Alcohol consumption around the world

Data analysis and statistical learning report

Professor Antonio Punzo

SALVATORE ROMANO 2021/2022 1000043560

INTRODUCTION

This report is based on a Kaggle dataset called Alcohol consumption around the world.

This dataset shows how much alcohol is consumed by people of various countries around the world. There are many substances that lies in the category of alcohol. In this dataset, only beer, spirit and wine are taken into consideration.

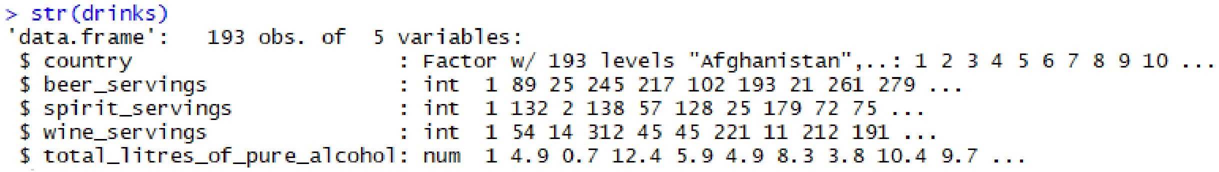
Contains the data behind the story "Dear Mona Follow-up: Where Do People Drink The Most Beer, Wine And Spirits?"

A consideration that must be done is that in the dataset there is a high presence of 1 value, indicates that the type of alcohol is not legal in this country or that there weren't data for this country, but to compute I inserted this value.

Of course, the analysis of this dataset has been processed with the use of RStudio.

DATA STRUCTURE DESCRIPTION

As we can see in RStudio, the dataset is made by 193 observations of 5 variables; it can be displayed with the command str().

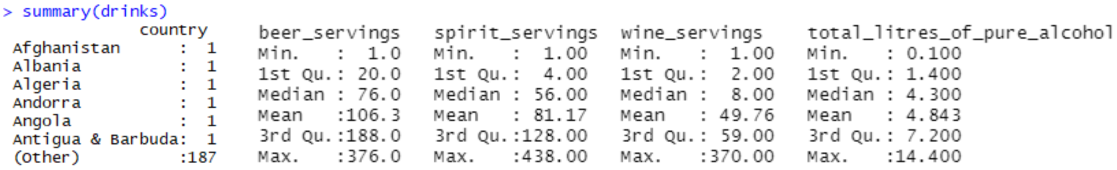


```
> str(drinks)
'data.frame':	193 obs. of  5 variables:
 $ country                     : Factor w/ 193 levels "Afghanistan",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ beer_servings               : int  1 89 25 245 217 102 193 21 261 279 ...
 $ spirit_servings             : int  1 132 2 138 57 128 25 179 72 75 ...
 $ wine_servings               : int  1 54 14 312 45 45 221 11 212 191 ...
 $ total_litres_of_pure_alcohol: num  1 4.9 0.7 12.4 5.9 4.9 8.3 3.8 10.4 9.7 ...
```

The variable types are:

- **Country:** categorical variable with 193 levels (of course we are talking about world country)
- **Beer_servings:** average beer serving per person, is a continