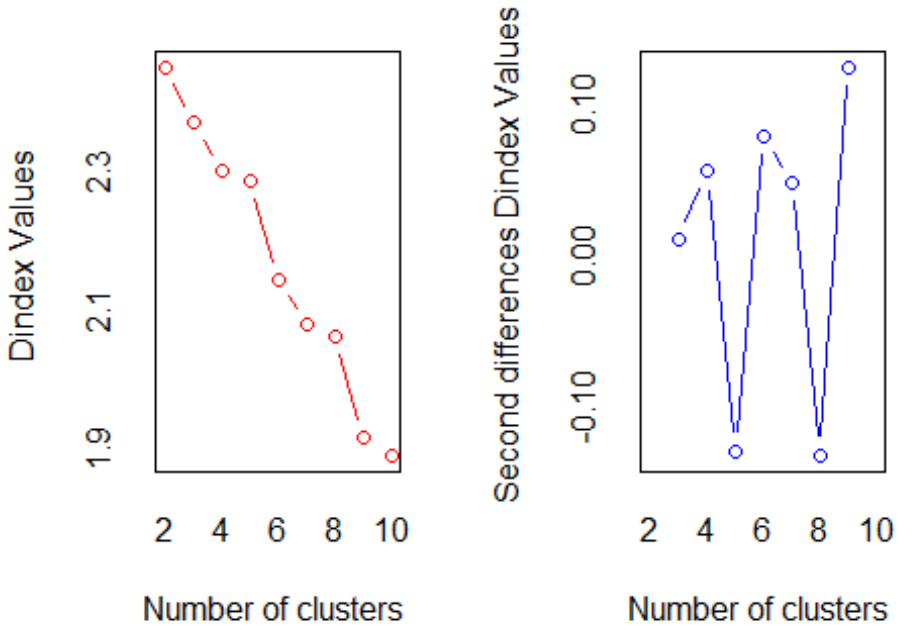
Complete Linkage Method & Euclidean Distance

```
nb <- NbClust(liver_scale, distance = "euclidean", min.nc = 2, max.nc = 10,
              method = "complete")
```

```
## Warning in pf(beale, pp, df2): NaNs produced
```



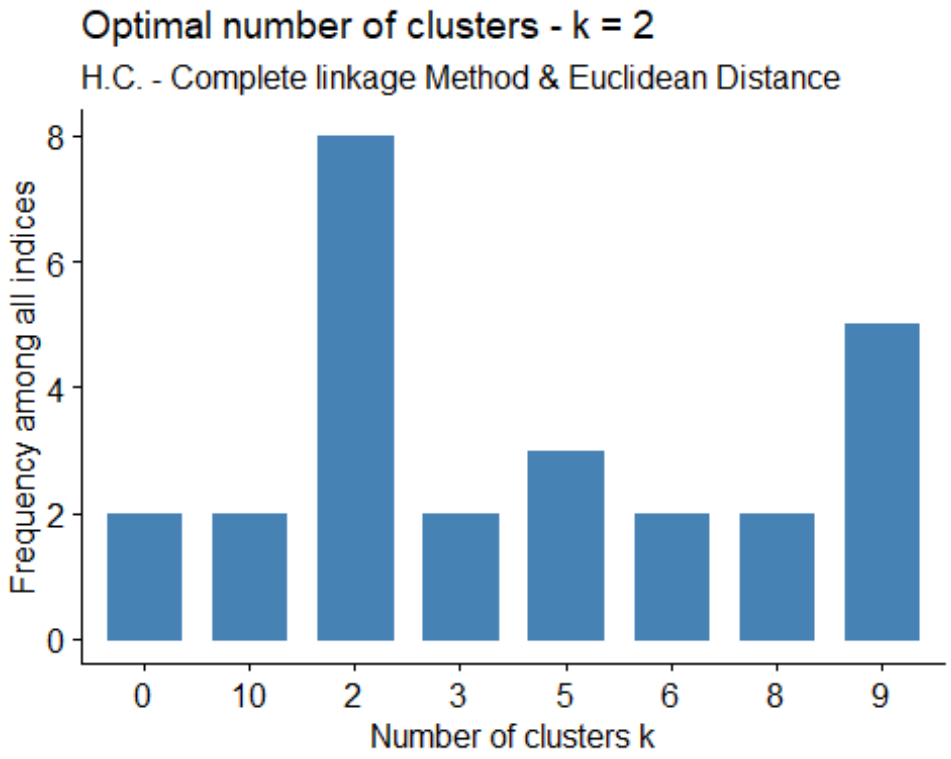

```
## *** : The Hubert index is a graphical method of determining the number of
clusters.
##                 In the plot of Hubert index, we seek a significant knee
that corresponds to a
##                 significant increase of the value of the measure i.e the
significant peak in Hubert
##       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



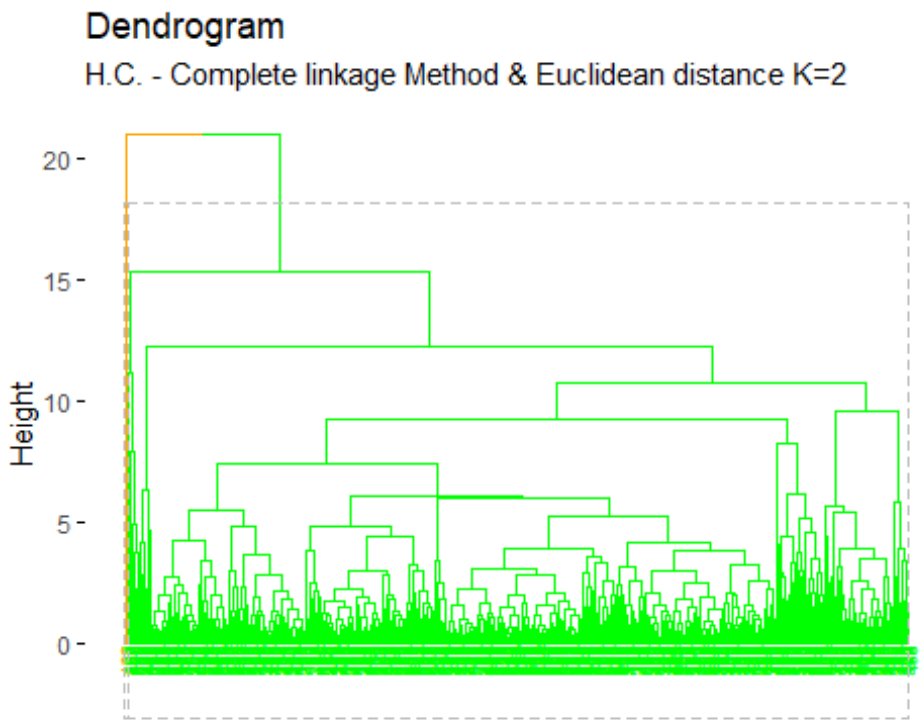

```
## *** : The D index is a graphical method of determining the number of
clusters.
##                 In the plot of D index, we seek a significant knee (the
significant peak in Dindex
##                 second differences plot) that corresponds to a significant
increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 8 proposed 2 as the best number of clusters
## * 2 proposed 3 as the best number of clusters
## * 3 proposed 5 as the best number of clusters
## * 2 proposed 6 as the best number of clusters
## * 2 proposed 8 as the best number of clusters
## * 5 proposed 9 as the best number of clusters
## * 2 proposed 10 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************
```

```r
fviz_nbclust(nb) +
  labs(subtitle = "H.C. - Complete linkage Method & Euclidean Distance",
       cex.sub= 0.5)
```

```
## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : the condition has length > 1 and only the first
element
## will be used

## Warning in if (class(best_nc) == "matrix") .viz_NbClust(x, print.summary,
: the
## condition has length > 1 and only the first element will be used

## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : the condition has length > 1 and only the first
element
## will be used

## Warning in if (class(best_nc) == "matrix") {: the condition has length > 1
and
## only the first element will be used

## Among all indices:
## ===================
## * 2 proposed  0 as the best number of clusters
## * 8 proposed  2 as the best number of clusters
## * 2 proposed  3 as the best number of clusters
## * 3 proposed  5 as the best number of clusters
## * 2 proposed  6 as the best number of clusters
## * 2 proposed  8 as the best number of clusters
## * 5 proposed  9 as the best number of clusters
## * 2 proposed  10 as the best number of clusters
##
## Conclusion
## =========================
## * According to the majority rule, the best number of clusters is  2 .
```




```
hc <- hclust(dist.eucl, method = "complete")
grp <- cutree(hc, k=2)
table(grp)

## grp
##   1   2
## 577   2

fviz_dend(hc, k = 2, cex = 0.5, k_colors = c("orange", "green"),
          color_labels_by_k = TRUE, rect = TRUE) + labs(title = "Dendrogram",
          subtitle = "H.C. - Complete linkage Method & Euclidean distance
K=2",
          cex.subtitle= 0.5)
```

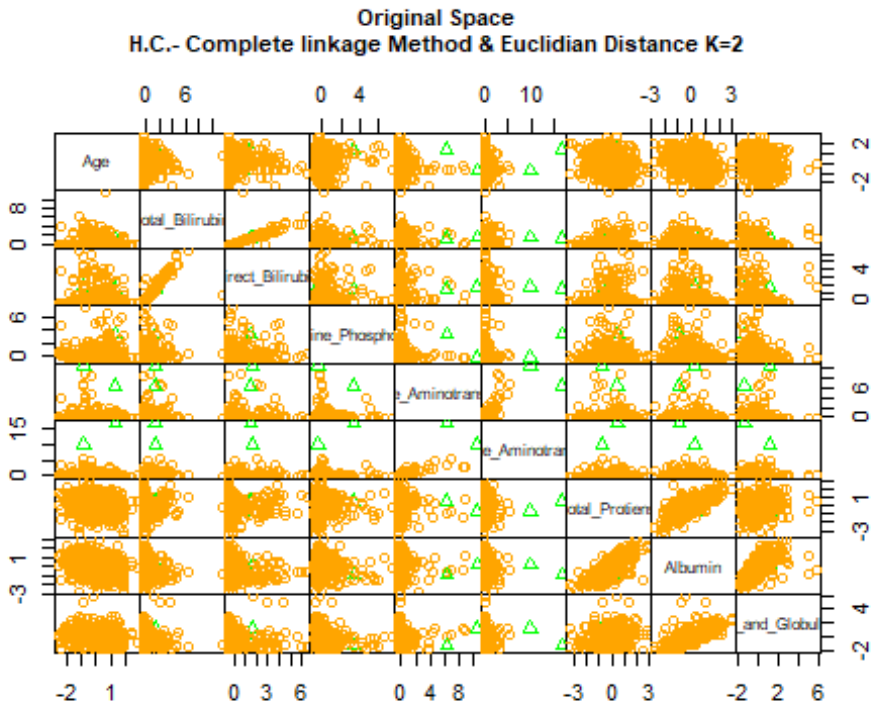



```
cor(dist.eucl, cophenetic(hc))
```

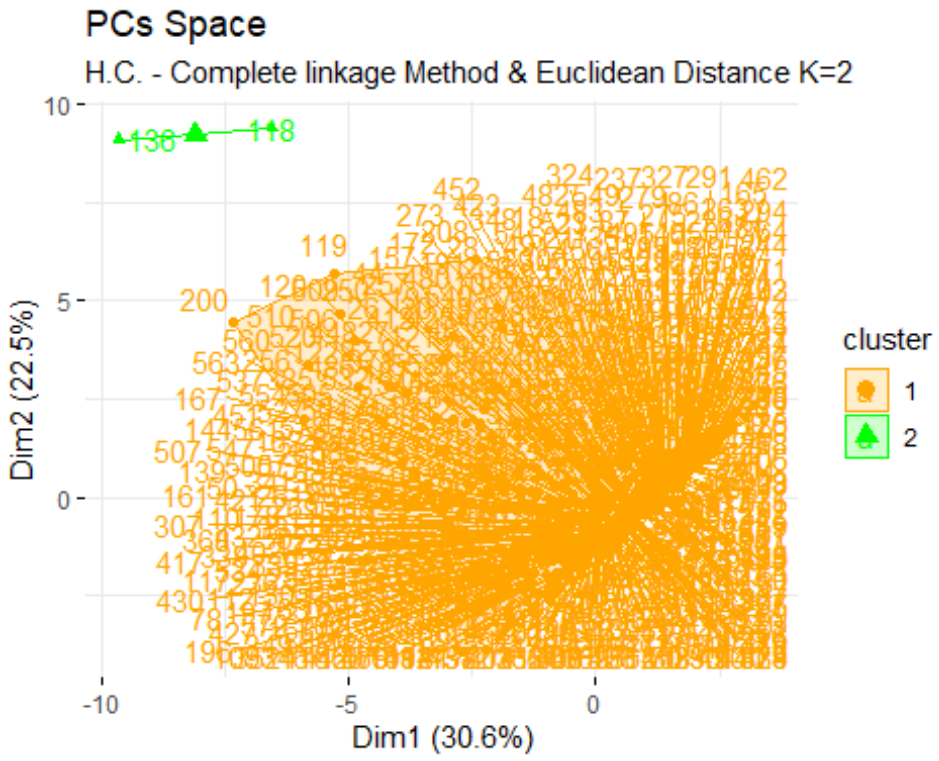
```
## [1] 0.76228
```

According to the result of "NbClust", the best number of clusters, applying hierarchical clustering using complete linkage method and euclidean distance is 2.

```
pairs(liver_scale, gap=0, pch=grp, cex.main= 0.7,
main="Original Space\nH.C.- Complete linkage Method & Euclidian Distance
K=2",
col=c("orange", "green")[grp])
```




```
fviz_cluster(list(data = liver_scale, cluster = grp),
             palette = c("orange", "green"), ellipse.type = "convex",
             main="PCs Space", repel = TRUE, show.clust.aver = FALSE,
             ggtheme = theme_minimal()) +
  labs(subtitle = "H.C. - Complete linkage Method & Euclidean Distance K=2",
       cex.sub= 0.5)
```

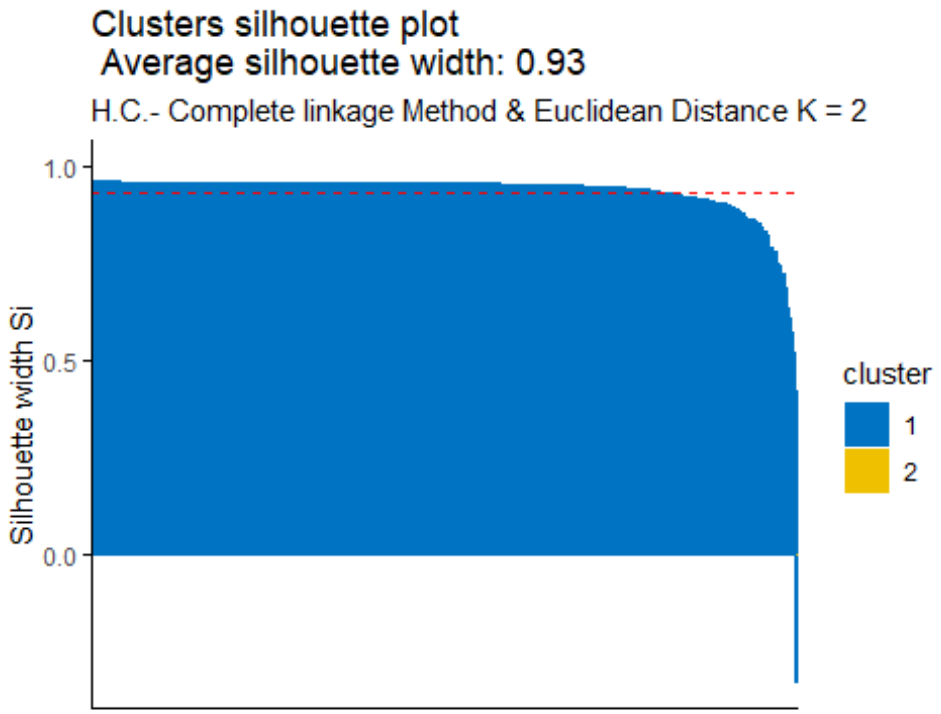


To evaluate the goodness of clustering algorithm results, internal and external validation measures will be analyzed as follows:

Internal validation measures

Silhouette width

```
hclust<- eclust(liver_sub, k = 2, "hclust", hc_method = "complete",
                nboot = 50, hc_metric = "euclidean")
silinfo <- hclust$silinfo
silinfo$avg.width

## [1] 0.9341632

fviz_silhouette(hclust, palette = "jco", ggtheme = theme_classic()) +
labs(subtitle = "H.C.- Complete linkage Method & Euclidean Distance K = 2",
cex.sub= 0.5)

##   cluster size ave.sil.width
## 1       1  578          0.94
## 2       2    1          0.00
```




```
silinfo$clus.avg.widths
```

```
## [1] 0.9357794 0.0000000
```

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

```
##     cluster neighbor  sil_width
## 118       1        2 -0.3302797
```

The value of complete silhouette width indicates that on average the units are well enough clustered. In cluster 1 (blue cluster) the units are on average the same silhouette value with respect to the silhouette width, in cluster 2 (the yellow one) the units are on average having the lower silhouette value with respect to silhouette width. According to the index, unit that belong to cluster 1 are not well clustered, it should belong to the neighbor cluster 2.

Dunn Index

```
stats <- cluster.stats(dist(liver_scale), hclust$cluster)
stats$dunn
```

```
## [1] 0.5317513
```

According to the Dunn index, the units are not clustered well enough.

External Validation Measures

Confusion Matrix

According to the Confusion matrix, the number of clusters is equal to the nominal values.

```
table(liver$Liver_Disease, hclust$cluster)
```

```
##
##       1   2
##   0 413   1
##   1 165   0
```

For the liver disease, data has been classified mostly in cluster 1, 578 units in cluster 1, and 1 in cluster 2. For the patients having liver disease classified mostly in cluster 1 while cluster 2 has 0 values. Safe to say data are not well balanced in both cluster.

Correct Rand Index

```
liver.disease <- as.numeric(liver$Liver_Disease)
stats<- cluster.stats(d = dist(liver_scale), liver.disease, hclust$cluster)
stats$corrected.rand
```

```
## [1] -0.002074324
```

According to the Correct Rand Index, there is no agreement between the numerical values and the cluster solution. From -1 to +1, the agreement is very close to 0.

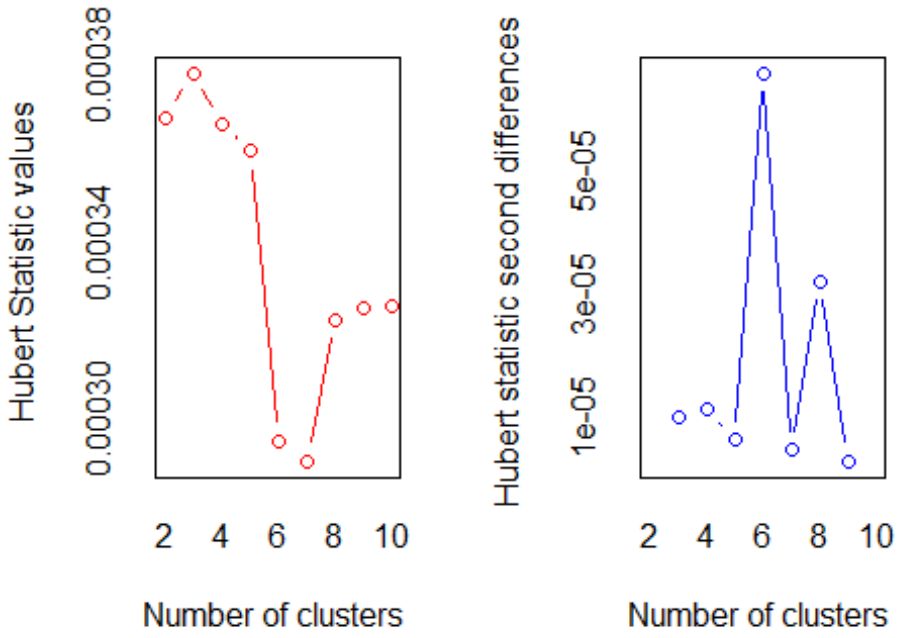
Meila's VI Index

```
stats$vi
```

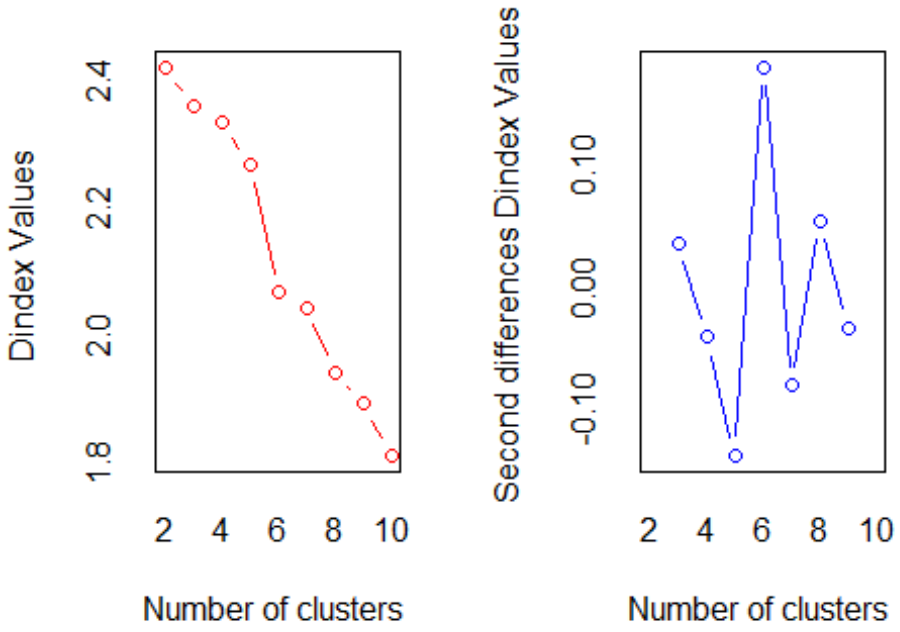
```
## [1] 0.6091425
```

Complete linkage Method & Manhattan Distance

```
nb <- NbClust(liver_scale, distance = "manhattan", min.nc = 2, max.nc = 10,
              method = "complete")
```



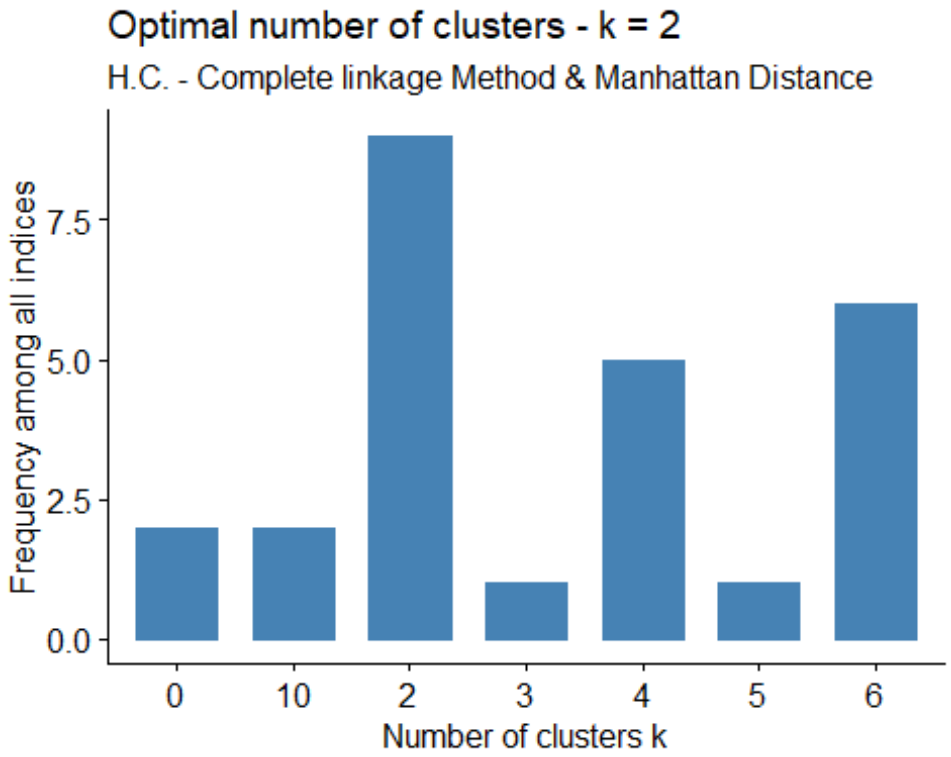

```
## *** : The Hubert index is a graphical method of determining the number of
clusters.
##                 In the plot of Hubert index, we seek a significant knee
that corresponds to a
##                 significant increase of the value of the measure i.e the
significant peak in Hubert
##       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



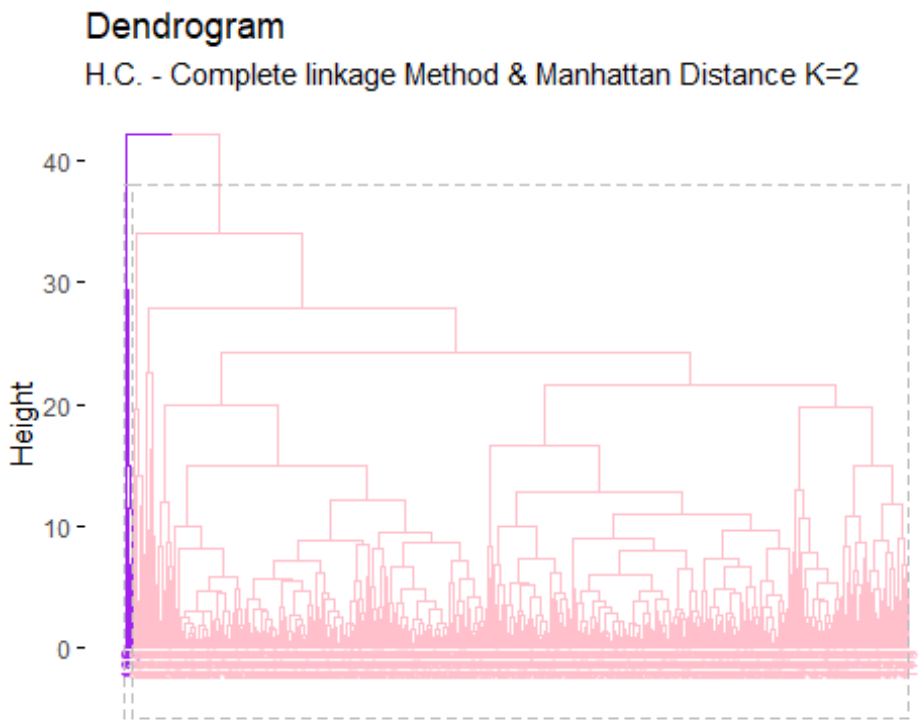

```
## *** : The D index is a graphical method of determining the number of
clusters.
##                 In the plot of D index, we seek a significant knee (the
significant peak in Dindex
##                 second differences plot) that corresponds to a significant
increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 9 proposed 2 as the best number of clusters
## * 1 proposed 3 as the best number of clusters
## * 5 proposed 4 as the best number of clusters
## * 1 proposed 5 as the best number of clusters
## * 6 proposed 6 as the best number of clusters
## * 2 proposed 10 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************
```

```
fviz_nbclust(nb) +
  labs(subtitle = "H.C. - Complete linkage Method & Manhattan Distance",
       cex.sub= 0.5)
```

```
## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : the condition has length > 1 and only the first
element
## will be used

## Warning in if (class(best_nc) == "matrix") .viz_NbClust(x, print.summary,
: the
## condition has length > 1 and only the first element will be used

## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : the condition has length > 1 and only the first
element
## will be used

## Warning in if (class(best_nc) == "matrix") {: the condition has length > 1
and
## only the first element will be used

## Among all indices:
## ===================
## * 2 proposed  0 as the best number of clusters
## * 9 proposed  2 as the best number of clusters
## * 1 proposed  3 as the best number of clusters
## * 5 proposed  4 as the best number of clusters
## * 1 proposed  5 as the best number of clusters
## * 6 proposed  6 as the best number of clusters
## * 2 proposed  10 as the best number of clusters
##
## Conclusion
## =========================
## * According to the majority rule, the best number of clusters is  2 .
```



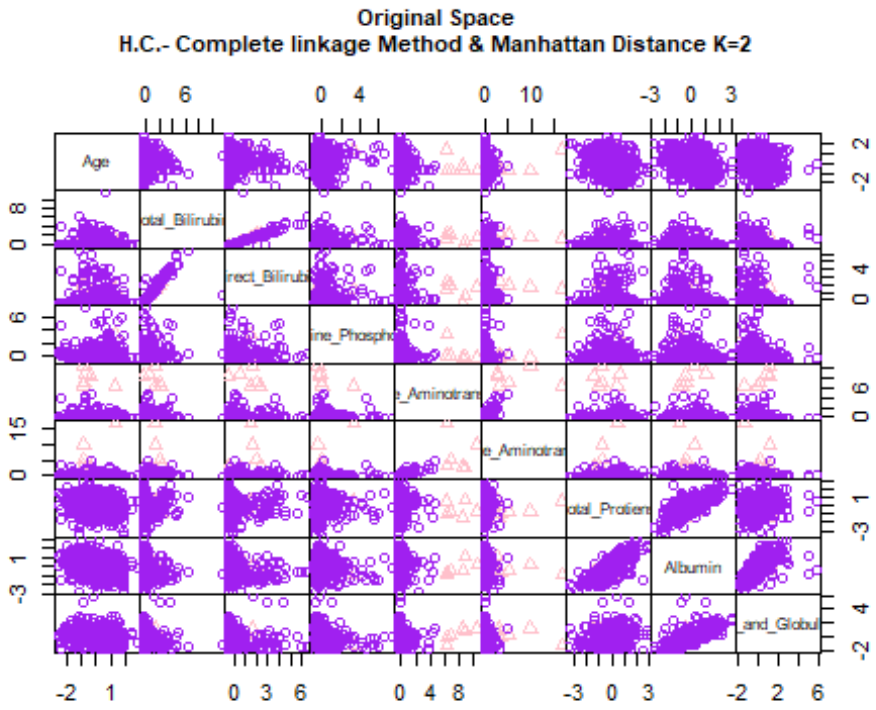

```r
dist.man <- dist(liver_scale, method = "manhattan")
hc <- hclust(dist.man, method = "complete")
grp <- cutree(hc, k=2)
table(grp)

## grp
##   1   2
## 573   6

fviz_dend(hc, k = 2, cex = 0.5, k_colors = c("purple", "pink"),
          r_labels_by_k = TRUE, rect = TRUE) + labs(title = "Dendrogram",
          subtitle = "H.C. - Complete linkage Method & Manhattan Distance
K=2",
          cex.subtitle= 0.5)
```



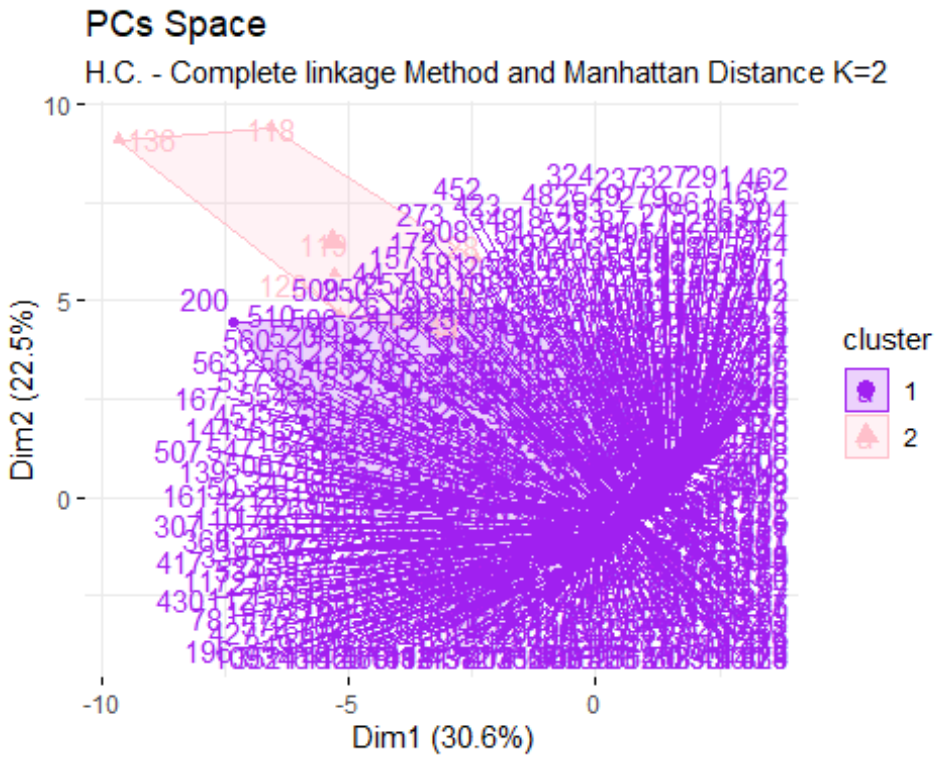

```
cor(dist.man, cophenetic(hc))

## [1] 0.6077513
```

According to the result of "NbClust", the best number of clusters, applying hierarchical clustering using complete linkage method and manhattan distance is 2.

```
pairs(liver_scale, gap=0, pch=grp, cex.main= 0.7,
      main="Original Space\nH.C.- Complete linkage Method & Manhattan
Distance K=2",
      col=c("purple", "pink")[grp])
```




```
fviz_cluster(list(data = liver_scale, cluster = grp),  palette = c("purple",
"pink"),
             ellipse.type = "convex", main="PCs Space", repel = TRUE,
             show.clust.aver = FALSE, ggtheme = theme_minimal()) +
  labs(subtitle = "H.C. - Complete linkage Method and Manhattan Distance
K=2",
       cex.sub= 0.5)
```

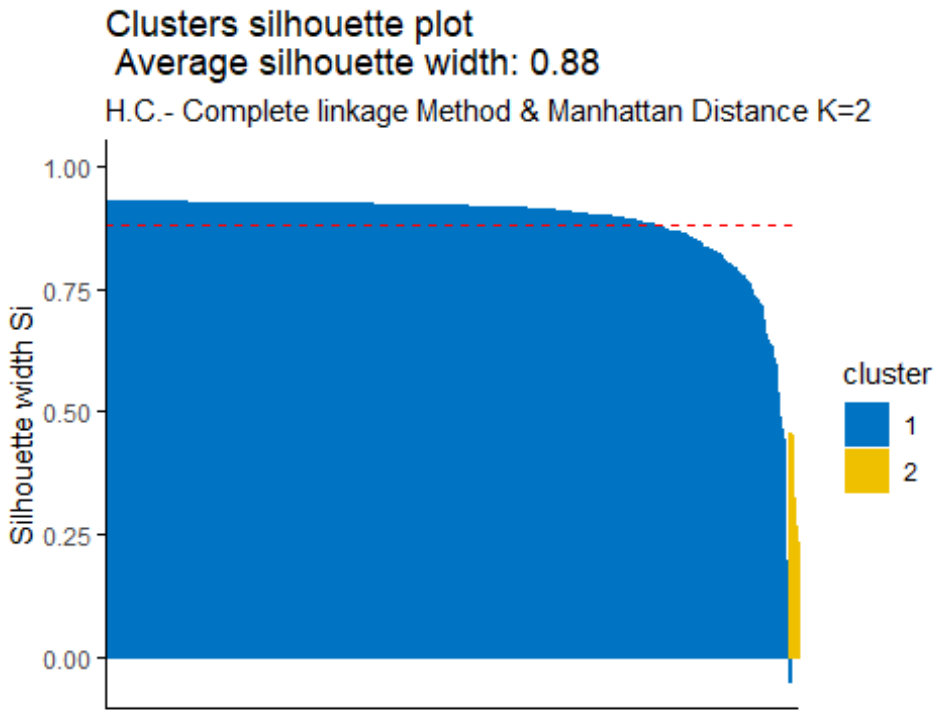


To evaluate the goodness of clustering algorithm results, internal and external validation measures will be analyzed as follows:

Internal Validation Measures

Silhouette width

```
hclust<- eclust(liver_sub, k=2, "hclust", hc_method = "complete",
                nboot = 50, hc_metric = "manhattan")
silinfo <- hclust$silinfo
silinfo$avg.width

## [1] 0.8807343

fviz_silhouette(hclust, palette = "jco", ggtheme = theme_classic()) +
labs(subtitle = "H.C.- Complete linkage Method & Manhattan Distance K=2",
cex.sub= 0.5)

##   cluster size ave.sil.width
## 1       1  572          0.89
## 2       2    7          0.33
```




```
silinfo$clus.avg.widths

## [1] 0.8875222 0.3260695

sil <- hclust$silinfo$widths[, 1:3]
neg_sil_index_aver.eu<- which(sil[, "sil_width"] < 0)
sil[neg_sil_index_aver.eu, , drop = FALSE]

##     cluster neighbor   sil_width
## 200       1        2 -0.05135232
```

The value of complete silhouette width indicates that on average the units are well enough clustered. In cluster 1 (blue cluster) the units are on average the same silhouette value with respect to the silhouette width, in cluster 2 (the yellow one) the units are on average having the lower silhouette value with respect to silhouette width. According to the index, unit that belong to cluster 1 are not well clustered, it should belong to the neighbor cluster 2.

Dunn Index

```
stats <- cluster.stats(dist(liver_scale), hclust$cluster)
stats$dunn

## [1] 0.2188165
```

According to the Dunn index, the units are not clustered well enough.

External Validation Measures

Confusion Matrix

According to the Confusion matrix, the number of clusters is equal to nominal values. The clusters found are 2 and the nominal variable can take 2 possible values.

```
table(liver$Liver_Disease, hclust$cluster)

##
##       1   2
##   0 407   7
##   1 165   0
```

For the liver disease, data has been classified mostly in cluster 1, 572 units in cluster 1, and 7 in cluster 2. For the patients having liver disease classified mostly in cluster 1 while cluster 2 has 0 values. Safe to say data are not well balanced in both cluster.

Correct Rand Index

```
liver.disease <- as.numeric(liver$Liver_Disease)
stats <- cluster.stats(d = dist(liver_scale), liver.disease, hclust$cluster)
stats$corrected.rand

## [1] -0.01389137
```

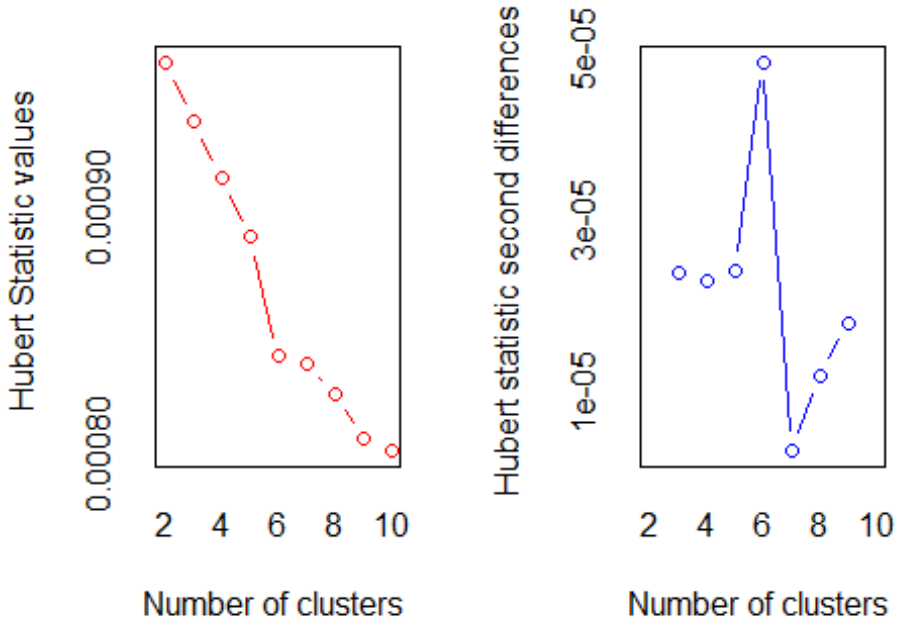
According to the Correct Rand Index, there is no agreement between the numerical value and the cluster. solution. From -1 to +1, the agreement is very close to 0.

Meila's VI Index

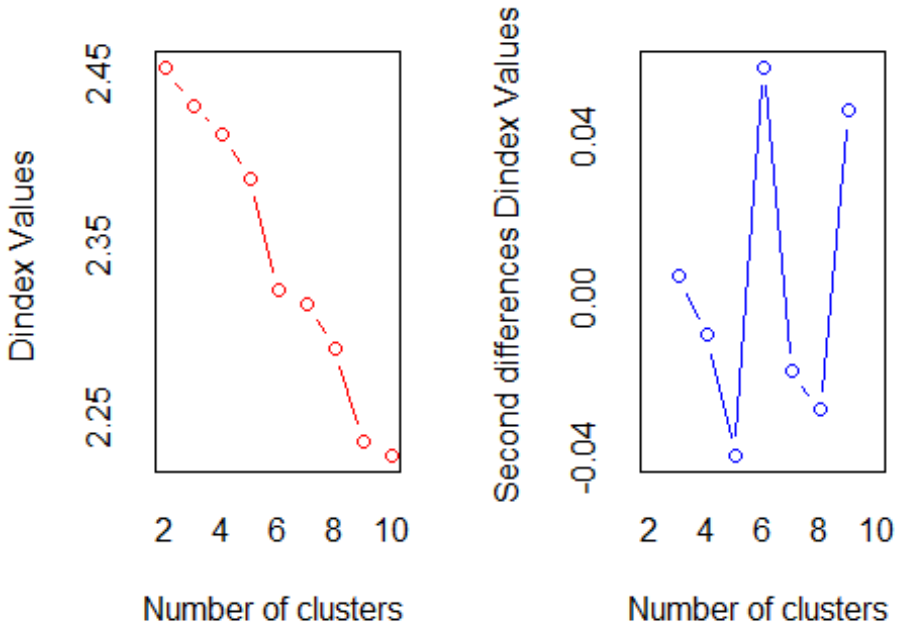
```
stats$vi

## [1] 0.6548182
```

Centroid Linkage Method & Euclidean Distance

```
library(NbClust)
nb <- NbClust(liver_scale, distance = "euclidean", min.nc = 2, max.nc = 10,
              method = "centroid")

## Warning in pf(beale, pp, df2): NaNs produced

## Warning in pf(beale, pp, df2): NaNs produced

## [1] "Frey index : No clustering structure in this data set"
```




```
## *** : 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:
## * 8 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
## * 3 proposed 6 as the best number of clusters
## * 1 proposed 7 as the best number of clusters
## * 3 proposed 9 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************
```

```
library(factoextra)
fviz_nbclust(nb) +
```

```
  labs(subtitle = "H.C. - Centroid linkage Method & Euclidian Distance",
       cex.sub= 0.5)
```

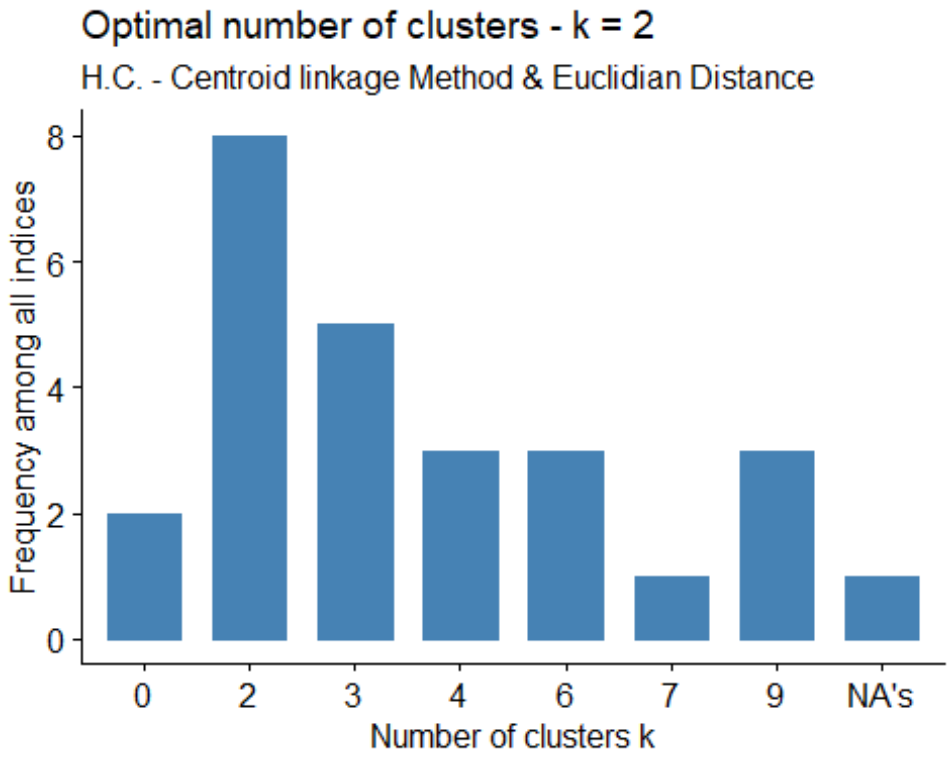
```
## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : the condition has length > 1 and only the first
element
## will be used
```

```
## Warning in if (class(best_nc) == "matrix") .viz_NbClust(x, print.summary,
: the
## condition has length > 1 and only the first element will be used
```

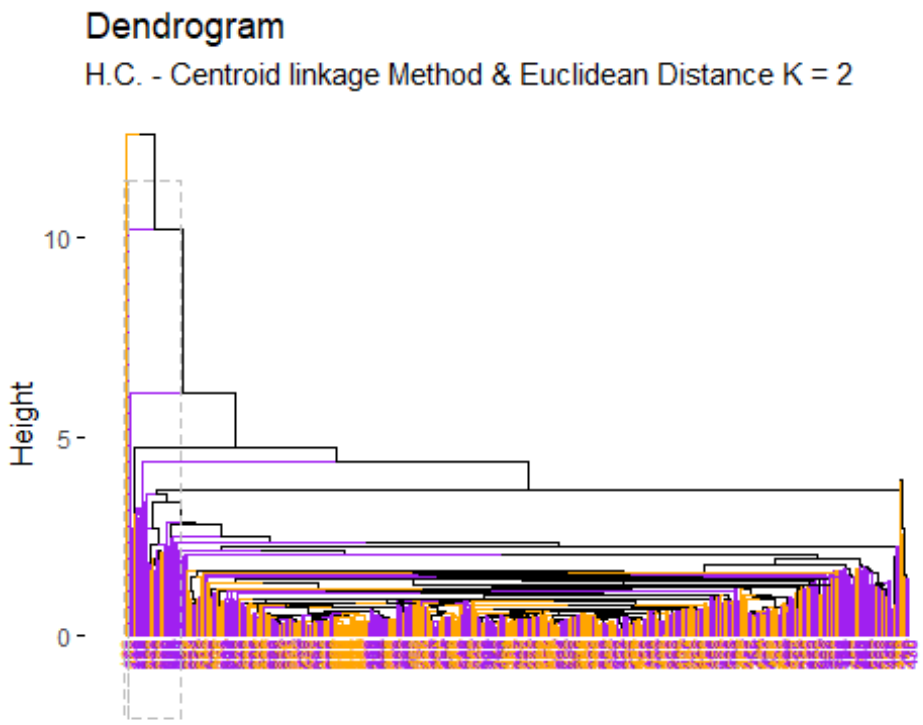
```
## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : the condition has length > 1 and only the first
element
## will be used
```

```
## Warning in if (class(best_nc) == "matrix") {: the condition has length > 1
and
## only the first element will be used
```

```
## Among all indices:
## ===================
## * 2 proposed  0 as the best number of clusters
## * 8 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
## * 3 proposed  6 as the best number of clusters
## * 1 proposed  7 as the best number of clusters
## * 3 proposed  9 as the best number of clusters
## * 1 proposed  NA's as the best number of clusters
##
## Conclusion
## =========================
## * According to the majority rule, the best number of clusters is  2 .
```



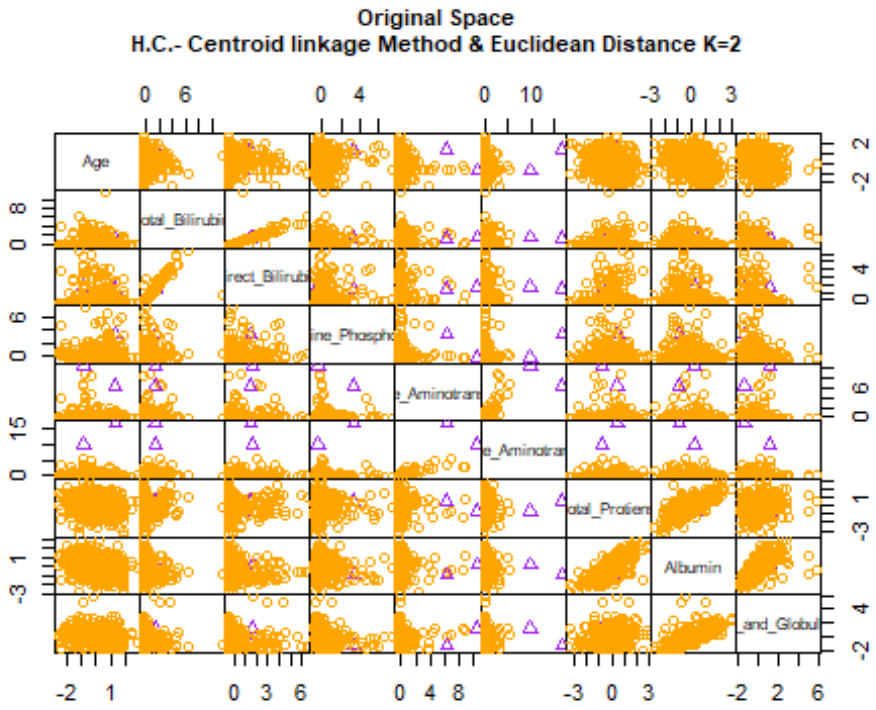

```
dist.eucl <- dist(liver_scale, method = "euclidian")
hc <- hclust(dist.eucl, method = "centroid")
grp <- cutree(hc, k=2)
table(grp)

## grp
##   1   2
## 577   2

fviz_dend(hc, k = 2, cex = 0.5, k_colors = c("orange", "purple"),
          color_labels_by_k = TRUE, rect = TRUE) + labs(title = "Dendrogram",
          subtitle = "H.C. - Centroid linkage Method & Euclidean Distance K =
2",
          cex.subtitle= 0.5)

## Warning in get_col(col, k): Length of color vector was shorter than the
number
## of clusters - color vector was recycled
```



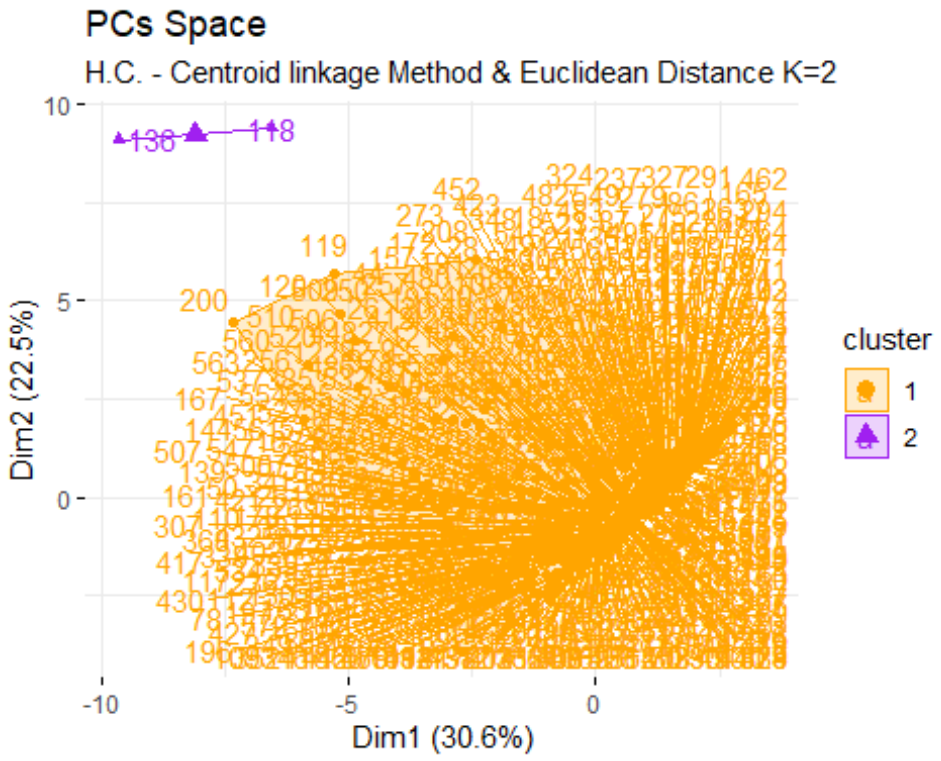

```
cor(dist.eucl, cophenetic(hc))

## [1] 0.8727531
```

According to the result of "NbClust", the best number of clusters, applying hierarchical clustering using the Centroid linkage method and euclidean distance is 2.

```
pairs(liver_scale, gap=0, pch=grp, cex.main= 0.7,
      main="Original Space\nH.C.- Centroid linkage Method & Euclidean
Distance K=2",
      col=c("orange", "purple")[grp])
```




```
fviz_cluster(list(data = liver_scale, cluster = grp),
             palette = c("orange", "purple"), ellipse.type = "convex",
             main="PCs Space", repel = TRUE, show.clust.aver = FALSE,
             ggtheme = theme_minimal()) +
  labs(subtitle = "H.C. - Centroid linkage Method & Euclidean Distance K=2",
       cex.sub= 0.5)
```

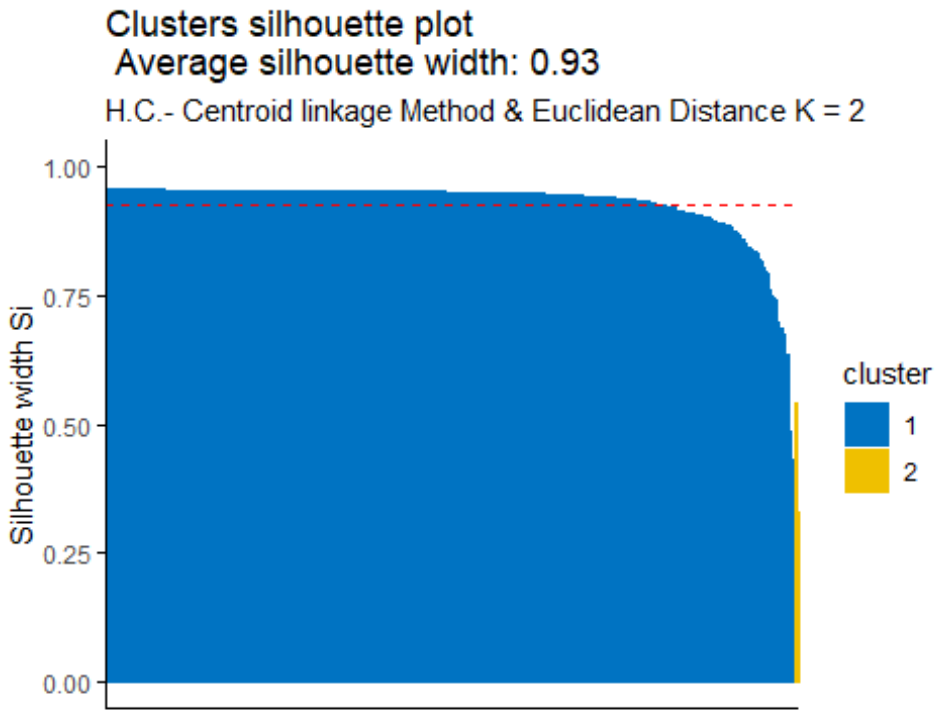


To evaluate the goodness of clustering algorithm results, internal and external validation measures will be analyzed as follows:

Internal Validation Measures

Silhouette width

```
hclust<- eclust(liver_sub, k = 2, "hclust", hc_method = "centroid",
                nboot = 50, hc_metric = "euclidean")

## Warning in get_col(col, k): Length of color vector was shorter than the
number
## of clusters - color vector was recycled

silinfo <- hclust$silinfo
silinfo$avg.width

## [1] 0.9252641

fviz_silhouette(hclust, palette = "jco", ggtheme = theme_classic()) +
  labs(subtitle = "H.C.- Centroid linkage Method & Euclidean Distance K = 2",
       cex.sub= 0.5)

##   cluster size ave.sil.width
## 1       1  577          0.93
## 2       2    2          0.44
```




```
silinfo$clus.avg.widths
```

```
## [1] 0.9269589 0.4363246
```

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

```
## [1] cluster   neighbor  sil_width
## <0 rows> (or 0-length row.names)
```

The value of complete silhouette width indicates that on average the units are well enough clustered. In cluster 1 (blue cluster) the units are on average the same silhouette value with respect to the silhouette width, in cluster 2 (the yellow one) the units are on average having the lower silhouette value with respect to silhouette width.

Dunn Index

```
library(fpc)
stats <- cluster.stats(dist(liver_scale), hclust$cluster)
stats$dunn
```

```
## [1] 0.3913763
```

According to the Dunn index, the units are not clustered well enough.

External Validation Measures

Confusion Matrix

According to the Confusion matrix, the number of clusters is equal to the nominal values.

```
table(liver$Liver_Disease, hclust$cluster)
```

```
##
##       1   2
##   0 412   2
##   1 165   0
```

For the liver disease, data has been classified mostly in cluster 1, 577 units in cluster 1, and 2 in cluster 2. For the patients having liver disease classified mostly in cluster 1 while cluster 2 has 0 values. Safe to say data are not well balanced in both cluster.

Correct Rand Index

```
liver.disease <- as.numeric(liver$Liver_Disease)
stats<- cluster.stats(d = dist(liver_scale), liver.disease, hclust$cluster)
stats$corrected.rand
```

```
## [1] -0.004118416
```

According to the Correct Rand Index, there is no agreement between the numerical values and the cluster. Solution. From -1 to +1, the agreement is very close to 0.

Meila's VI Index

```
stats$vi
```

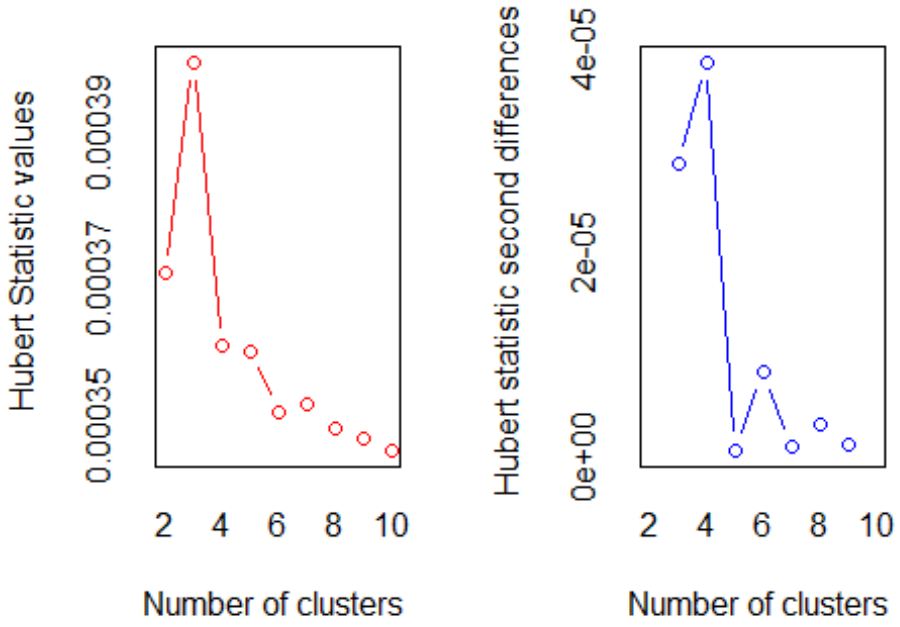
```
## [1] 0.6182953
```

Centroid linkage Method & Manhattan Distance

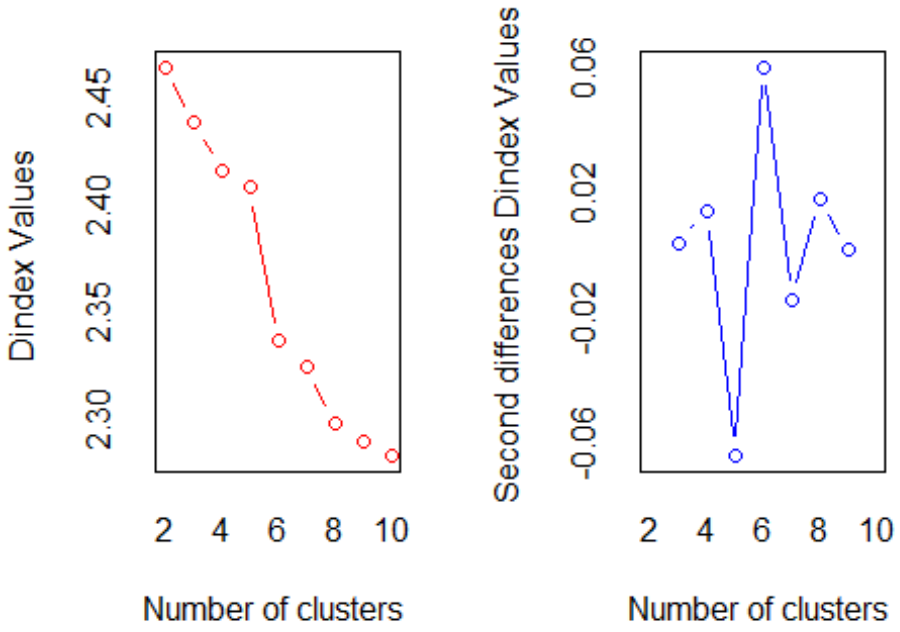
```
nb <- NbClust(liver_scale, distance = "manhattan", min.nc = 2, max.nc = 10,
              method = "centroid")
```

```
## Warning in pf(beale, pp, df2): NaNs produced
```

```
## [1] "Frey index : No clustering structure in this data set"
```



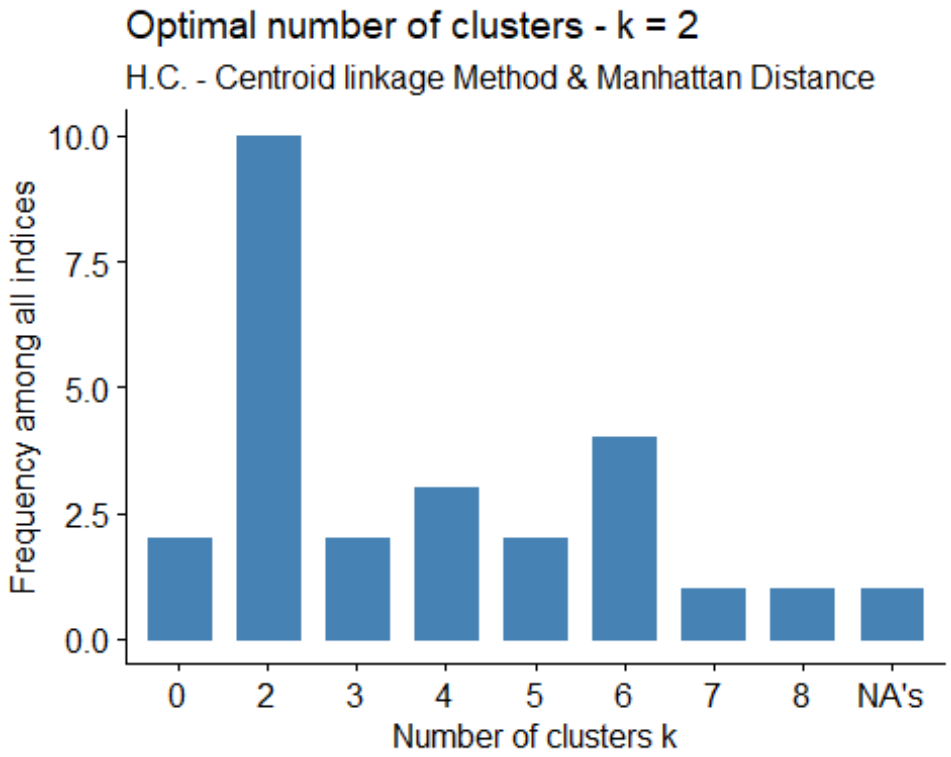

```
## *** : The Hubert index is a graphical method of determining the number of
clusters.
##                 In the plot of Hubert index, we seek a significant knee
that corresponds to a
##                 significant increase of the value of the measure i.e the
significant peak in Hubert
##       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



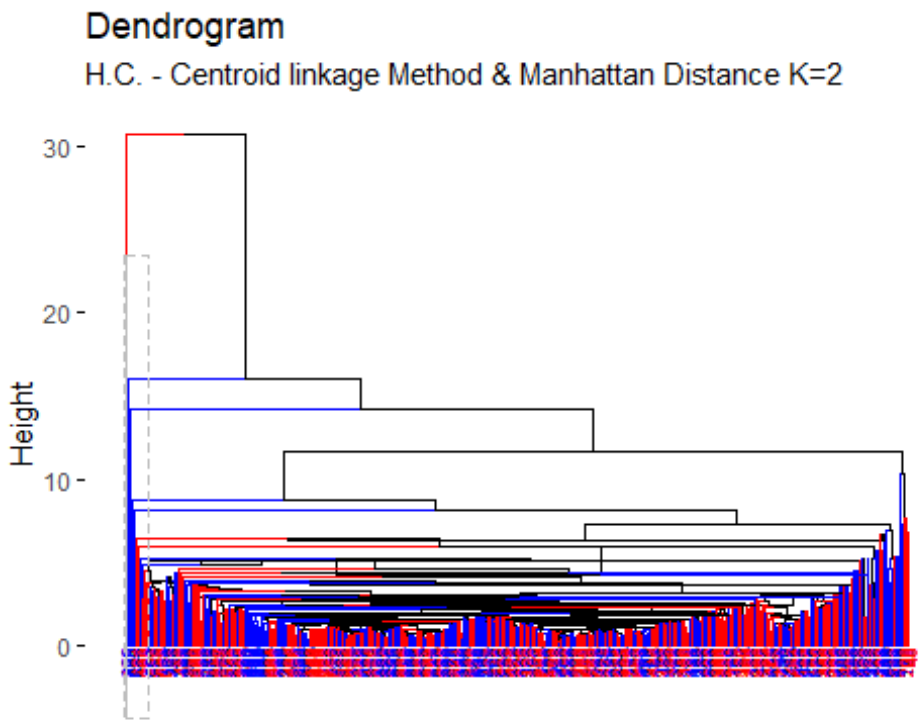

```
## *** : The D index is a graphical method of determining the number of
clusters.
##                 In the plot of D index, we seek a significant knee (the
significant peak in Dindex
##                 second differences plot) that corresponds to a significant
increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 10 proposed 2 as the best number of clusters
## * 2 proposed 3 as the best number of clusters
## * 3 proposed 4 as the best number of clusters
## * 2 proposed 5 as the best number of clusters
## * 4 proposed 6 as the best number of clusters
## * 1 proposed 7 as the best number of clusters
## * 1 proposed 8 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************
```

```r
fviz_nbclust(nb) +
  labs(subtitle = "H.C. - Centroid linkage Method & Manhattan Distance",
       cex.sub= 0.5)
```

```
## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : the condition has length > 1 and only the first
element
## will be used

## Warning in if (class(best_nc) == "matrix") .viz_NbClust(x, print.summary,
: the
## condition has length > 1 and only the first element will be used

## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : the condition has length > 1 and only the first
element
## will be used

## Warning in if (class(best_nc) == "matrix") {: the condition has length > 1
and
## only the first element will be used

## Among all indices:
## ===================
## * 2 proposed  0 as the best number of clusters
## * 10 proposed  2 as the best number of clusters
## * 2 proposed  3 as the best number of clusters
## * 3 proposed  4 as the best number of clusters
## * 2 proposed  5 as the best number of clusters
## * 4 proposed  6 as the best number of clusters
## * 1 proposed  7 as the best number of clusters
## * 1 proposed  8 as the best number of clusters
## * 1 proposed  NA's as the best number of clusters
##
## Conclusion
## ========================
## * According to the majority rule, the best number of clusters is  2 .
```




```
dist.man <- dist(liver_scale, method = "manhattan")
hc <- hclust(dist.man, method = "centroid")
grp <- cutree(hc, k = 2)
table(grp)

## grp
##   1   2
## 578   1

head(grp)

## 1 2 3 4 5 6
## 1 1 1 1 1 1

fviz_dend(hc, k = 2, cex = 0.5, k_colors = c("red", "blue"),
          color_labels_by_k = TRUE, rect = TRUE) + labs(title = "Dendrogram",
          subtitle = "H.C. - Centroid linkage Method & Manhattan Distance
K=2",
          cex.subtitle= 0.5)

## Warning in get_col(col, k): Length of color vector was shorter than the
number
## of clusters - color vector was recycled
```

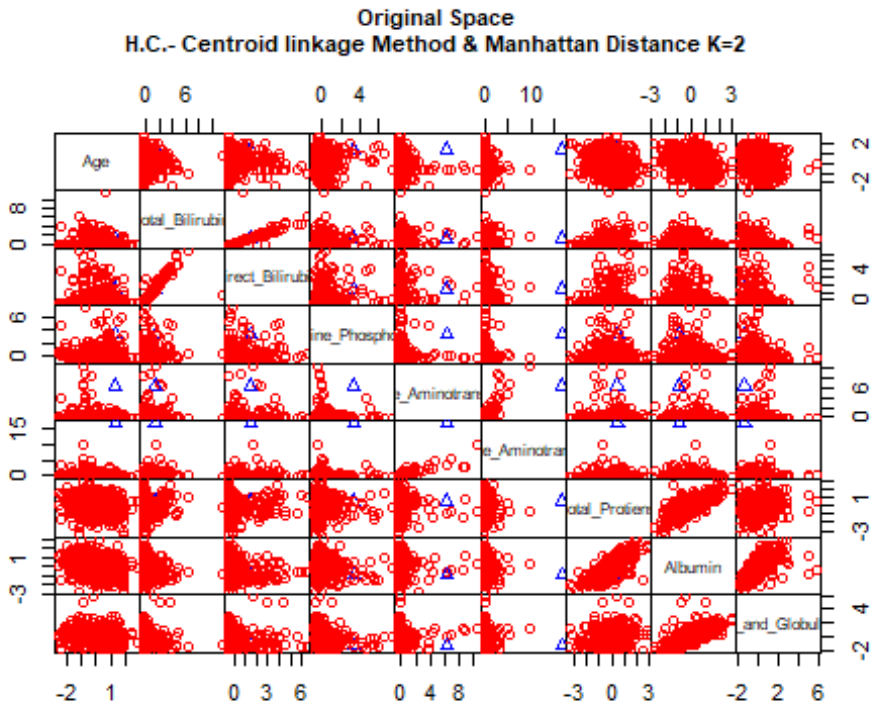



```
cor(dist.eucl, cophenetic(hc))
```

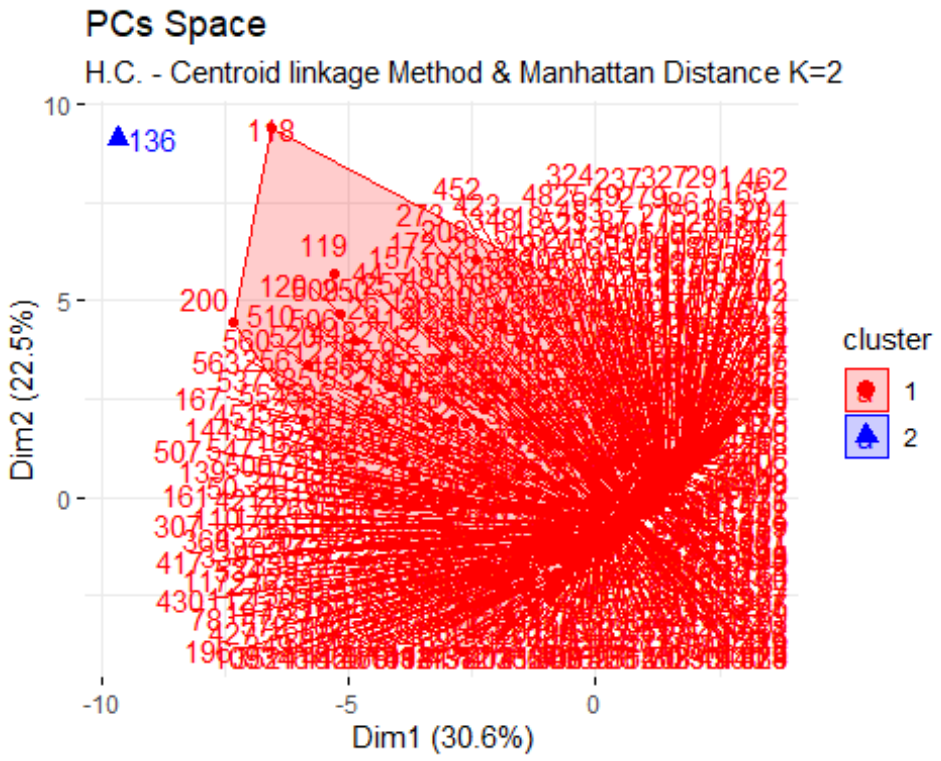
```
## [1] 0.8532441
```

According to the result of `NbClust`, the best number of clusters, applying hierarchical clustering using centroid linkage method and manhattan distance is 2.

```
pairs(liver_scale, gap=0, pch = grp, cex.main = 0.7,
      main = "Original Space\nH.C.- Centroid linkage Method & Manhattan
Distance K=2",
      col=c("red", "blue")[grp])
```




```
fviz_cluster(list(data = liver_scale, cluster = grp),
             palette = c("red", "blue"), ellipse.type = "convex",
             main = "PCs Space", repel = TRUE, show.clust.aver = FALSE,
             ggtheme = theme_minimal()) +
  labs(subtitle = "H.C. - Centroid linkage Method & Manhattan Distance K=2",
       cex.sub= 0.5)
```

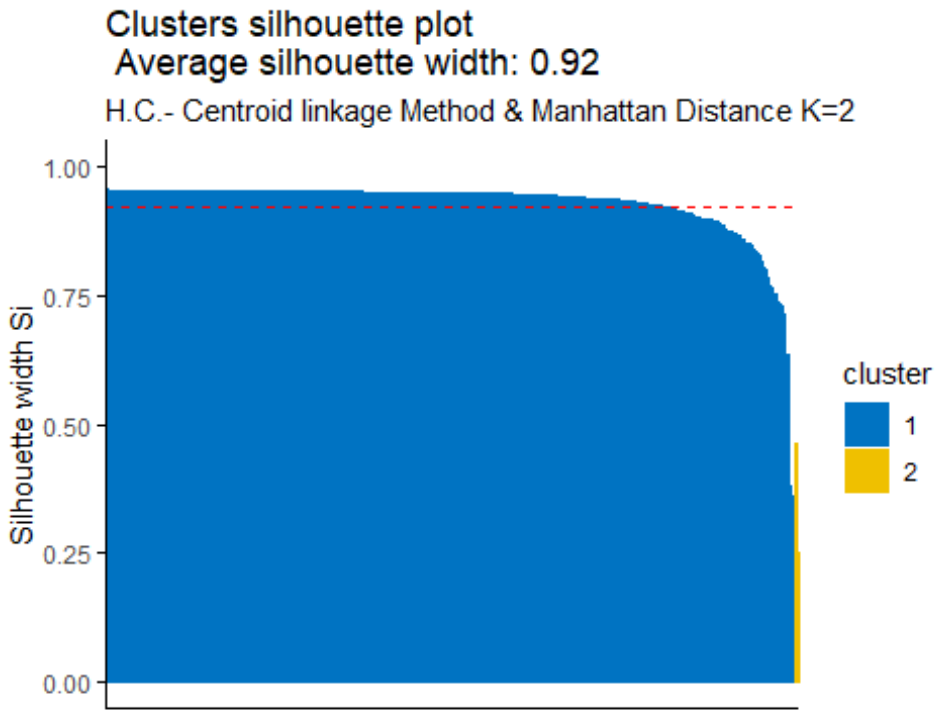


To evaluate the goodness of clustering algorithm results, internal and external validation measures will be analyzed as follows:

Internal Validation Measures

Silhouette width

```
hclust<- eclust(liver_sub, k = 2, "hclust", hc_method = "centroid",
                nboot = 50, hc_metric = "manhattan")

## Warning in get_col(col, k): Length of color vector was shorter than the
number
## of clusters - color vector was recycled

silinfo <- hclust$silinfo
silinfo$avg.width

## [1] 0.9228996

fviz_silhouette(hclust, palette = "jco", ggtheme = theme_classic()) +
  labs(subtitle = "H.C.- Centroid linkage Method & Manhattan Distance K=2",
cex.sub= 0.5)

##   cluster size ave.sil.width
## 1       1  577          0.92
## 2       2    2          0.36
```




```
silinfo$clus.avg.widths
```

```
## [1] 0.9248572 0.3581227
```

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

```
## [1] cluster   neighbor  sil_width
## <0 rows> (or 0-length row.names)
```

The value of complete silhouette width indicates that on average the units are well enough clustered. In cluster 1 (blue cluster) the units are on average the same silhouette value with respect to the silhouette width, in cluster 2 (the yellow one) the units are on average having the lower silhouette value with respect to silhouette width.

Dunn index

```
stats <- cluster.stats(dist(liver_scale), hclust$cluster)
stats$dunn
```

```
## [1] 0.3913763
```

External Validation Measures

Confusion Matrix

According to the Confusion matrix, the number of clusters is equal to nominal values. The clusters found are 2 and the nominal variable can take 2 possible values.

```
table(liver$Liver_Disease, hclust$cluster)

##
##       1   2
##   0 412   2
##   1 165   0
```

For the liver disease, data has been classified mostly in cluster 1, 577 units in cluster 1, and 2 in cluster 2. For the patients having liver disease classified mostly in cluster 1 while cluster 2 has 0 values. Safe to say data are not well balanced in both cluster.

Correct Rand Index

```
liver.disease <- as.numeric(liver$Liver_Disease)
stats <- cluster.stats(d = dist(liver_scale), liver.disease, hclust$cluster)
stats$corrected.rand

## [1] -0.004118416
```

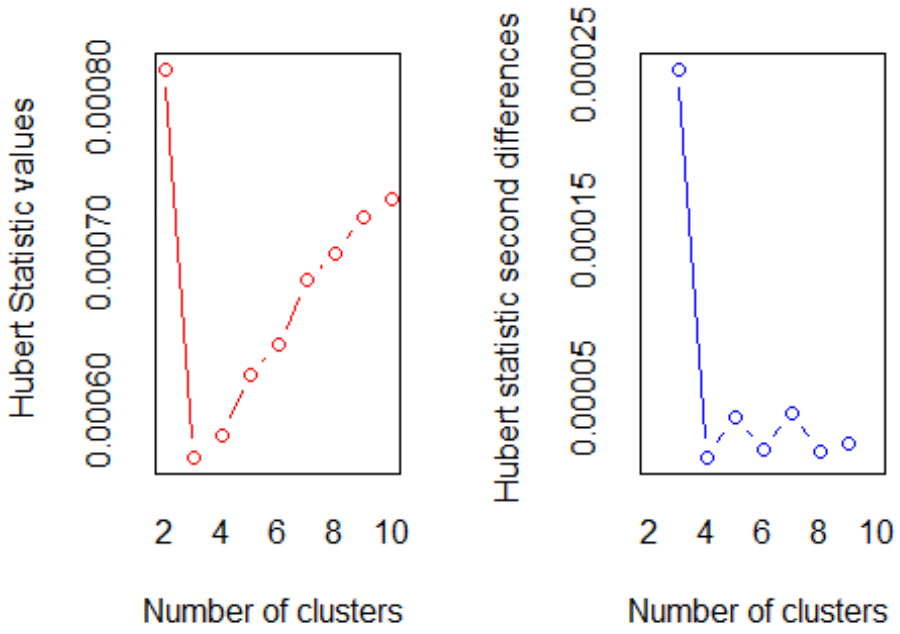
According to the Correct Rand Index, there is no agreement between the numerical value and the cluster. solution. From -1 to +1, the agreement is very close to 0.

Meila's VI Index

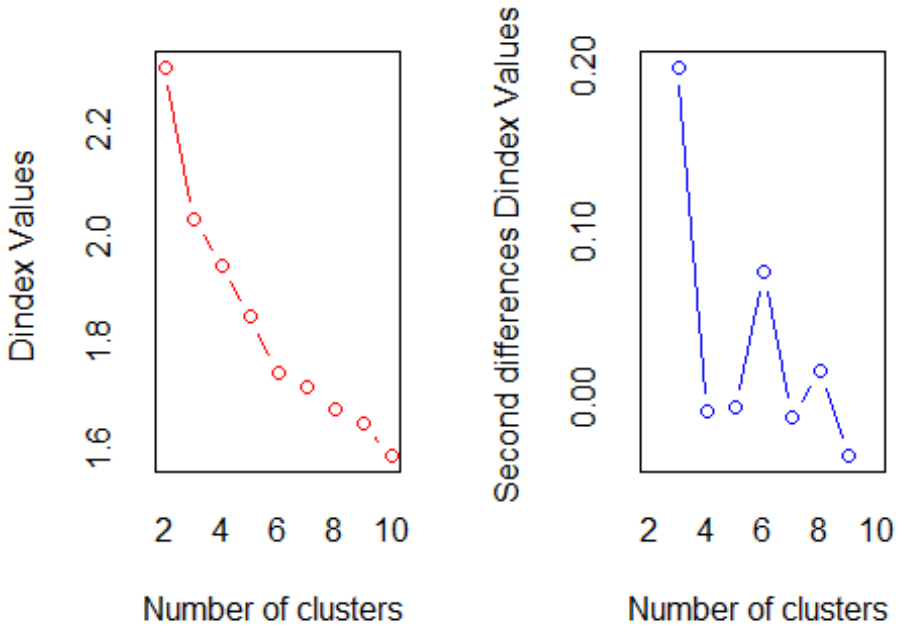
```
stats$vi

## [1] 0.6182953
```

Ward's Method - Minimum Deviance

```
nb <- NbClust(liver_scale, distance = "euclidean", min.nc = 2, max.nc = 10,
              method = "ward.D2")
```



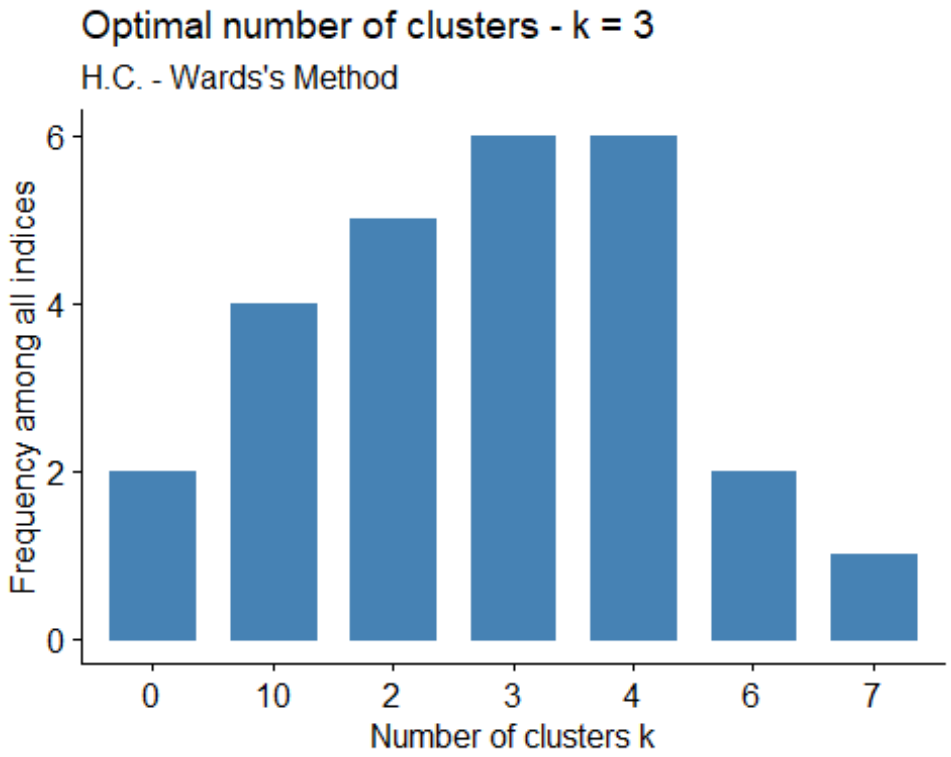

```
## *** : The Hubert index is a graphical method of determining the number of
clusters.
##                 In the plot of Hubert index, we seek a significant knee
that corresponds to a
##                 significant increase of the value of the measure i.e the
significant peak in Hubert
##       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



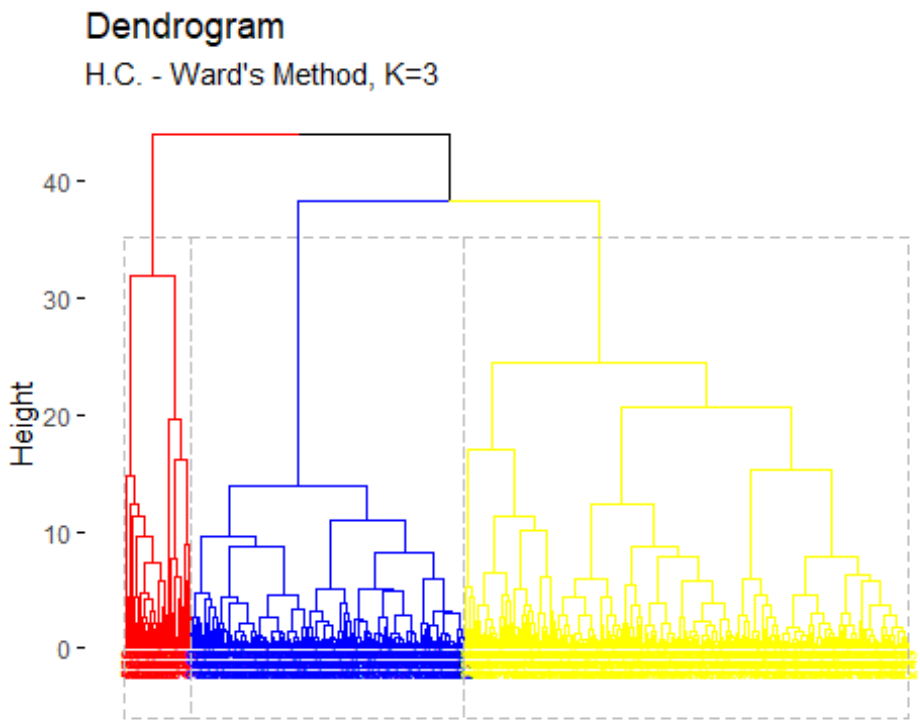

```
## *** : The D index is a graphical method of determining the number of
clusters.
##                 In the plot of D index, we seek a significant knee (the
significant peak in Dindex
##                 second differences plot) that corresponds to a significant
increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 5 proposed 2 as the best number of clusters
## * 6 proposed 3 as the best number of clusters
## * 6 proposed 4 as the best number of clusters
## * 2 proposed 6 as the best number of clusters
## * 1 proposed 7 as the best number of clusters
## * 4 proposed 10 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *******************************************************************
```

```
fviz_nbclust(nb) +
  labs(subtitle = "H.C. - Wards's Method", cex.sub= 0.5)
```

```
## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : the condition has length > 1 and only the first
element
## will be used

## Warning in if (class(best_nc) == "matrix") .viz_NbClust(x, print.summary,
: the
## condition has length > 1 and only the first element will be used

## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : the condition has length > 1 and only the first
element
## will be used

## Warning in if (class(best_nc) == "matrix") {: the condition has length > 1
and
## only the first element will be used

## Among all indices:
## ===================
## * 2 proposed  0 as the best number of clusters
## * 5 proposed  2 as the best number of clusters
## * 6 proposed  3 as the best number of clusters
## * 6 proposed  4 as the best number of clusters
## * 2 proposed  6 as the best number of clusters
## * 1 proposed  7 as the best number of clusters
## * 4 proposed  10 as the best number of clusters
##
## Conclusion
## =========================
## * According to the majority rule, the best number of clusters is  3 .
```




```
hc <- hclust(dist.eucl, method = "ward.D2")
grp <- cutree(hc, k=3)
table(grp)

## grp
##   1   2   3
## 328 202  49

head(grp)

## 1 2 3 4 5 6
## 1 1 1 1 1 2

fviz_dend(hc, k = 3, cex = 0.5, k_colors = c("red", "blue","yellow"),
          color_labels_by_k = TRUE, rect = TRUE) +
  labs(title = "Dendrogram", subtitle = "H.C. - Ward's Method, K=3",
       cex.subtitle= 0.5)
```

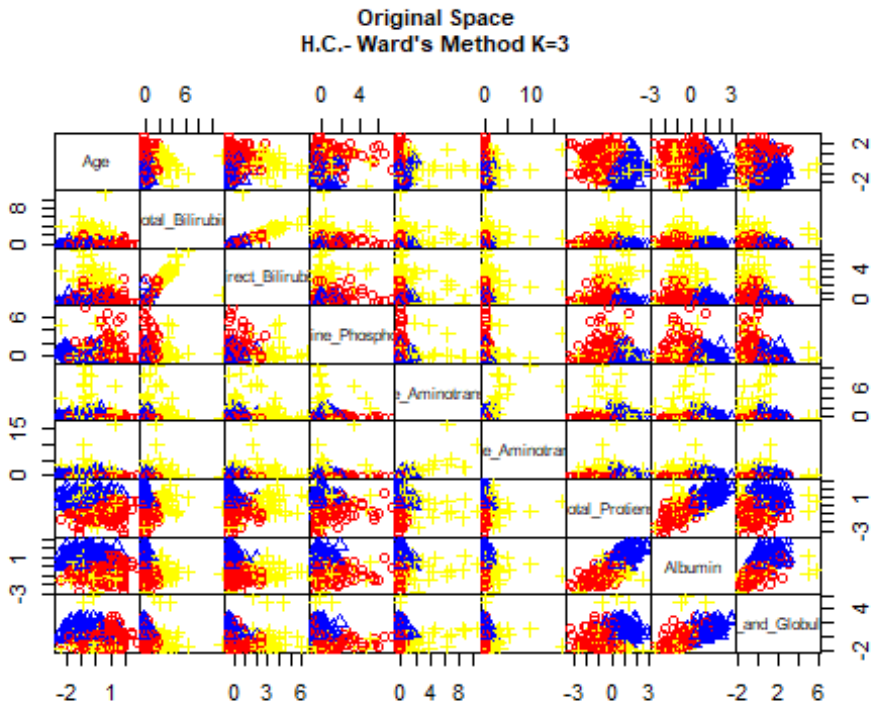



```
cor(dist.eucl, cophenetic(hc))
```

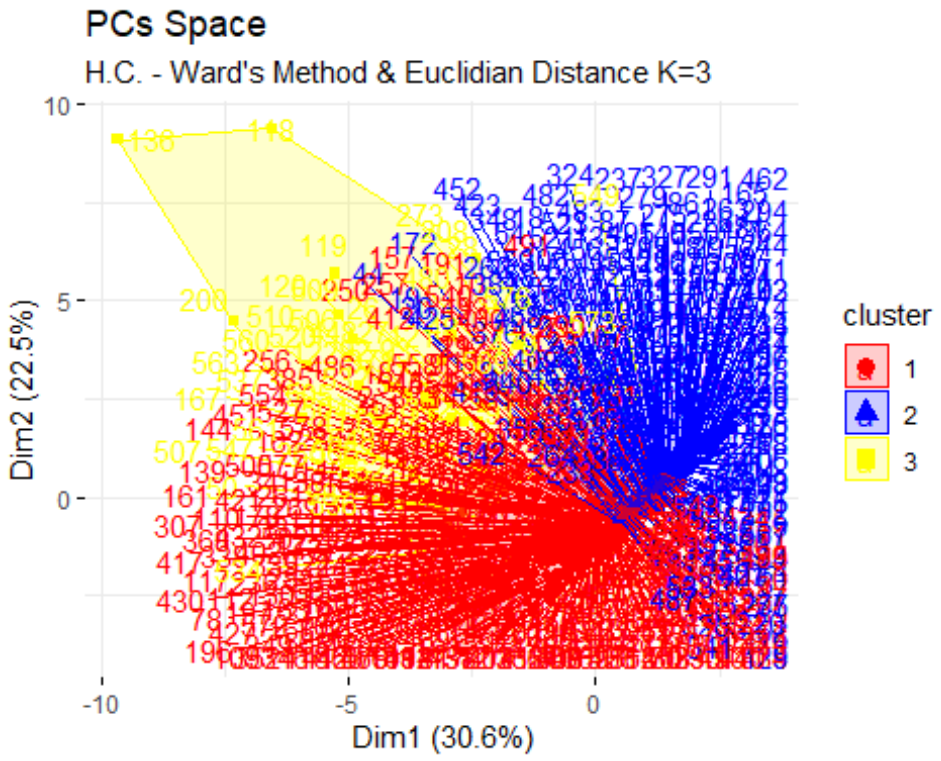
```
## [1] 0.5078094
```

According to the function `NbClust`, the best number of clusters, applying hierarchical clustering using Ward's method and euclidean distance is 3.

```
pairs(liver_scale, gap=0, pch=grp, cex.main= 0.7,
      main="Original Space\nH.C.- Ward's Method K=3",
      col=c("red", "blue", "yellow")[grp])
```




```
fviz_cluster(list(data = liver_scale, cluster = grp),
             palette = c("red", "blue", "yellow"), ellipse.type = "convex",
             main="PCs Space", repel = TRUE, show.clust.aver = FALSE,
             ggtheme = theme_minimal()) +
  labs(subtitle = "H.C. - Ward's Method & Euclidian Distance K=3", cex.sub=
0.5)
```

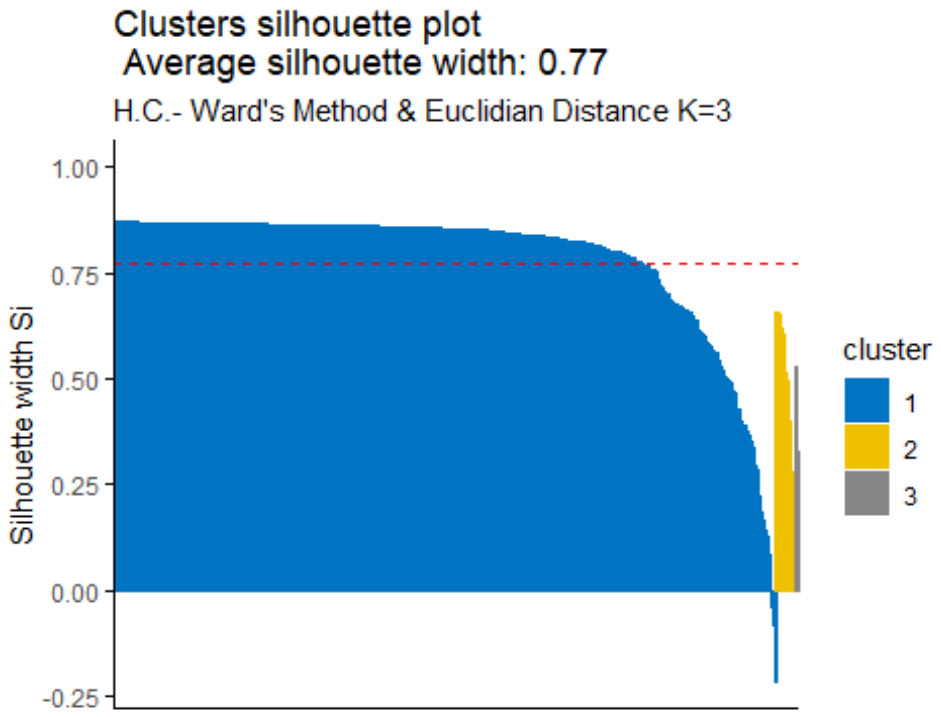


To evaluate the goodness of clustering algorithm results, internal and external validation measures will be analyzed as follows:

Internal Validation Measures

Silhouette Width

```
hclust<- eclust(liver_sub, k=3, "hclust", hc_method = "ward.D2", nboot = 50)
silinfo <- hclust$silinfo
silinfo$avg.width

## [1] 0.7738898

fviz_silhouette(hclust, palette = "jco", ggtheme = theme_classic()) +
  labs(subtitle = "H.C.- Ward's Method & Euclidian Distance K=3", cex.sub=
0.5)

##   cluster size ave.sil.width
## 1       1  560          0.78
## 2       2   17          0.48
## 3       3    2          0.43
```




```
silinfo$clus.avg.widths

## [1] 0.7840013 0.4812584 0.4300389

sil <- hclust$silinfo$widths[, 1:3]
neg_sil_index_aver.eu <- which(sil[, "sil_width"] < 0)
sil[neg_sil_index_aver.eu, , drop = FALSE]

##     cluster neighbor   sil_width
## 486       1        2 -0.04182439
## 256       1        2 -0.08167575
## 417       1        2 -0.08297610
## 81        1        2 -0.21688832
```

The value of complete silhouette width indicates that on average the units are well enough clustered. As in particular, in cluster 1 the units are on above average the silhouette value with respect to the silhouette width, in cluster 2 the units are on average having the lower silhouette value with respect to the silhouette width, and in cluster 3 on average having the lower silhouette value with respect to the silhouette width. According to above index, units that belong to cluster 1 are not well clustered, they should belong to the neighbor cluster 2.

Dunn Index

```
stats <- cluster.stats(dist(liver_scale), hclust$cluster)
stats$dunn

## [1] 0.07148566
```

According to the Dunn index, the units are not clustered well enough.

External Validation Measures

Confusion Matrix

According to the Confusion matrix, the number of clusters is equal to nominal values. The clusters found are 2 and the nominal variable can take 2 possible values.

```
table(liver$Liver_Disease, hclust$cluster)

##
##       1   2   3
##   0 396  16   2
##   1 164   1   0
```

For the liver disease, data has been classified mostly in cluster 1, 560 units in cluster 1, 17 in cluster 2, and 2 in cluster 3. For the patients having liver disease classified mostly in cluster 1 while cluster 2 has 1 value and cluster 3 has 0 value. Safe to say data are not well balanced in both cluster.

Correct Rand Index

```
liver.disease <- as.numeric(liver$Liver_Disease)
stats<- cluster.stats(d = dist(liver_scale), liver.disease, hclust$cluster)
stats$corrected.rand

## [1] -0.02867484
```

According to the Correct Rand Index, there is no agreement between the numerical value and the cluster solution. From -1 to +1, the agreement is very close to 0.

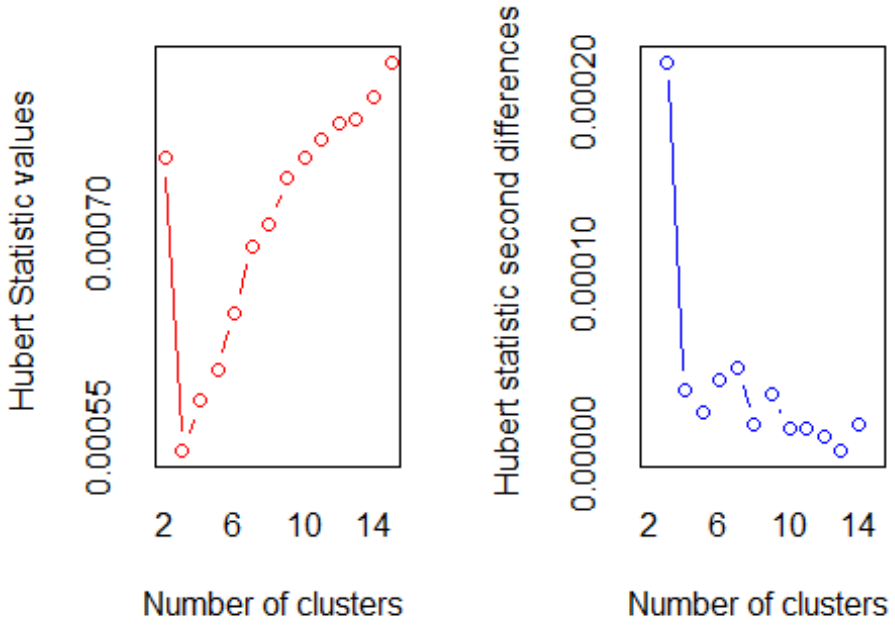
Meila's VI Index

```
stats$vi

## [1] 0.7406794
```

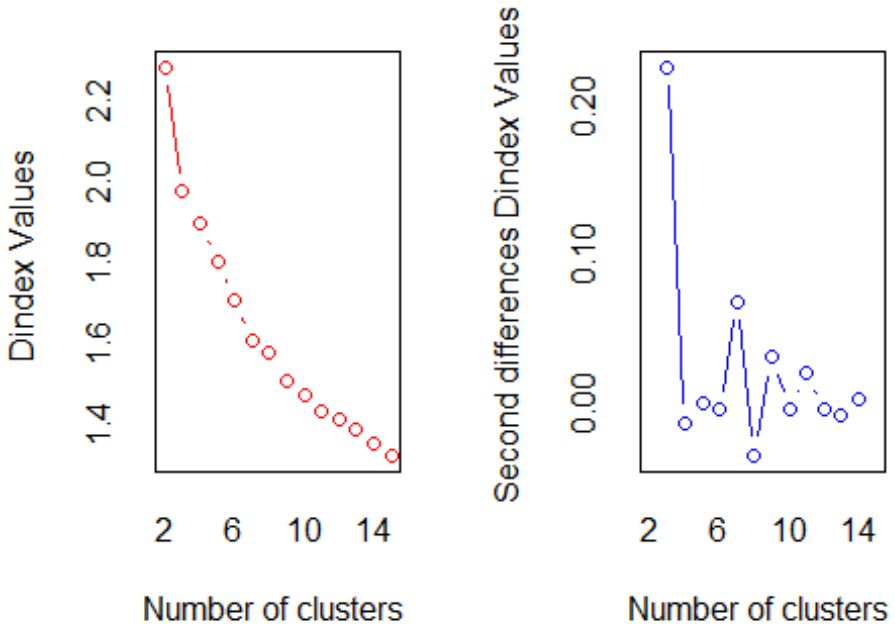
Partitional Method

K-Means

```
library(ggplot2)
nb <- NbClust(liver_scale, min.nc=2, max.nc=15, method="kmeans")
```



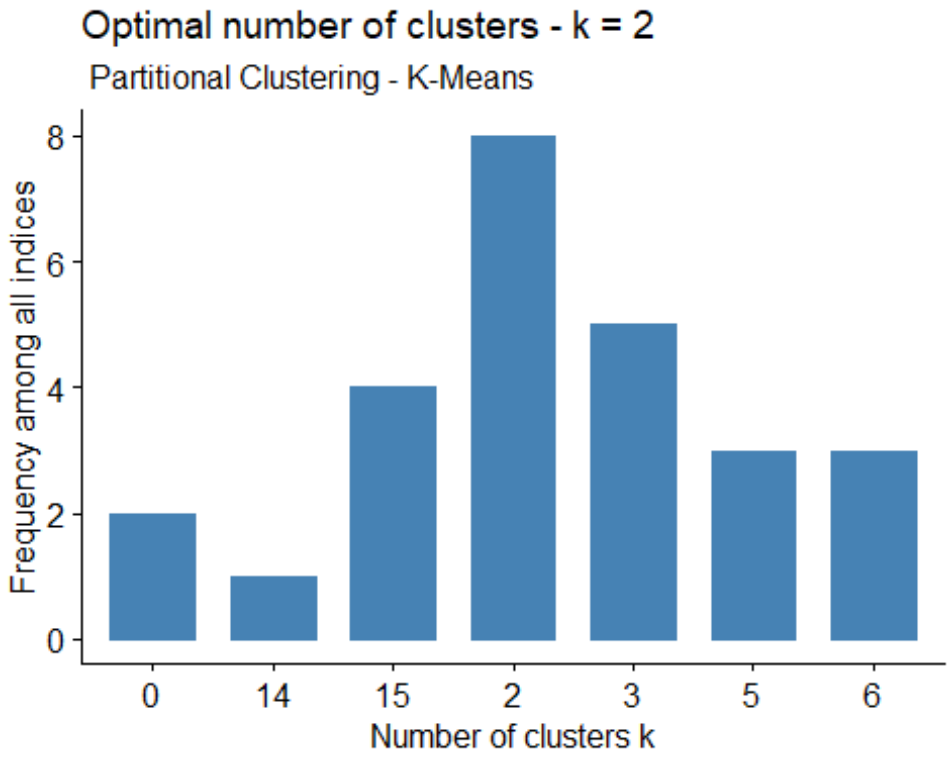

```
## *** : The Hubert index is a graphical method of determining the number of
clusters.
##                 In the plot of Hubert index, we seek a significant knee
that corresponds to a
##                 significant increase of the value of the measure i.e the
significant peak in Hubert
##       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



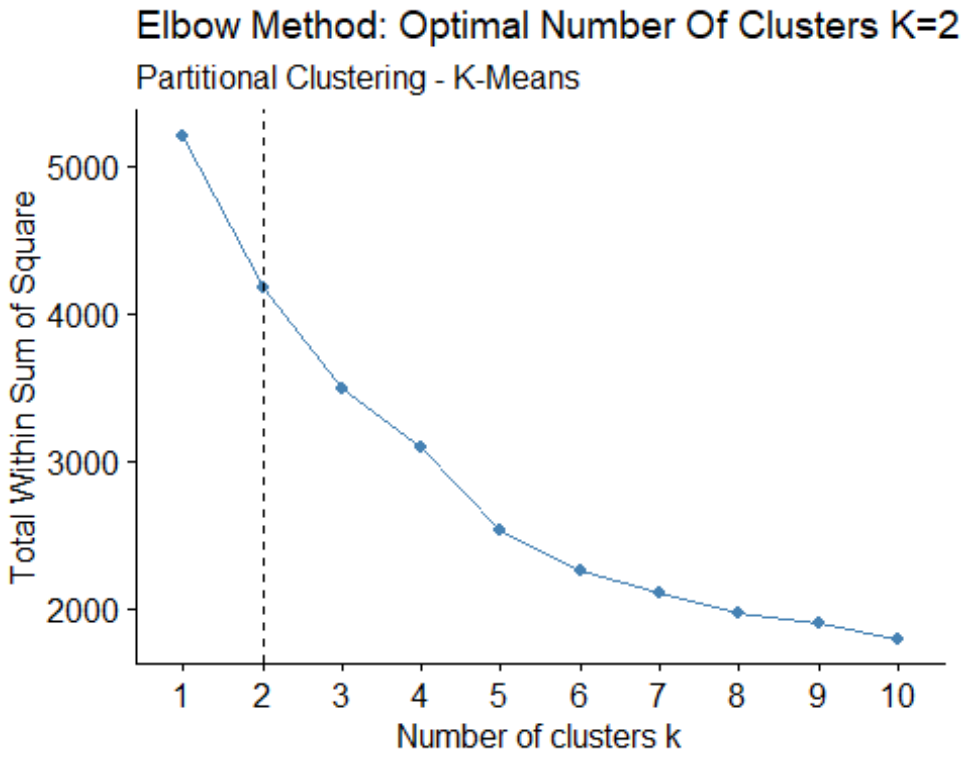

```
## *** : The D index is a graphical method of determining the number of
clusters.
##                 In the plot of D index, we seek a significant knee (the
significant peak in Dindex
##                 second differences plot) that corresponds to a significant
increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 8 proposed 2 as the best number of clusters
## * 5 proposed 3 as the best number of clusters
## * 3 proposed 5 as the best number of clusters
## * 3 proposed 6 as the best number of clusters
## * 1 proposed 14 as the best number of clusters
## * 4 proposed 15 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************
```

```
fviz_nbclust(nb) +
  labs(subtitle = " Partitional Clustering - K-Means")
```

```
## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : the condition has length > 1 and only the first
element
## will be used

## Warning in if (class(best_nc) == "matrix") .viz_NbClust(x, print.summary,
: the
## condition has length > 1 and only the first element will be used

## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : the condition has length > 1 and only the first
element
## will be used

## Warning in if (class(best_nc) == "matrix") {: the condition has length > 1
and
## only the first element will be used

## Among all indices:
## ===================
## * 2 proposed  0 as the best number of clusters
## * 8 proposed  2 as the best number of clusters
## * 5 proposed  3 as the best number of clusters
## * 3 proposed  5 as the best number of clusters
## * 3 proposed  6 as the best number of clusters
## * 1 proposed  14 as the best number of clusters
## * 4 proposed  15 as the best number of clusters
##
## Conclusion
## =========================
## * According to the majority rule, the best number of clusters is  2 .
```




```
fviz_nbclust(liver_scale, kmeans, method = "wss") +
  geom_vline(xintercept = 2, linetype = 2) +
  labs(title= "Elbow Method: Optimal Number Of Clusters K=2",
       subtitle = "Partitional Clustering - K-Means")
```




```
set.seed(123)
(km.res<- kmeans(liver_scale, 2, nstart = 25))

## K-means clustering with 2 clusters of sizes 517, 62
##
## Cluster means:
##            Age Total_Bilirubin Direct_Bilirubin Alkaline_Phosphotase
## 1 -0.002204957      -0.2651922       -0.2757749          -0.08439623
## 2  0.018386492       2.2113604        2.2996070           0.70375568
##   Alamine_Aminotransferase Aspartate_Aminotransferase Total_Protiens
Albumin
## 1               -0.1661591                 -0.1540735     0.01545215
0.0715517
## 2                1.3855523                  1.2847745    -0.12885100 -
0.5966488
##   Albumin_and_Globulin_Ratio
## 1                  0.0632937
## 2                 -0.5277878
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
19  20
##   1   2   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
39  40
##   1   1   1   1   1   2   2   2   1   1   1   1   1   1   1   1   1   2
```

```
1   1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58
59  60
##   1   1   1   1   1   1   1   2   1   2   1   1   1   1   1   1   1   1
1   1
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78
79  80
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   1
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98
99 100
##   1   1   1   1   1   1   1   1   1   1   2   2   2   1   1   1   1   2
1   1
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118
119 120
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   2   1   2
2   2
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138
139 140
##   2   2   1   1   1   1   1   1   1   1   1   1   1   1   1   2   1   1
1   1
## 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158
159 160
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   1
## 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178
179 180
##   2   2   1   1   1   1   2   1   2   1   1   1   2   1   1   1   1   2
2   1
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
199 200
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   2
## 201 202 203 204 205 206 207 208 209 211 212 213 214 215 216 217 218 219
220 221
##   1   1   1   1   1   1   1   2   1   1   1   1   1   1   1   1   1   1
1   1
## 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239
240 241
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   1
## 243 244 245 246 247 248 249 250 251 252 253 255 256 257 258 259 260 261
262 263
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   2   2
1   1
## 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281
282 283
##   1   1   1   1   1   2   1   1   1   2   1   1   1   1   1   1   1   1
1   1
## 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301
```

```
302 303
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   1
## 304 305 306 307 308 309 310 311 312 314 315 316 317 318 319 320 321 322
323 324
##   1   1   1   1   1   1   1   1   2   1   1   1   1   1   1   1   1   1
1   1
## 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
343 344
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   1
## 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362
363 364
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   1
## 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382
383 384
##   1   1   1   1   1   1   2   1   1   1   1   1   1   1   1   1   1   1
1   1
## 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402
403 404
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   1
## 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422
423 424
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   2   2   1
1   1
## 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442
443 444
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   1
## 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462
463 464
##   1   1   1   2   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   1
## 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482
483 484
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   2   2   1   1
1   1
## 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502
503 504
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   2
## 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522
523 524
##   2   2   2   1   2   2   1   1   1   1   1   1   1   1   1   2   1   1
1   1
## 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542
543 544
##   1   1   1   1   1   1   1   2   1   2   1   2   2   1   1   1   1   1
```

```
1   1
## 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562
563 564
##   1   1   2   2   1   1   1   1   1   1   1   1   1   1   1   2   2   2
2   1
## 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582
583
##   1   2   1   1   2   1   1   1   2   1   2   2   2   2   1   1   1   1
1
##
## Within cluster sum of squares by cluster:
## [1] 2610.734 1556.996
##  (between_SS / total_SS =  19.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
aggregate(liver_sub, by=list(cluster=km.res$cluster), mean)
```

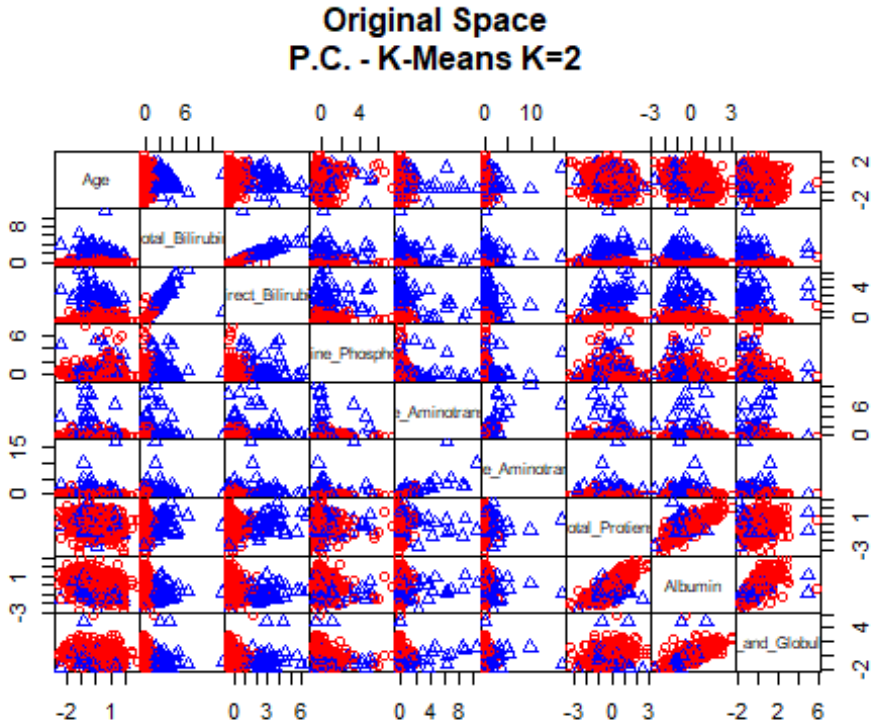
```
##   cluster      Age Total_Bilirubin Direct_Bilirubin Alkaline_Phosphotase
## 1       1 44.74662         1.66383        0.7174081             270.8104
## 2       2 45.08065        17.08710        7.9709677             462.7742
##   Alamine_Aminotransferase Aspartate_Aminotransferase Total_Protiens
Albumin
## 1                 50.68859                   65.75629       6.498453
3.195358
## 2                334.93548                  482.80645       6.341935
2.664516
##   Albumin_and_Globulin_Ratio
## 1                  0.9672921
## 2                  0.7783871
```

As the results shows first cluster contains lower units of variables.

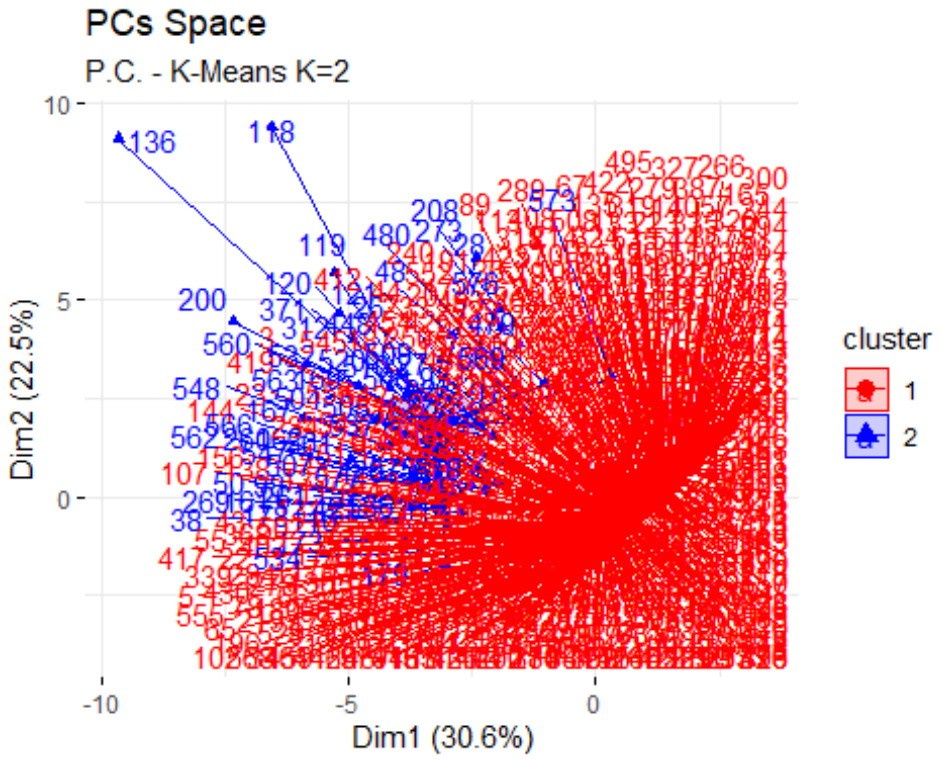
```
dd <- cbind(liver_sub, cluster = km.res$cluster)
head(dd)
```

```
##   Age Total_Bilirubin Direct_Bilirubin Alkaline_Phosphotase
## 1  65             0.7              0.1                  187
## 2  62            10.9              5.5                  699
## 3  62             7.3              4.1                  490
## 4  58             1.0              0.4                  182
## 5  72             3.9              2.0                  195
## 6  46             1.8              0.7                  208
##   Alamine_Aminotransferase Aspartate_Aminotransferase Total_Protiens
Albumin
## 1                       16                         18            6.8
3.3
```

```
## 2                          64                         100             7.5
3.2
## 3                          60                          68             7.0
3.3
## 4                          14                          20             6.8
3.4
## 5                          27                          59             7.3
2.4
## 6                          19                          14             7.6
4.4
##   Albumin_and_Globulin_Ratio cluster
## 1                       0.90       1
## 2                       0.74       2
## 3                       0.89       1
## 4                       1.00       1
## 5                       0.40       1
## 6                       1.30       1

cl <- km.res$cluster
table(cl)

## cl
##   1   2
## 517  62

pairs(liver_scale, gap=0, pch=cl, main="Original Space\nP.C. - K-Means K=2",
      cex.main= 1, col=c("red", "blue")[cl])
```



```
fviz_cluster(km.res, data = liver_scale, palette = c("red", "blue"),
             ellipse.type = "euclid", star.plot = TRUE, repel = TRUE,
             main= "PCs Space", ggtheme = theme_minimal()) +
  labs(subtitle = "P.C. - K-Means K=2")
```

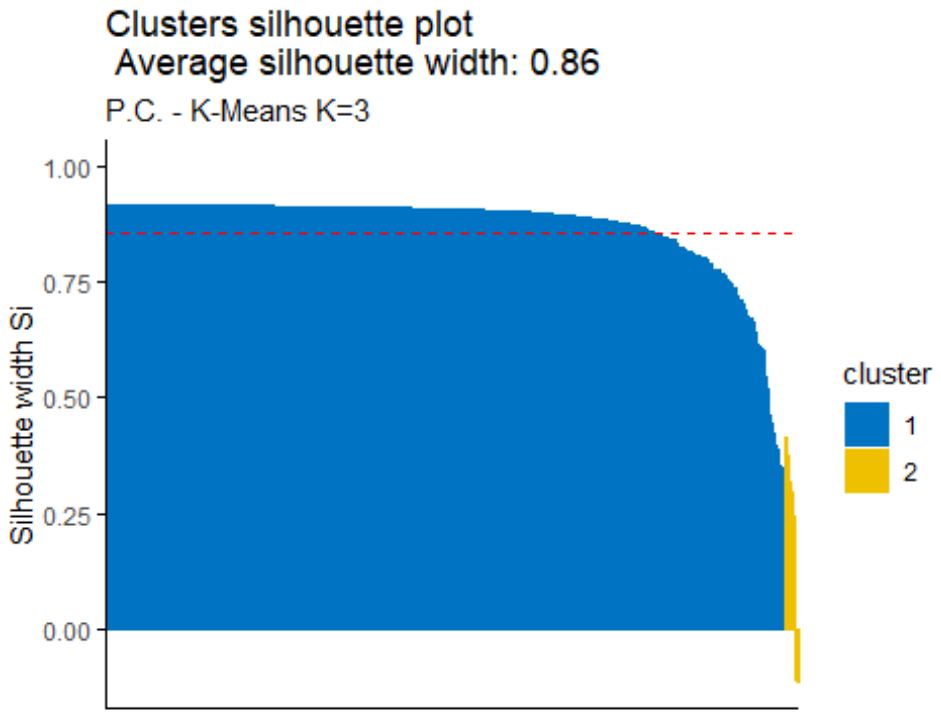


According to the function `NbClust`, the best number of clusters, applying partitional clustering using K-means method is 2. K-means clustering suggests 2 clusters of sizes 517 in cluster 1, and 62 in cluster 2. The quality between different clusters is 19.9 % of the total variability is explained by the separation between clusters. In the PCs space, there is no separation between clusters. To evaluate the goodness of clustering algorithm results, internal and external validation measures will be analyzed as follows:

Internal Validation Measures

Silhouette width

```
hclust<- eclust(liver_sub, k=2, "kmeans", nstart=25, graph = FALSE)
silinfo <- hclust$silinfo
silinfo$avg.width

## [1] 0.8566261

fviz_silhouette(hclust, palette = "jco", ggtheme = theme_classic()) +
  labs(subtitle = "P.C. - K-Means K=3")

##   cluster size ave.sil.width
## 1       1  568          0.87
## 2       2   11          0.18
```




```
silinfo$clus.avg.widths

## [1] 0.8697741 0.1777092

sil <- hclust$silinfo$widths[, 1:3]
neg_sil_index_aver.eu <- which(sil[, "sil_width"] < 0)
sil[neg_sil_index_aver.eu, , drop = FALSE]

##     cluster neighbor    sil_width
## 200       2        1 -0.003525611
## 26        2        1 -0.108790427
## 27        2        1 -0.108790427
## 480       2        1 -0.115812579
```

The value of average silhouette width indicates that in average the units are well enough clustered. In particular, in cluster 1 (blue cluster) the units are on average the same silhouette value with respect to the silhouette width, in cluster 2 (the yellow one) the units are below average the silhouette value with respect to silhouette width. According to the above index, 4 units that belong to cluster 2 are not well clustered, they should belong to the cluster 1.

Dunn index

```
stats <- cluster.stats(dist(liver_scale), hclust$cluster)
stats$dunn

## [1] 0.1439413
```

According to the Dunn index, the units are not clustered well enough.

External Validation Measures

Confusion matrix

```
table(liver$Liver_Disease, hclust$cluster)
```

```
##
##       1   2
##   0 403  11
##   1 165   0
```

According to the Confusion matrix, there is not a perfect agreement between the nominal variable Liver_Disease and the cluster solution. For the liver disease, data has been classified mostly in cluster 1, 568 units in cluster 1, and 11 in cluster 2. For the patients having liver disease classified mostly in cluster 1 while cluster 2 has 0 value. Safe to say data are not well balanced in both cluster.

Correct Rand Index

```
liver.disease <- as.numeric(liver$Liver_Disease)
stats<- cluster.stats(d = dist(liver_scale), liver.disease, hclust$cluster)
stats$corrected.rand
```

```
## [1] -0.02118264
```

According to the Correct Rand Index, there is no agreement between the numerical values and the cluster solution. From -1 to +1, the agreement is very close to 0.

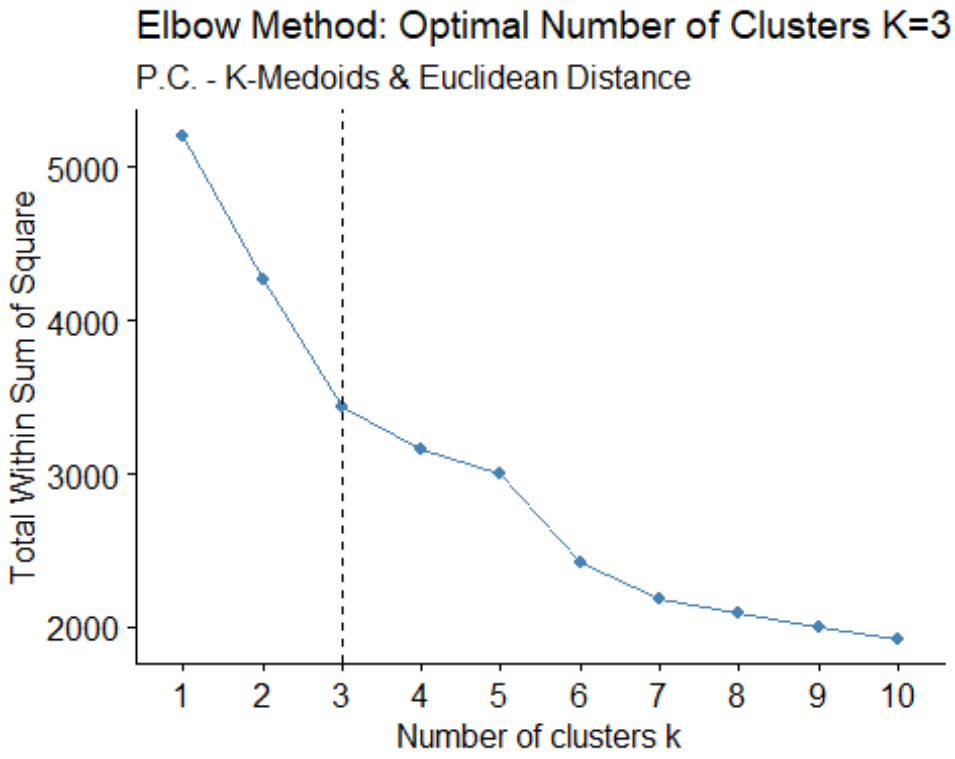
Meila's VI Index

```
stats$vi
```

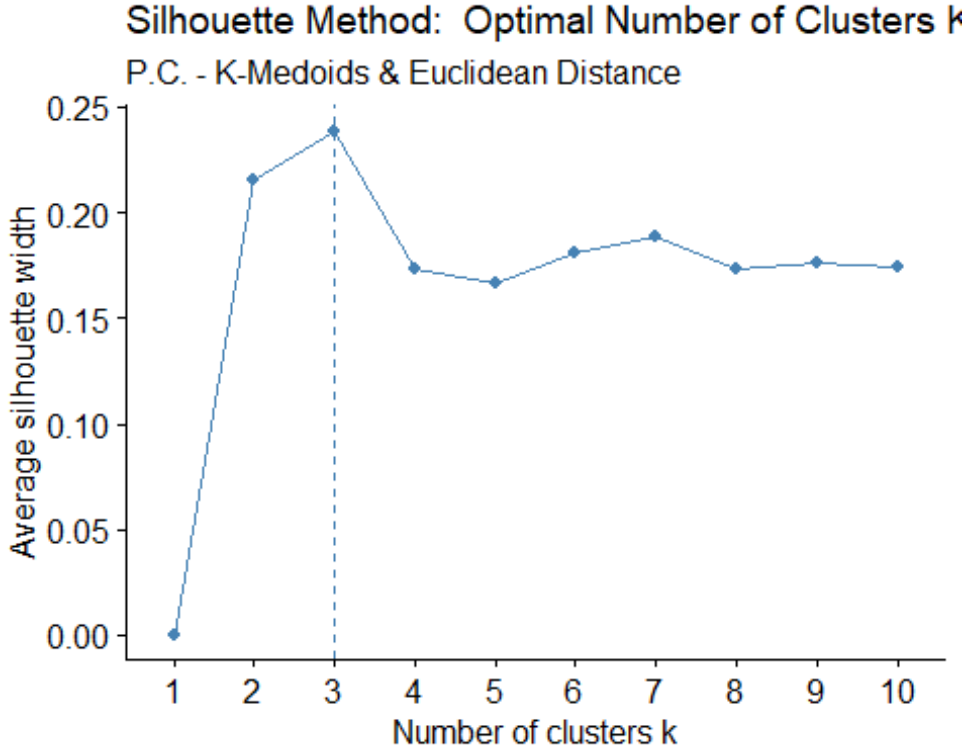
```
## [1] 0.6788131
```

Partitioning Around Medoids (PAM) & Euclidean Distance Method

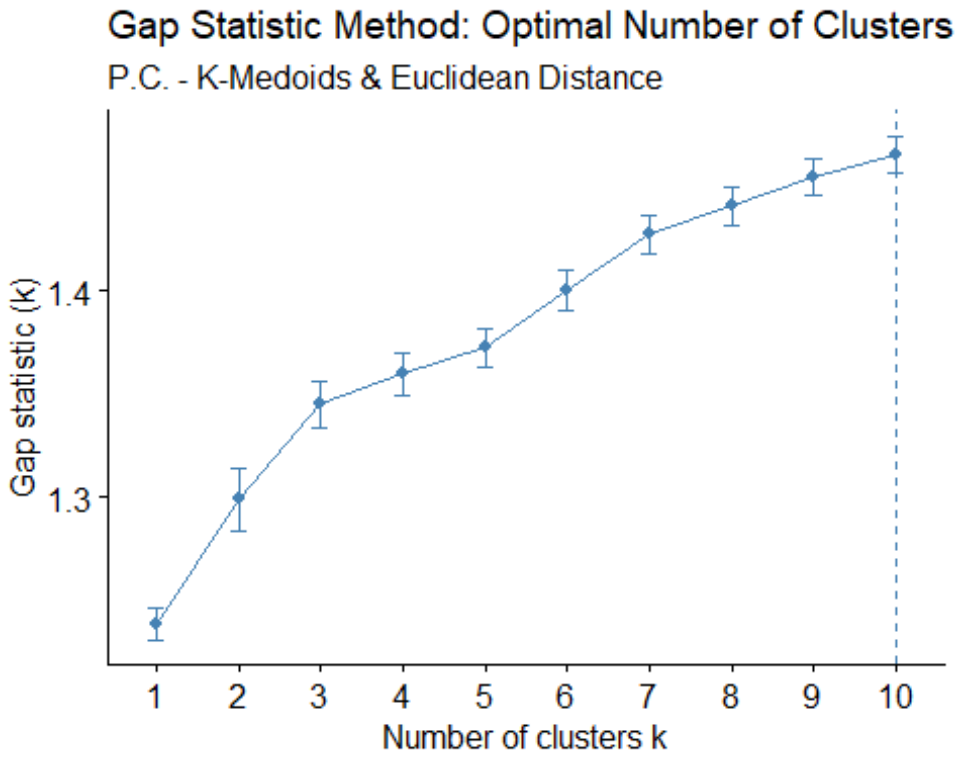
```
fviz_nbclust(liver_scale, cluster::pam, method = "wss") +
  geom_vline(xintercept = 3, linetype = 2) +
  labs(title = "Elbow Method: Optimal Number of Clusters K=3",
       subtitle="P.C. - K-Medoids & Euclidean Distance", cex.sub= 0.5)
```




```
fviz_nbclust(liver_scale, cluster::pam, method = "silhouette") +
  labs(title = "Silhouette Method:  Optimal Number of Clusters K=3",
       subtitle="P.C. - K-Medoids & Euclidean Distance", cex.sub= 0.5)
```



```r
fviz_nbclust(liver_scale, cluster::pam, method = "gap_stat", nboot = 500) +
  labs(title = "Gap Statistic Method: Optimal Number of Clusters K",
       subtitle="P.C. - K-Medoids & Euclidean Distance", cex.sub= 0.5)
```



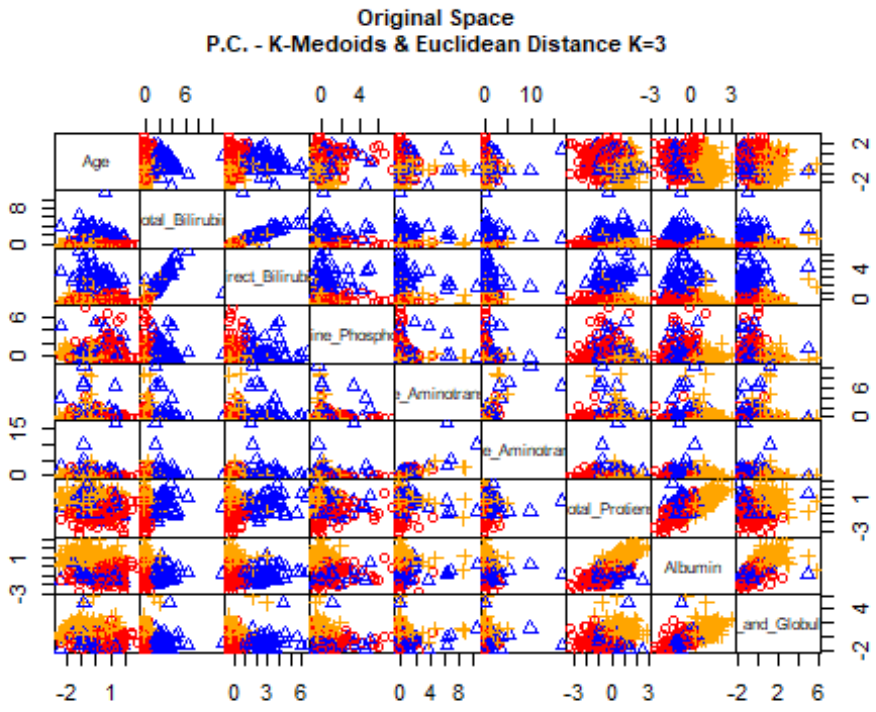

```r
library(cluster)
set.seed(123)
(pam.res <- pam(liver_scale, 3, metric = "euclidean"))
```

```
## Medoids:
##      ID         Age Total_Bilirubin Direct_Bilirubin Alkaline_Phosphotase
## 246 244  0.56822453      -0.3878422       -0.4594811         -0.005609041
## 537 533  0.07506058       2.0046880        2.0258743         -0.264270225
## 323 319 -0.54139437      -0.2593842       -0.3529658         -0.354596353
##     Alamine_Aminotransferase Aspartate_Aminotransferase Total_Protiens
## 246              -0.36098402                 -0.3188356     -0.3519068
## 537              -0.07711464                  0.3780765      0.3856644
## 323              -0.24634446                 -0.2636346      0.5700572
##        Albumin Albumin_and_Globulin_Ratio
## 246 -0.4261076                -0.4601612
## 537 -0.6778590                -1.0859589
## 323  0.9585247                 0.7914341
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
19  20
##   1   2   1   1   1   3   3   3   3   1   1   1   1   3   1   1   3   3
1   1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
```

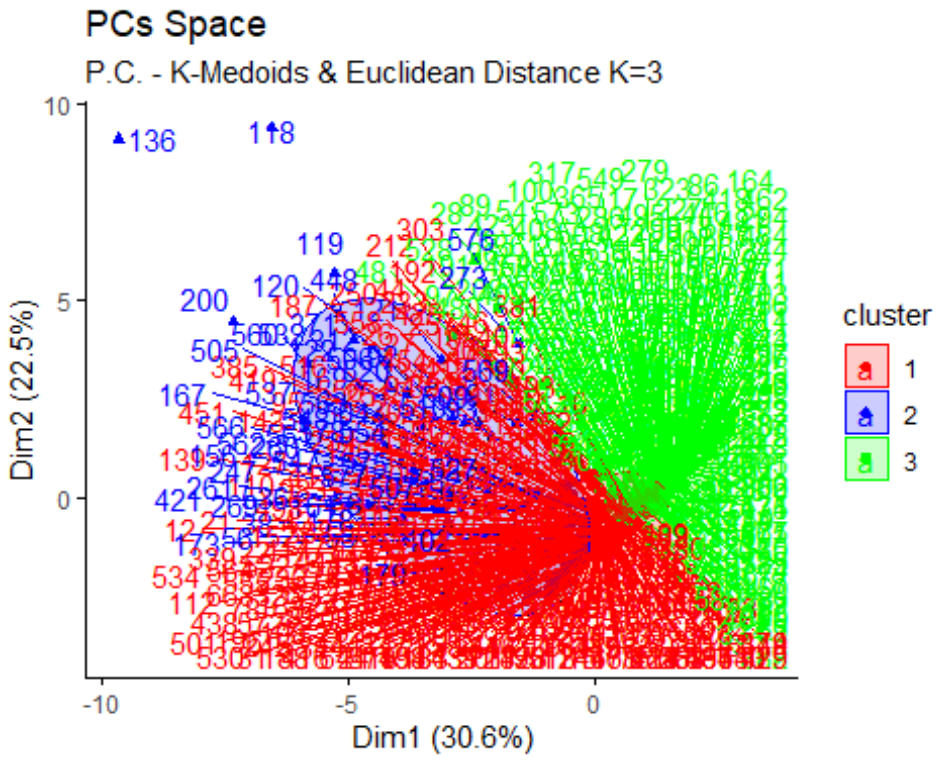
```
39  40
##   1   1   1   1   3   1   1   3   1   1   1   1   1   1   1   3   3   2
1   1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58
59  60
##   1   1   1   1   1   1   3   3   1   2   1   3   3   3   2   2   3   1
1   1
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78
79  80
##   1   3   3   1   1   1   3   3   3   1   3   1   1   3   1   1   1   1
3   1
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98
99 100
##   1   3   1   3   3   3   3   1   3   1   2   2   2   1   1   1   3   1
3   3
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118
119 120
##   1   1   3   1   1   1   1   1   1   1   1   1   3   1   1   1   1   2
2   2
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138
139 140
##   2   2   3   1   3   1   1   1   1   1   1   1   3   1   3   2   1   1
1   3
## 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158
159 160
##   1   1   1   1   1   1   1   1   3   3   3   1   1   3   1   2   1   1
1   3
## 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178
179 180
##   2   2   1   3   3   1   2   1   2   1   1   3   2   3   3   3   1   2
2   1
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
199 200
##   1   1   1   1   3   1   1   1   1   1   3   1   1   1   1   1   1   1
1   2
## 201 202 203 204 205 206 207 208 209 211 212 213 214 215 216 217 218 219
220 221
##   1   1   1   3   3   3   1   3   1   1   1   3   3   3   3   1   3   1
3   1
## 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239
240 241
##   3   3   3   1   3   3   3   1   1   3   3   1   1   3   1   3   3   3
1   3
## 243 244 245 246 247 248 249 250 251 252 253 255 256 257 258 259 260 261
262 263
##   3   3   1   1   2   1   1   1   1   1   3   3   1   1   1   3   2   2
3   3
## 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281
282 283
##   1   3   3   3   3   2   1   3   3   2   3   3   3   3   1   3   3   1
```

```
3   1
## 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301
302 303
##   1   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
1   1
## 304 305 306 307 308 309 310 311 312 314 315 316 317 318 319 320 321 322
323 324
##   3   3   3   1   1   3   3   3   2   3   3   3   3   3   3   3   3   1
3   3
## 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
343 344
##   1   1   3   3   1   3   1   1   1   3   3   3   1   1   1   1   1   1
3   1
## 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362
363 364
##   3   1   3   3   1   3   1   1   3   3   3   3   3   1   1   3   3   1
1   3
## 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382
383 384
##   3   1   3   1   3   3   2   3   3   3   3   3   3   3   1   1   1   3
1   1
## 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402
403 404
##   1   3   3   3   3   1   1   1   1   3   1   1   1   1   3   3   1   2
3   3
## 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422
423 424
##   3   3   3   3   3   1   1   1   1   1   1   1   1   3   1   1   2   3
3   3
## 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442
443 444
##   1   1   1   1   1   1   1   1   3   3   3   3   3   1   1   1   3   1
1   1
## 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462
463 464
##   1   3   1   2   1   1   1   3   1   1   3   3   3   1   1   1   1   3
1   3
## 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482
483 484
##   1   1   1   3   1   1   3   1   1   3   1   3   1   1   3   3   3   3
3   3
## 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502
503 504
##   1   1   1   1   1   1   3   3   1   1   3   1   3   1   3   1   1   1
1   2
## 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522
523 524
##   2   2   2   3   2   2   3   3   1   1   3   1   1   3   3   2   3   1
3   1
## 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542
```

```
543 544
##   3   3   2   3   1   1   3   2   1   1   3   2   2   3   1   1   1   1
3   3
## 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562
563 564
##   1   1   2   2   3   3   1   1   1   2   1   1   1   1   3   2   2   2
2   1
## 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582
583
##   1   2   3   1   2   3   1   1   3   1   2   2   2   2   1   3   1   3
3
## Objective function:
##    build     swap
## 2.066652 2.005459
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"

pam.res$clusinfo

##      size  max_diss  av_diss  diameter separation
## [1,]  290  7.614993 1.889974  9.601375  0.2933832
## [2,]   57 17.925607 3.792078 21.013509  0.9263908
## [3,]  232  9.598995 1.710860 10.751924  0.2933832

cc <- pam.res$cluster
pairs(liver_scale, gap=0, pch=cc,
      main="Original Space\nP.C. - K-Medoids & Euclidean Distance K=3",
      cex.main= 0.7, col=c("red", "blue", "orange")[cc])
```




```
fviz_cluster(pam.res, palette = c("red", "blue", "green"), ellipse.type =
"t",
             repel = TRUE, main= "PCs Space", ggtheme = theme_classic()) +
  labs(subtitle = "P.C. - K-Medoids & Euclidean Distance K=3", cex.sub= 0.5)
```




```
table(cc)

## cc
##   1   2   3
## 290  57 232
```

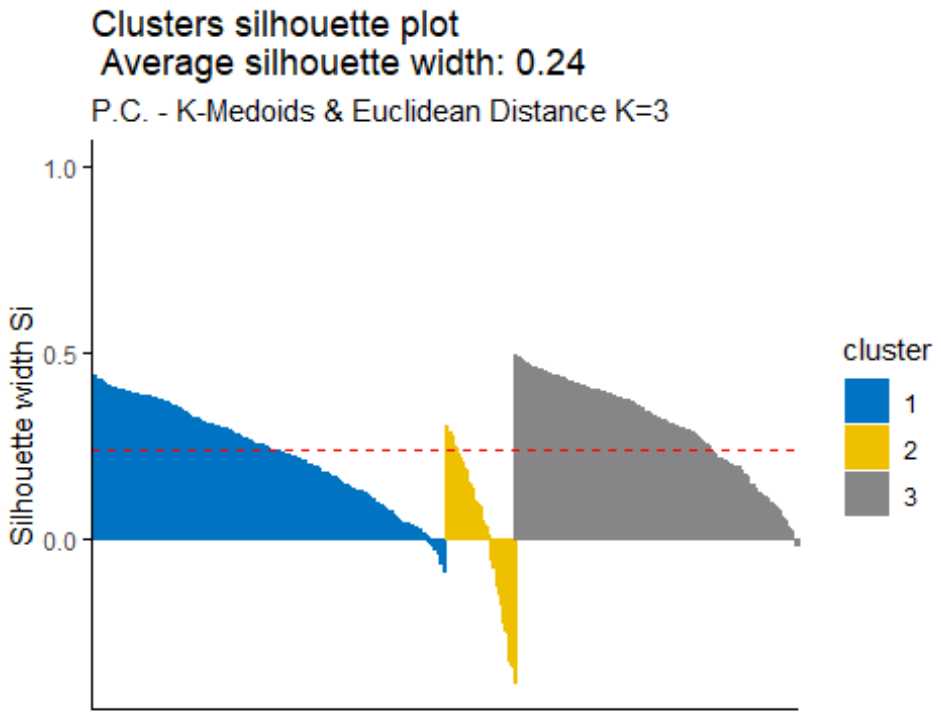
Applying partitioning clustering method and using PAM algorithm and euclidean distance, clusters are composed in this way: cluster 1 with 290 units, cluster 2 with 57 units and cluster 3 with 232 units. To evaluate the goodness of clustering algorithm results, internal and external validation measures will be analyzed as follows:

Internal Validation Measures

Silhouette width

```
hclust <- eclust(liver_sub, k = 3, "pam", graph = FALSE, hc_metric =
"euclidean")
silinfo <- hclust$silinfo
silinfo$avg.width

## [1] 0.6563575

fviz_silhouette(pam.res, palette = "jco", ggtheme = theme_classic()) +
  labs(subtitle = "P.C. - K-Medoids & Euclidean Distance K=3",
       cex.sub= 0.5)

##   cluster size ave.sil.width
## 1       1  290          0.23
```

```
## 2       2   57          0.03
## 3       3  232          0.30
```




```
silinfo$clus.avg.widths

## [1]  0.79842949  0.17747402 -0.06541546

sil <- hclust$silinfo$widths[, 1:3]
neg_sil_index_aver.eu <- which(sil[, "sil_width"] < 0)
sil[neg_sil_index_aver.eu, , drop = FALSE]

##     cluster neighbor   sil_width
## 385       2        1 -0.03525340
## 579       2        1 -0.04082789
## 250       2        1 -0.04883615
## 3         2        1 -0.05143776
## 177       2        1 -0.07069781
## 36        2        1 -0.07642448
## 110       2        1 -0.09948974
## 22        2        1 -0.11948583
## 251       2        1 -0.12733909
## 272       2        1 -0.12901159
## 66        2        1 -0.14092109
## 337       2        1 -0.14469730
## 509       2        1 -0.17036132
## 536       2        1 -0.17576401
## 157       2        1 -0.17670219
```

```
## 419       2        1 -0.24048823
## 548       2        1 -0.29425858
## 303       2        1 -0.34485285
## 367       2        1 -0.39104500
## 34        2        1 -0.41151262
## 35        2        1 -0.41151262
## 339       2        1 -0.42249684
## 200       3        2 -0.02551749
## 561       3        1 -0.06630383
## 273       3        1 -0.09078114
## 520       3        1 -0.18045397
## 97        3        1 -0.19330062
## 510       3        1 -0.28699184
## 100       3        1 -0.32409255
## 121       3        1 -0.33688590
## 559       3        1 -0.37265836
## 17        3        1 -0.38860134
## 562       3        1 -0.42656660
## 94        3        1 -0.44157478
## 495       3        1 -0.46335807
## 563       3        1 -0.47797999
## 71        3        1 -0.49335725
## 53        3        1 -0.52732712
## 236       3        1 -0.53899193
## 482       3        1 -0.54486317
```

The value of average silhouette width indicates that in average the units are not well enough clustered. In particular, in cluster 1 the units are in average having the same silhouette value with respect to the silhouette width, in cluster 2 the units have in average a lowest value with respect to the silhouette width while in cluster 3 the units are in average having the higher value with respect to the silhouette width. According to the index, 40 units are not well clustered, units that belong to cluster 2 and 3, should belong to cluster 1.

Dunn Index

```
stats <- cluster.stats(dist(liver_scale), hclust$cluster)
stats$dunn
```

```
## [1] 0.02673178
```

According to the Dunn index, the units are not clustered well enough.

External validation Measures

Confusion Matrix

```
table(liver$Liver_Disease, hclust$cluster)
```

```
##
##       1   2   3
##   0 307  70  37
##   1 154  11   0
```

According to the Confusion matrix, there is not a perfect agreement between the nominal variable Liver_Disease and the cluster solution. A large number of patients not having liver disease 461 has been classified in cluster 1, 81 in cluster 2, and 37 in cluster 3. For the patients who have liver disease mostly classified in cluster 1 with 154 units and 11 units in cluster 2 but none in cluster 3.

Correct Rand Index

```
liver.disease <- as.numeric(liver$Liver_Disease)
stats<- cluster.stats(d = dist(liver_scale), liver.disease, hclust$cluster)
stats$corrected.rand

## [1] -0.08037447
```

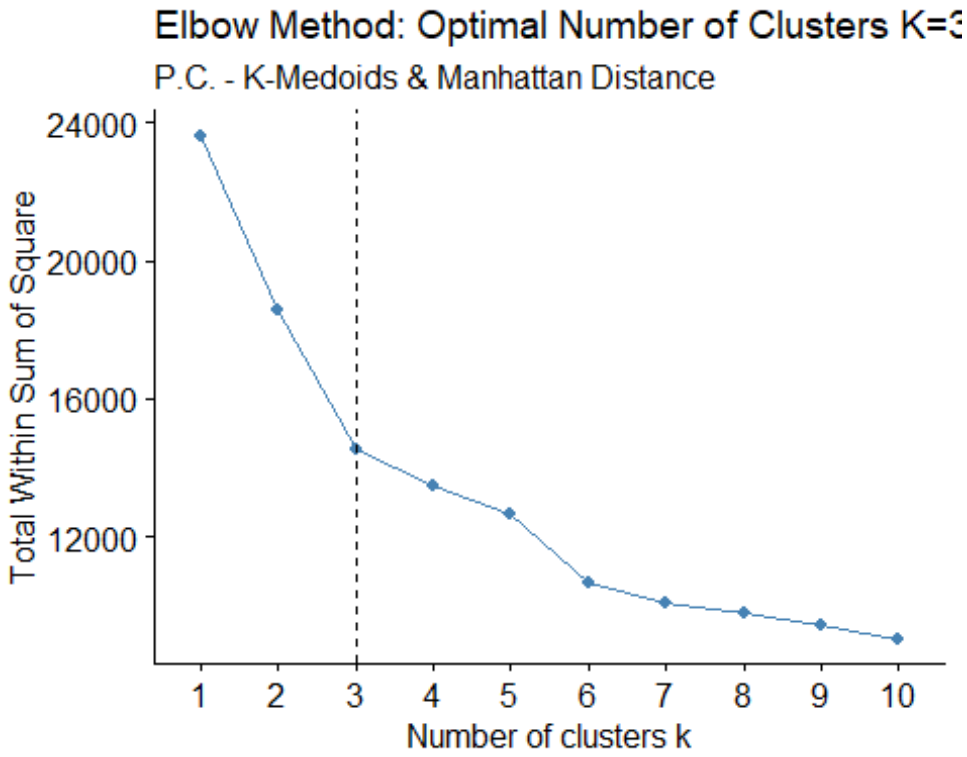
According to the Correct Rand Index, there is no agreement between the numerical values and the cluster solution. From -1 to +1, the agreement is very close to 0.

Meila's VI Index

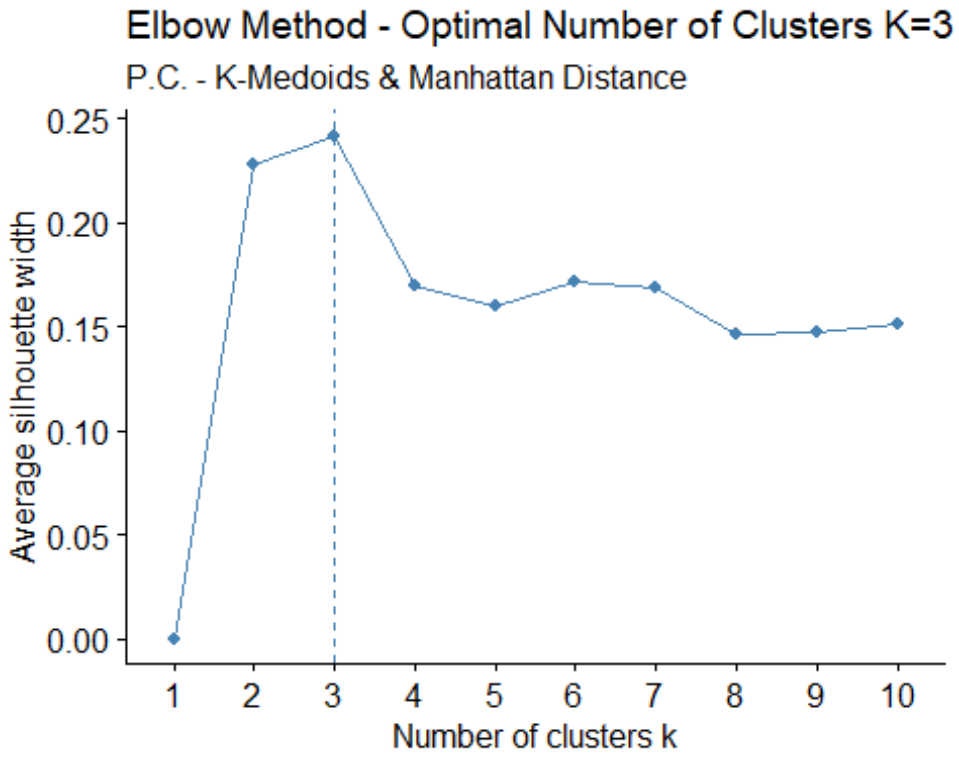
```
stats$vi

## [1] 1.160317
```

Partitioning Around Medoids (PAM) & Manhattan Distance

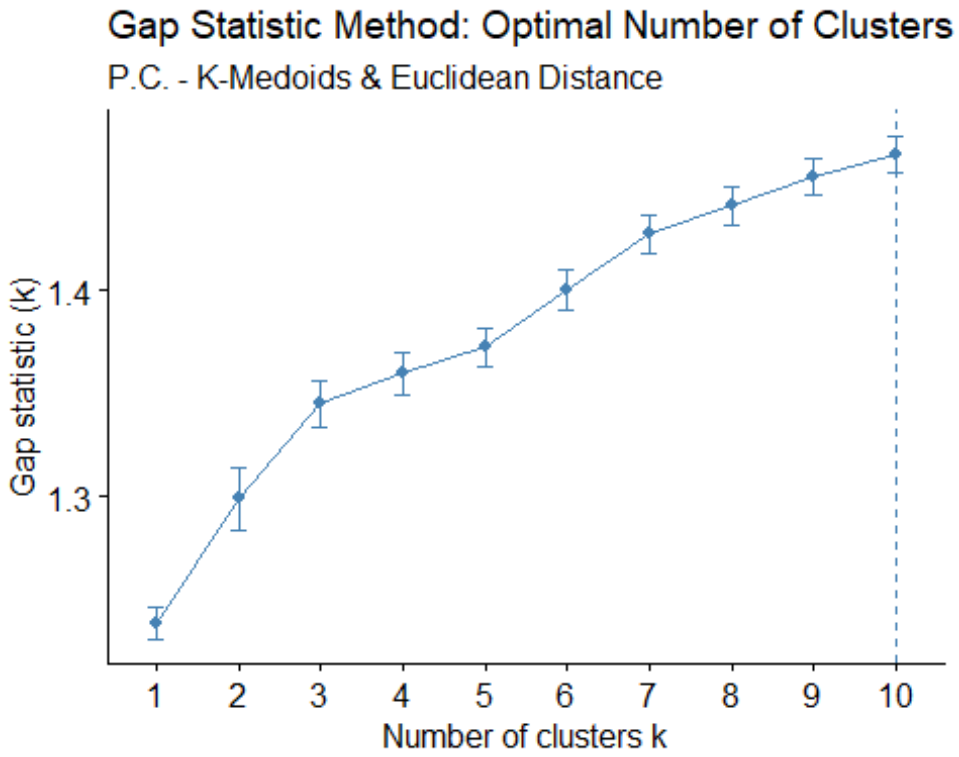
```
set.seed(123)
fviz_nbclust(liver_scale, cluster::pam, method = "wss",
             diss = dist(liver_scale, method = "manhattan")) +
  geom_vline(xintercept = 3, linetype = 2) +
  labs(title = "Elbow Method: Optimal Number of Clusters K=3",
       subtitle="P.C. - K-Medoids & Manhattan Distance", cex.sub= 0.5)
```




```
fviz_nbclust(liver_scale, cluster::pam, method = "silhouette",
             diss = dist(liver_scale, method = "manhattan")) +
  labs(title = "Elbow Method - Optimal Number of Clusters K=3",
       subtitle="P.C. - K-Medoids & Manhattan Distance", cex.sub= 0.5)
```




```
set.seed(123)
fviz_nbclust(liver_scale, cluster::pam, method = "gap_stat", nboot = 500,
             diss=dist(liver_scale, method = "manhattan")) +
  labs(title = "Gap Statistic Method: Optimal Number of Clusters K",
       subtitle="P.C. - K-Medoids & Euclidean Distance", cex.sub= 0.5)
```



In order to find the optimal number of clusters, three indices were used. The elbow method seem suggest k=3, Silhouette method suggests 3 clusters, Gap statistics 10 clusters. It was decided to proceed by identifying 3 clusters.

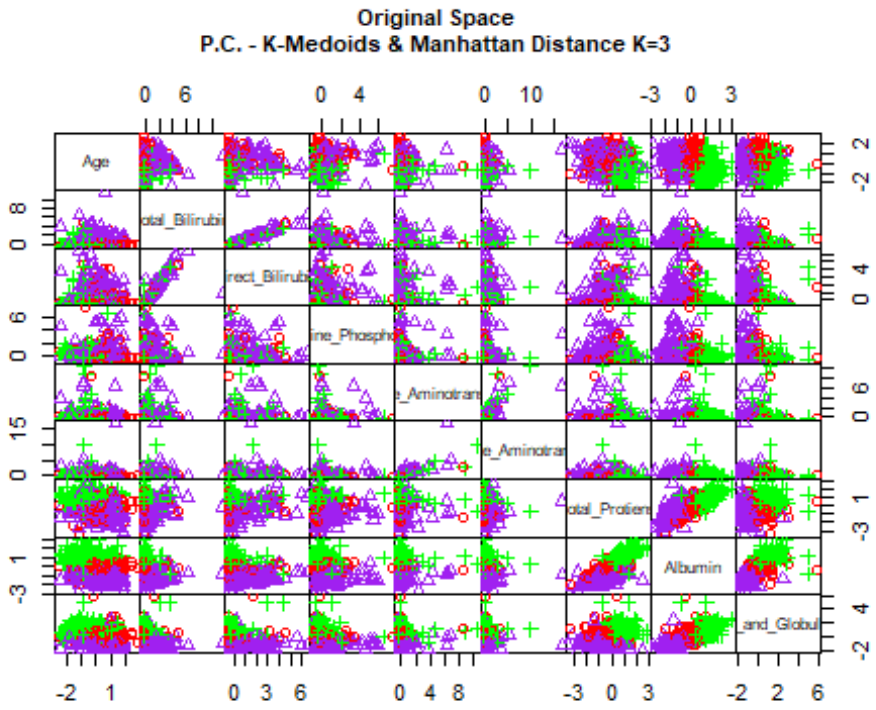
```
library(cluster)
set.seed(123)
(pam.res <- pam(liver_scale, 3, metric="manhattan"))
```

```
## Medoids:
##      ID        Age Total_Bilirubin Direct_Bilirubin Alkaline_Phosphotase
## 450 446  0.1983516     -0.38784221       -0.4594811           -0.4859798
## 112 112  0.1983516     -0.01852546        0.0375900           -0.1410982
## 176 176 -0.8496218     -0.40389946       -0.4594811           -0.3833365
##     Alamine_Aminotransferase Aspartate_Aminotransferase Total_Protiens
## 450               -0.3009347                -0.28778505     -0.2597104
## 112               -0.2627215                 0.01927028     -0.7206924
## 176               -0.2081313                -0.27398481      0.7544500
##         Albumin Albumin_and_Globulin_Ratio
## 450 -0.04848061                 0.1656364
## 112 -1.18136162                -1.0233792
## 176  1.08440040                 0.7914341
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
19  20
##   1   2   1   1   2   3   3   3   3   1   1   2   1   3   1   2   3   3
1   1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
```

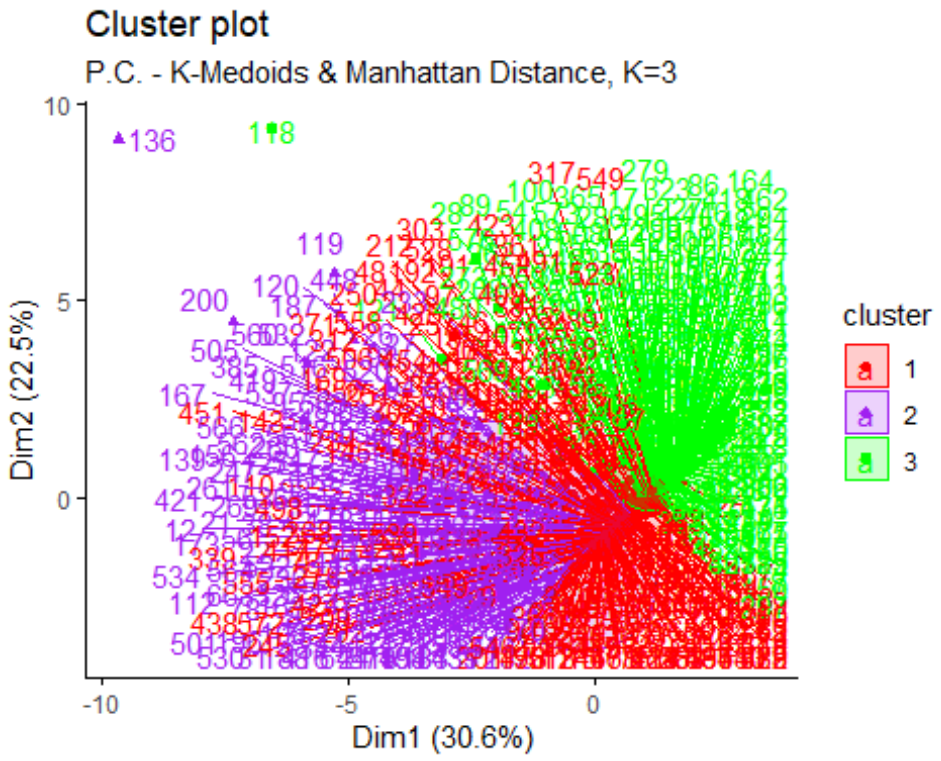
```
39  40
##   2   2   1   2   1   2   2   3   1   1   2   1   1   2   2   3   3   2
2   1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58
59  60
##   1   1   2   1   1   1   3   1   1   2   1   1   3   3   2   2   3   1
2   1
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78
79  80
##   1   1   1   2   2   2   1   3   1   1   1   2   2   1   1   1   2   2
1   2
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98
99 100
##   2   1   1   1   3   3   3   1   3   1   1   2   2   2   2   1   1   2
1   3
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118
119 120
##   2   2   3   1   1   2   2   2   2   1   2   2   1   2   2   2   2   3
2   2
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138
139 140
##   3   2   1   1   3   2   2   1   3   2   2   2   3   1   3   2   2   2
2   3
## 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158
159 160
##   1   2   1   1   1   1   2   1   3   3   3   1   1   1   2   2   1   2
2   3
## 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178
179 180
##   2   2   2   3   3   2   2   2   1   1   1   1   2   3   3   3   2   2
2   2
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
199 200
##   2   2   2   1   3   1   2   2   2   1   1   1   1   1   1   2   2   1
1   2
## 201 202 203 204 205 206 207 208 209 211 212 213 214 215 216 217 218 219
220 221
##   1   1   1   3   1   3   2   3   1   1   1   1   3   3   1   1   3   2
1   1
## 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239
240 241
##   3   3   3   1   3   3   1   1   1   3   3   2   1   3   2   3   3   3
1   3
## 243 244 245 246 247 248 249 250 251 252 253 255 256 257 258 259 260 261
262 263
##   3   3   1   1   2   1   1   1   1   2   3   3   1   1   1   3   2   2
3   3
## 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281
282 283
##   1   3   3   1   3   2   1   3   1   3   3   3   3   3   1   3   3   1
```

```
3   1
## 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301
302 303
##   2   3   3   3   3   1   3   3   3   1   3   3   3   3   3   3   3   3
1   1
## 304 305 306 307 308 309 310 311 312 314 315 316 317 318 319 320 321 322
323 324
##   3   3   1   2   1   3   3   3   1   3   3   3   1   1   3   3   3   1
3   3
## 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
343 344
##   1   1   3   3   1   3   1   2   2   1   3   3   1   1   1   2   1   1
3   2
## 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362
363 364
##   3   2   1   3   1   1   1   2   3   3   3   3   3   1   1   3   1   1
1   3
## 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382
383 384
##   3   1   3   1   3   3   1   3   3   3   3   1   3   1   1   1   1   1
1   1
## 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402
403 404
##   2   3   3   1   3   1   1   1   1   3   2   1   1   1   3   1   1   2
3   1
## 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422
423 424
##   3   3   3   3   1   1   2   1   2   2   2   2   1   3   2   2   2   3
1   3
## 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442
443 444
##   1   1   1   1   1   2   2   1   3   3   3   3   1   1   2   1   3   2
2   1
## 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462
463 464
##   1   3   1   2   1   1   1   1   1   1   3   3   3   1   2   2   2   3
1   3
## 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482
483 484
##   2   2   2   1   2   2   3   1   2   3   2   3   1   1   3   3   3   3
3   3
## 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502
503 504
##   2   2   1   1   1   1   1   3   3   2   3   1   3   1   3   2   2   2
1   2
## 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522
523 524
##   2   1   2   3   2   1   3   1   1   1   3   2   1   3   1   2   3   2
1   1
## 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542
```

```
543 544
##   3   3   2   1   2   2   3   2   1   2   3   2   2   3   1   1   1   1
3   3
## 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562
563 564
##   1   1   2   2   1   3   2   1   1   2   1   1   1   1   3   2   2   2
2   2
## 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582
583
##   1   2   3   2   3   3   2   1   3   2   2   3   2   2   2   1   1   3
3
## Objective function:
##    build     swap
## 4.145986 4.145986
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"

pam.res$clusinfo

##      size max_diss  av_diss diameter separation
## [1,]  229 15.28487 3.307738 27.59425  0.5847884
## [2,]  164 32.40721 5.989488 39.37034  0.8141526
## [3,]  186 27.52860 3.552567 36.70780  0.5847884

cm <- pam.res$cluster
pairs(liver_scale, gap=0,
      main="Original Space\nP.C. - K-Medoids & Manhattan Distance K=3",
      cex.main= 0.7, pch = cm, col = c("red", "purple", "green")[cm])
```




```
fviz_cluster(pam.res, palette = c("red", "purple", "green"), ellipse.type =
"t",
             repel = TRUE, ggtheme = theme_classic()) +
  labs(subtitle = "P.C. - K-Medoids & Manhattan Distance, K=3", cex.sub= 0.5)
```




```
table(cm)

## cm
##   1   2   3
## 229 164 186
```

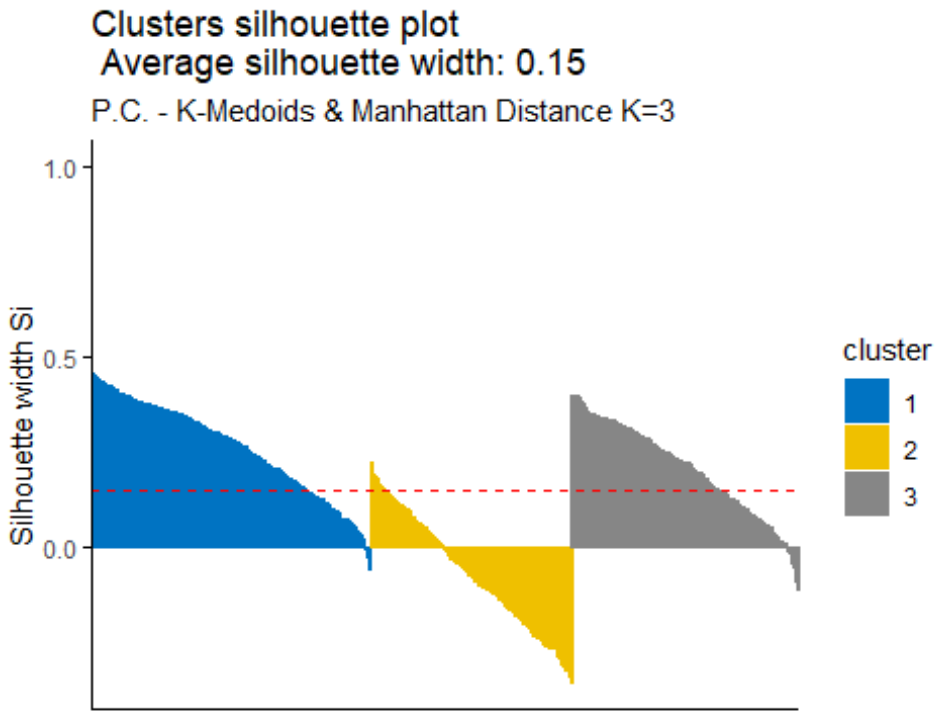
Applying partitioning clustering using PAM algorithm and euclidean distance, three clusters are composed in this way: cluster 1 with 290 units, cluster 2 with 57 units and cluster 3 with 232 units. To evaluate the goodness of clustering algorithm results, internal and external validation measures will be analyzed as follows:

Internal Validation Measures

Silhouette width

```
hclust<- eclust(liver_sub, k=3, "pam", graph = FALSE, hc_metric =
"manhattan")
silinfo <- hclust$silinfo
silinfo$avg.width

## [1] 0.6563575

fviz_silhouette(pam.res, palette = "jco", ggtheme = theme_classic()) +
  labs(subtitle = "P.C. - K-Medoids & Manhattan Distance K=3", cex.sub= 0.5)

##   cluster size ave.sil.width
## 1       1  229          0.26
```

```
## 2       2  164          -0.07
## 3       3  186           0.20
```




```
silinfo$clus.avg.widths

## [1]  0.79842949  0.17747402 -0.06541546

sil <- hclust$silinfo$widths[, 1:3]
neg_sil_index_aver.eu <- which(sil[, "sil_width"] < 0)
sil[neg_sil_index_aver.eu, , drop = FALSE]

##     cluster neighbor   sil_width
## 385       2        1 -0.03525340
## 579       2        1 -0.04082789
## 250       2        1 -0.04883615
## 3         2        1 -0.05143776
## 177       2        1 -0.07069781
## 36        2        1 -0.07642448
## 110       2        1 -0.09948974
## 22        2        1 -0.11948583
## 251       2        1 -0.12733909
## 272       2        1 -0.12901159
## 66        2        1 -0.14092109
## 337       2        1 -0.14469730
## 509       2        1 -0.17036132
## 536       2        1 -0.17576401
## 157       2        1 -0.17670219
```

```
## 419       2        1 -0.24048823
## 548       2        1 -0.29425858
## 303       2        1 -0.34485285
## 367       2        1 -0.39104500
## 34        2        1 -0.41151262
## 35        2        1 -0.41151262
## 339       2        1 -0.42249684
## 200       3        2 -0.02551749
## 561       3        1 -0.06630383
## 273       3        1 -0.09078114
## 520       3        1 -0.18045397
## 97        3        1 -0.19330062
## 510       3        1 -0.28699184
## 100       3        1 -0.32409255
## 121       3        1 -0.33688590
## 559       3        1 -0.37265836
## 17        3        1 -0.38860134
## 562       3        1 -0.42656660
## 94        3        1 -0.44157478
## 495       3        1 -0.46335807
## 563       3        1 -0.47797999
## 71        3        1 -0.49335725
## 53        3        1 -0.52732712
## 236       3        1 -0.53899193
## 482       3        1 -0.54486317
```

The value of average silhouette width indicates that in average the units are not well enough clustered. In particular, in cluster 1 the units are in average having the same silhouette value with respect to the silhouette width, in cluster 2 the units have in average a lowest value with respect to the silhouette width while in cluster 3 the units are in average having the higher value with respect to the silhouette width. According to the index, 40 units are not well clustered, units that belong to cluster 2 and 3, should belong to cluster 1.

Dunn index

```
stats <- cluster.stats(dist(liver_scale), hclust$cluster)
stats$dunn
```

```
## [1] 0.02673178
```

According to the Dunn index, the units are not clustered well enough.

External Validation Measures

Confusion Matrix

```
table(liver$Liver_Disease, hclust$cluster)
```

```
##
##       1   2   3
##   0 307  70  37
##   1 154  11   0
```

According to the Confusion matrix, there is not a perfect agreement between the nominal variable Liver_disease and the cluster solution. A large number of patients not having liver disease has been classified in cluster 1, 70 in cluster 2, and 37 have been classified in cluster 3. For patients having a liver disease, large number of data - 154 units has been classified in cluster 1, 11 in cluster 2, and 0 in cluster 3.

Correct Rand Index

```
liver.disease <- as.numeric(liver$Liver_Disease)
stats<- cluster.stats(d = dist(liver_scale), liver.disease, hclust$cluster)
stats$corrected.rand
```

```
## [1] -0.08037447
```

According to the Correct Rand Index, there is no agreement between the numerical values and the cluster solution. From -1 to +1, the agreement is very close to 0.

Meila's VI Index

```
stats$vi
```

```
## [1] 1.160317
```

Soft Clustering Approach

Model-Based Clustering

```
summary(liver$Liver_Disease)
```

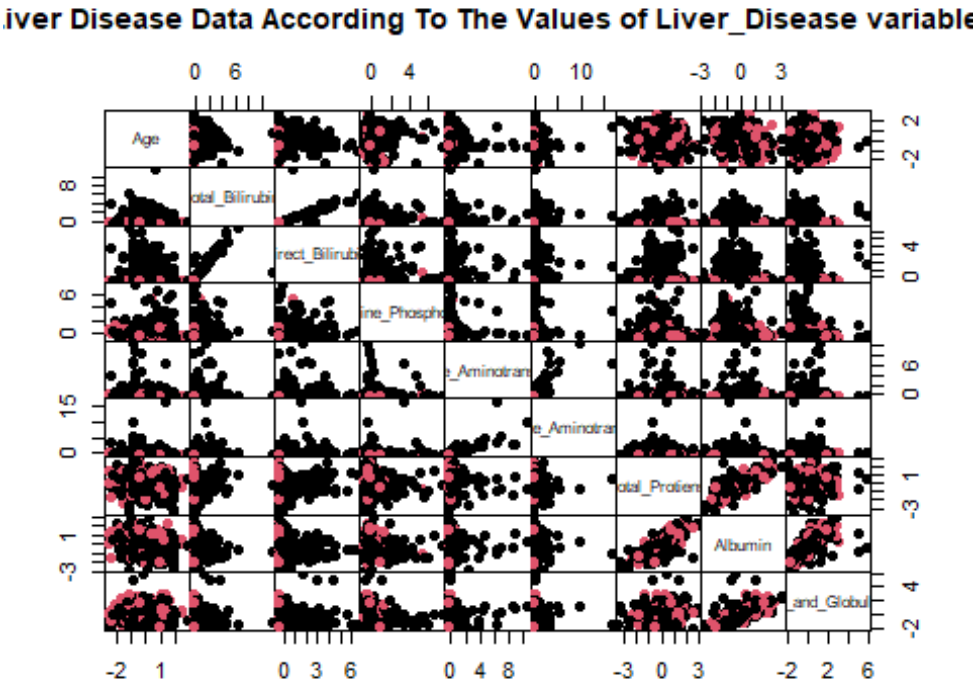
```
##   0   1
## 414 165
```

```
head(liver)
```

```
##   Age Gender Total_Bilirubin Direct_Bilirubin Alkaline_Phosphotase
## 1  65      0             0.7               0.1                  187
## 2  62      1            10.9               5.5                  699
## 3  62      1             7.3               4.1                  490
## 4  58      1             1.0               0.4                  182
## 5  72      1             3.9               2.0                  195
## 6  46      1             1.8               0.7                  208
##   Alamine_Aminotransferase Aspartate_Aminotransferase Total_Protiens
Albumin
## 1                       16                         18            6.8
3.3
## 2                       64                        100            7.5
3.2
## 3                       60                         68            7.0
3.3
## 4                       14                         20            6.8
3.4
## 5                       27                         59            7.3
2.4
```

```
## 6                          19                          14             7.6
4.4
##   Albumin_and_Globulin_Ratio Liver_Disease
## 1                       0.90             0
## 2                       0.74             0
## 3                       0.89             0
## 4                       1.00             0
## 5                       0.40             0
## 6                       1.30             0
```

```
X <- data.matrix(liver_sub)
sX <- scale(X)
pairs(sX, gap=0, pch = 16, col = as.numeric(liver$Liver_Disease), cex.main =
0.9,
      main="Liver Disease Data According To The Values of Liver_Disease
variable")
```

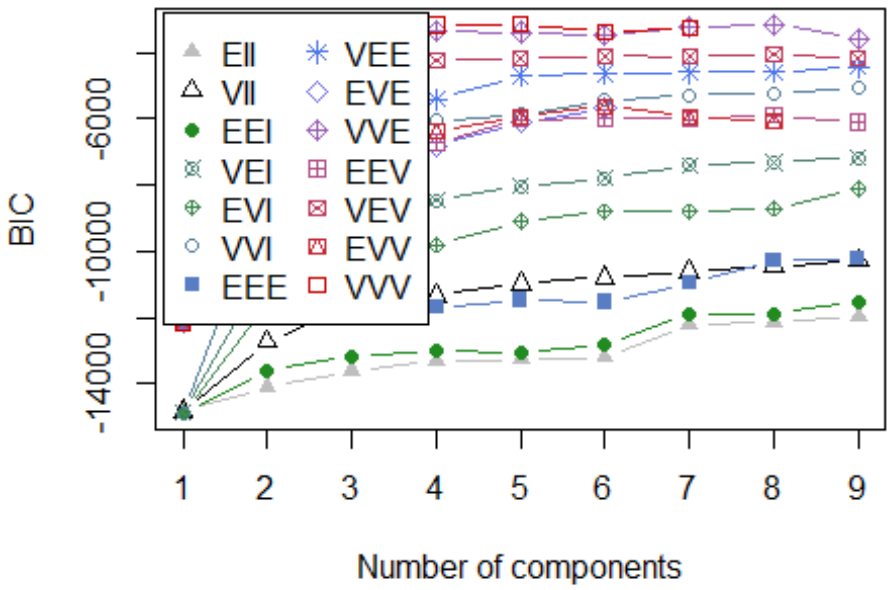


To evaluate if there is a relation between the categorical variable Liver_Disease and the underlying clustering, the variable is deleted and the data are standardized. The data are visualized by pairwise scatterplots, in which the colors represent the two possible values of Liver_Disease: `0` or `1`. It seems difficult to distinguish separate groups. Different Parsimonious Gaussian mixtures are fitted on the standardized data by using the function Mclust() in R.

```
library(mclust)
```

```
## Warning: package 'mclust' was built under R version 4.0.5
```

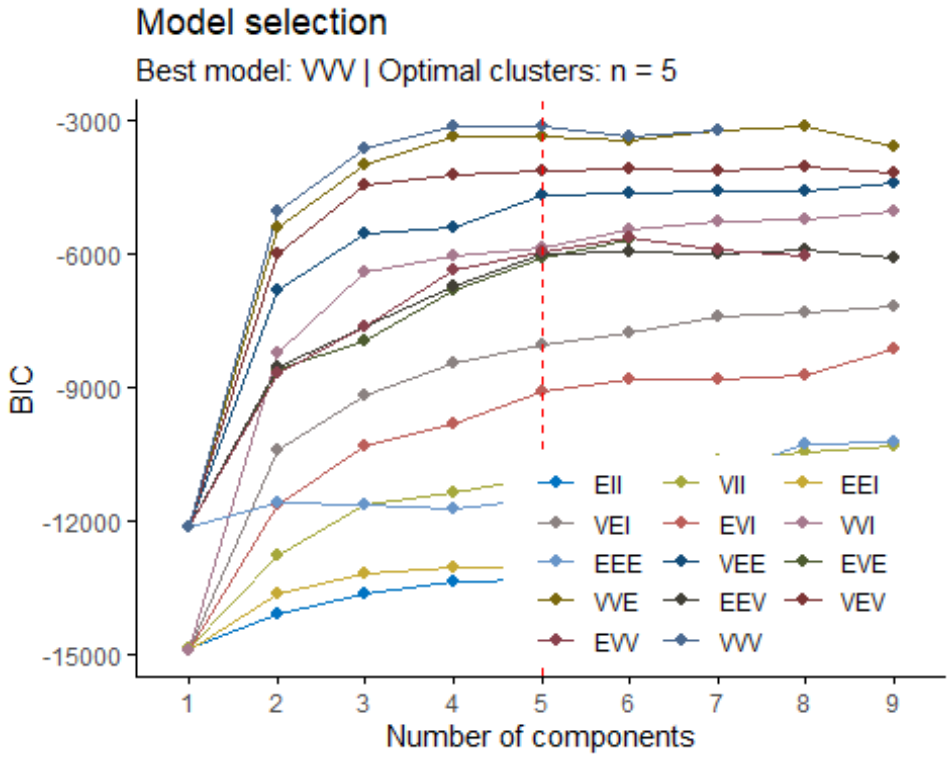
```
## Package 'mclust' version 5.4.7
## Type 'citation("mclust")' for citing this R package in publications.

##
## Attaching package: 'mclust'

## The following object is masked from 'package:psych':
##
##     sim

library(psych)
mod <- Mclust(liver_scale)
summary(mod$BIC)

## Best BIC values:
##              VVV,5        VVE,8        VVV,4
## BIC      -3155.656 -3157.139575 -3162.055214
## BIC diff     0.000    -1.484033    -6.399673

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




```
summary(mod)

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

```
## Mclust VVV (ellipsoidal, varying volume, shape, and orientation) model
with 5
## components:
##
##  log-likelihood   n  df       BIC       ICL
##       -706.3293 579 274 -3155.656 -3206.669
##
## Clustering table:
##   1   2   3   4   5
##  91  48 106 235  99
```

```
fviz_mclust(mod, "BIC", palette = "jco")
```



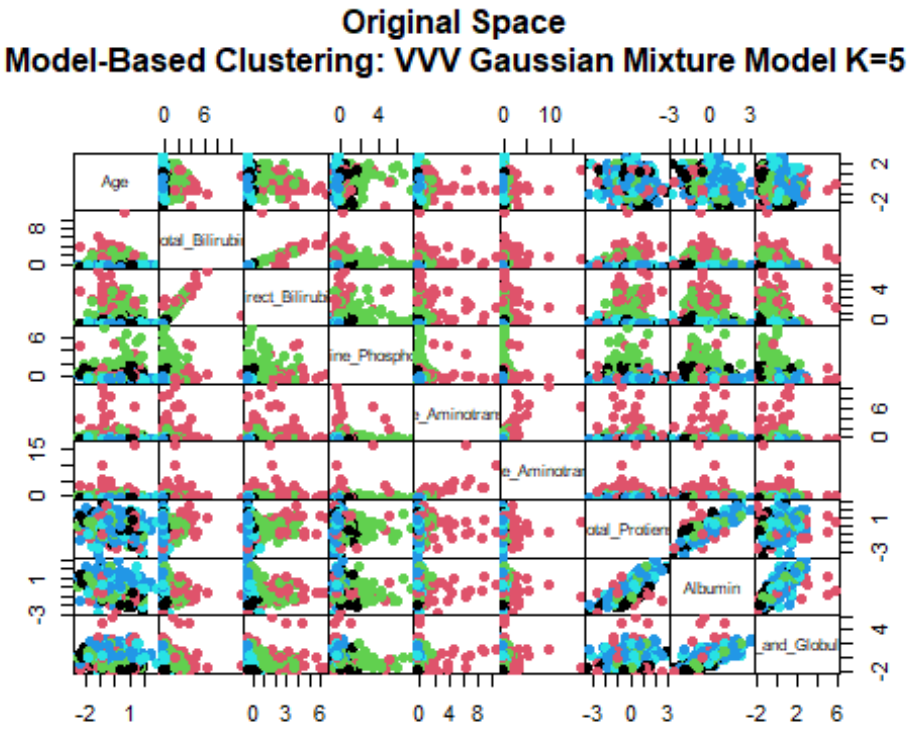

```
head(round(mod$z, 6), 20)
```

```
##        [,1]    [,2]     [,3]     [,4]     [,5]
## 1  0.001855 0.0e+00 0.000001 0.994983 0.003161
## 2  0.000000 1.1e-05 0.999989 0.000000 0.000000
## 3  0.000002 1.5e-05 0.999983 0.000000 0.000000
## 4  0.074006 0.0e+00 0.000041 0.542297 0.383656
## 5  0.999995 0.0e+00 0.000005 0.000000 0.000000
## 6  0.294785 0.0e+00 0.000135 0.000000 0.705080
## 7  0.002809 0.0e+00 0.000002 0.997005 0.000185
## 8  0.005433 0.0e+00 0.000002 0.989533 0.005032
## 9  0.004158 0.0e+00 0.000000 0.994037 0.001805
## 10 0.056921 0.0e+00 0.000146 0.906932 0.036001
## 11 0.025260 0.0e+00 0.000043 0.803215 0.171483
```

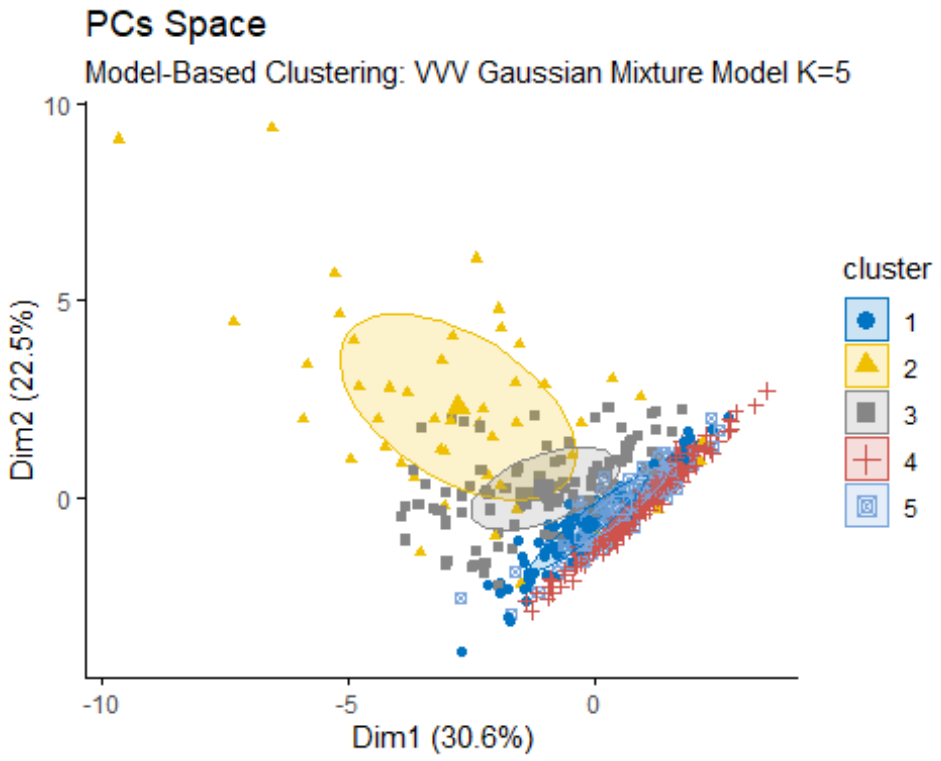
```
## 12 0.999266 0.0e+00 0.000393 0.000000 0.000342
## 13 0.667823 0.0e+00 0.001682 0.255157 0.075338
## 14 0.003366 0.0e+00 0.000004 0.841177 0.155453
## 15 0.003446 0.0e+00 0.000024 0.923137 0.073393
## 16 0.000000 0.0e+00 0.003354 0.000001 0.996646
## 17 0.000000 1.3e-04 0.999870 0.000000 0.000000
## 18 0.997169 0.0e+00 0.000368 0.000000 0.002463
## 19 0.000000 2.8e-05 0.999972 0.000000 0.000000
## 20 0.000000 2.8e-05 0.999972 0.000000 0.000000
```

According to the penalized selection criterion called "BIC" (Bayesian Information Criterion), the three best Gaussian mixture models are: VVV with 5 clusters, VVE with 8 clusters, and VVV with 4 clusters. The number of clusters that maximizes the BIC of this model is 5, cluster 1 with 91 units, cluster 2 with 48 units, cluster 3 with 106 units, cluster 4 with 235 units and cluster 5 with 99 units.

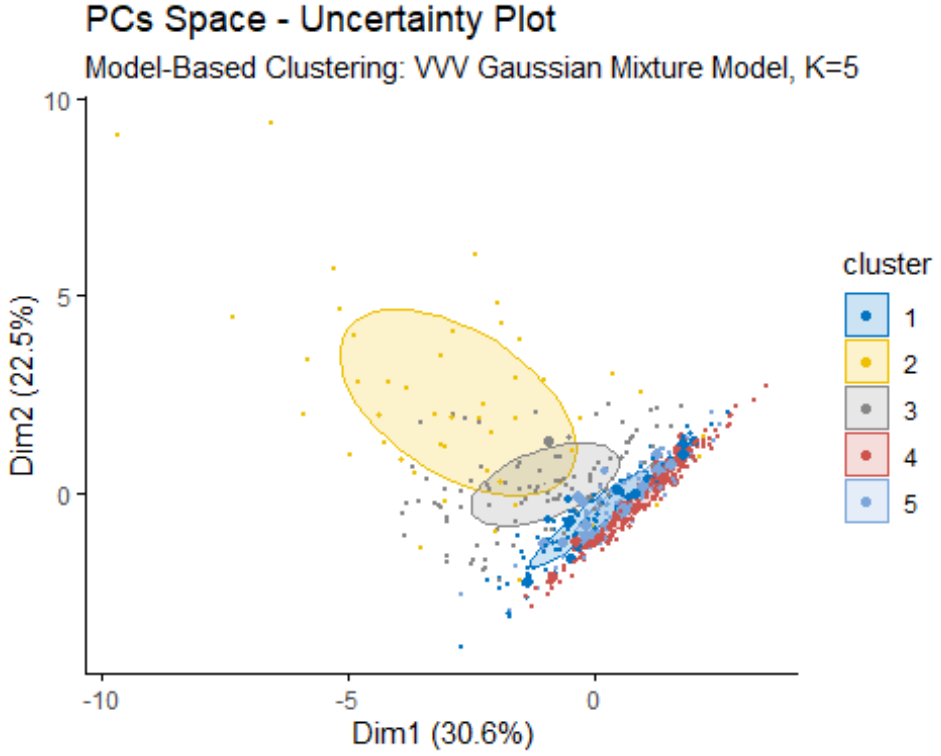
```
pairs(sX, gap=0, pch = 16, col = mod$classification, cex.main = 1,
      main="Original Space\nModel-Based Clustering: VVV Gaussian Mixture
Model K=5")
```




```
fviz_mclust(mod, "classification", geom = "point", pointsize = 1.5,
palette = "jco", main = "PCs Space") +
labs(subtitle= "Model-Based Clustering: VVV Gaussian Mixture Model K=5")
```




```
fviz_mclust(mod, "uncertainty", palette = "jco",
            main = "PCs Space - Uncertainty Plot") +
  labs(subtitle= "Model-Based Clustering: VVV Gaussian Mixture Model, K=5")
```



Both in the original space and in the PCs space there is a great separation between clusters. Furthermore, from the Uncertainty plot, it is noted that some units (big points) are problematic for the soft approach because they belong to different clusters with the same (or similar) probability.

External Validation Measures

Confusion Matrix

```
table(liver$Liver_Disease, mod$classification)
```

```
##
##       1   2   3   4   5
##   0  68  46 100 131  69
##   1  23   2   6 104  30
```

According to the Confusion matrix, there is a good agreement between the nominal variable Liver_Disease and the cluster solution. A large number of patients not having liver disease (n = 131) has been classified in cluster 4. A large number of patients having liver disease (n=104) have also been classified in the same cluster 4.

Correct Rand Index

```
adjustedRandIndex(liver$Liver_Disease, mod$classification)
```

```
## [1] -0.007492218
```

According the Correct Rand Index, there is a good agreement between the Liver_Disease nominal variable and the cluster solution.

The Best Clustering Algorithm

In order to choose the best clustering algorithm among those proposed, the `clValid` package of R. is used. The clValid function enables to compare clustering algorithms using two cluster validation measures is Internal Measures (Connectivity, Silhouette coefficient and Dunn index). The methods considered are: hierarchical method (Euclidean-Manhattan), k-Means, PAM (Euclidean-Manhattan) and Model-Based Clustering.

```
library(clValid)
```

```
## Warning: package 'clValid' was built under R version 4.0.5
```

```
clmethods <- c ("hierarchical", "kmeans", "pam")
V_eucl<-clValid(liver_scale, nClust=2:6, clMethods= clmethods,
                metric="euclidean", validation="internal")
summary(V_eucl)
```

```
##
## Clustering Methods:
##  hierarchical kmeans pam
##
## Cluster sizes:
##  2 3 4 5 6
```

```
##
## Validation Measures:
##                                   2        3        4        5        6
##
## hierarchical Connectivity    3.0956   6.0246  14.5020  18.3171  20.7254
##              Dunn            0.5318   0.4789   0.2824   0.2824   0.2824
##              Silhouette      0.8040   0.6973   0.6493   0.6061   0.5948
## kmeans       Connectivity   10.5190  40.0226  49.2111 129.9921 132.8639
##              Dunn            0.1439   0.0599   0.0769   0.0291   0.0291
##              Silhouette      0.6403   0.4761   0.4769   0.2643   0.2664
## pam          Connectivity  138.1683 147.4345 228.6667 271.0968 273.3825
##              Dunn            0.0128   0.0140   0.0083   0.0102   0.0138
##              Silhouette      0.2155   0.2386   0.1730   0.1666   0.1814
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 3.0956 hierarchical 2
## Dunn         0.5318 hierarchical 2
## Silhouette   0.8040 hierarchical 2
```

According to the result, the best clustering algorithm using the `Euclidean Distance` is the `Hierarchical Clustering` method with 2 clusters.

```
clmethods <- c ("hierarchical", "kmeans", "pam")
V_eucl <- clValid(liver_scale, nClust=2:6, clMethods= clmethods,
                  metric="manhattan", validation="internal")
summary(V_eucl)
```

```
##
## Clustering Methods:
##  hierarchical kmeans pam
##
## Cluster sizes:
##  2 3 4 5 6
##
## Validation Measures:
##                                   2        3        4        5        6
##
## hierarchical Connectivity    3.0290   7.7619  15.3579  18.2869  31.9294
##              Dunn            0.5931   0.2270   0.2672   0.2672   0.1777
##              Silhouette      0.7722   0.6542   0.6251   0.5714   0.5384
## kmeans       Connectivity   11.6222  41.8865  54.5698 128.9794 134.8909
##              Dunn            0.1647   0.0600   0.0658   0.0249   0.0298
##              Silhouette      0.6274   0.5025   0.4998   0.2662   0.2681
## pam          Connectivity  154.1667 254.6214 256.1413 277.3071 275.6333
##              Dunn            0.0120   0.0149   0.0090   0.0062   0.0081
##              Silhouette      0.2438   0.1482   0.1719   0.1680   0.1841
##
## Optimal Scores:
```

```
##
##              Score  Method       Clusters
## Connectivity 3.0290 hierarchical 2
## Dunn         0.5931 hierarchical 2
## Silhouette   0.7722 hierarchical 2
```

According to the result, the best clustering algorithm using the `Manhattan Distance` is the `Hierarchical Clustering` method with 2 clusters.

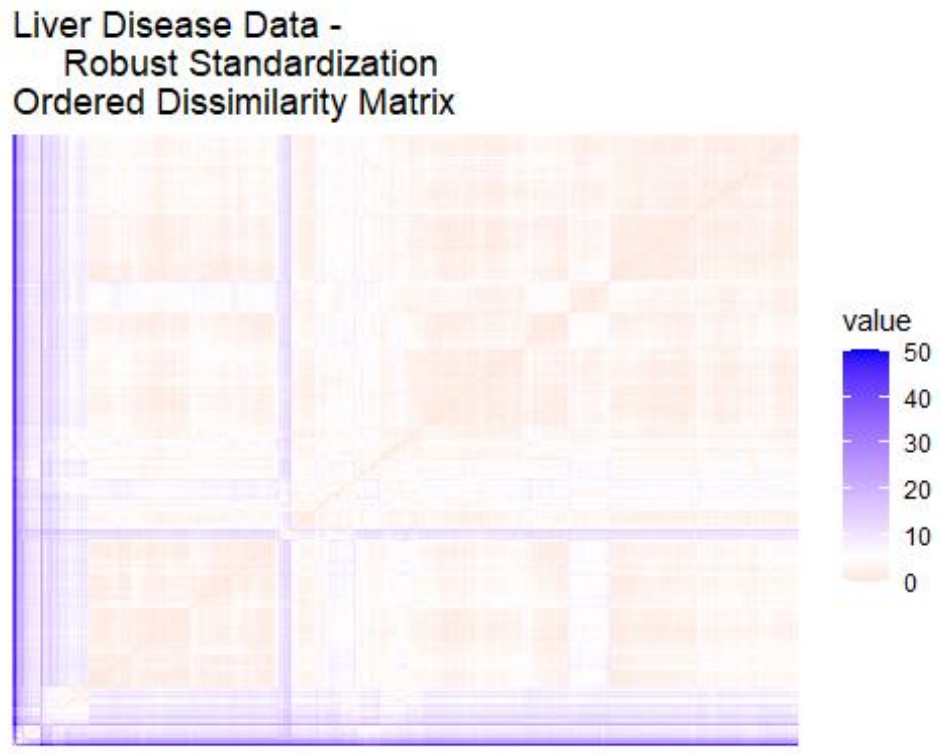
```
clmethods <- c ("hierarchical", "kmeans", "pam")
V_eucl <- clValid(liver_scale, nClust=2:6, clMethods= clmethods,
                  metric="manhattan", validation="stability")
summary(V_eucl)
```

```
##
## Clustering Methods:
##  hierarchical kmeans pam
##
## Cluster sizes:
##  2 3 4 5 6
##
## Validation Measures:
##                         2      3      4      5      6
##
## hierarchical APN  0.0011 0.0029 0.0058 0.0079 0.0124
##              AD   7.6286 7.5371 7.3617 7.3082 7.1719
##              ADM  0.0158 0.0376 0.0770 0.0831 0.1713
##              FOM  0.9686 0.9618 0.9598 0.9419 0.9405
## kmeans       APN  0.0176 0.1602 0.2178 0.0967 0.1352
##              AD   7.4381 6.8605 6.7396 5.7955 5.9327
##              ADM  0.1187 0.5450 0.6793 0.3377 0.6452
##              FOM  0.9770 0.9632 0.9142 0.8012 0.8510
## pam          APN  0.1497 0.2030 0.1966 0.3021 0.4065
##              AD   6.8566 6.5238 5.9394 5.8733 5.7696
##              ADM  0.4090 0.6878 0.5010 0.7500 0.9540
##              FOM  0.9528 0.9346 0.9075 0.9025 0.9026
##
## Optimal Scores:
##
##     Score  Method       Clusters
## APN 0.0011 hierarchical 2
## AD  5.7696 pam          6
## ADM 0.0158 hierarchical 2
## FOM 0.8012 kmeans       5
```

In this measure according to the `APN` and `ADM` the best method is `Hierarchical` with 2 clusters, according to `AD` the best method is `pam` with 6 clusters and according to `FOM` the best method is `kmeans` with 5 number of clusters.

```
scale_rob <- scale(liver_sub, center = apply(liver_sub, 2, median),
                   scale = apply(liver_sub, 2, meanabsdev))
```

```
rownames(scale_rob) <- rownames(liver)
fviz_dist(dist(scale_rob), show_labels = FALSE) +
labs(title = "Liver Disease Data -
     Robust Standardization\nOrdered Dissimilarity Matrix",
     gradient = list(low = "blue", mid = "green", high = "red"))
```




```
hopkins(scale_rob, n = nrow(scale_rob)-1)

## $H
## [1] 0.0817502
```

According to the Hopkins statistic (H) the data set is uniformly distributed because the values is close to 0, also the dissimilarity matrix image the data contain a clusters structure.

Clustering Result

According to the proposed indices, among the clustering algorithms adopted, the `Hierarchical Method`, computed using the `Euclidean Distance` with 2 clusters seems to be the most suitable.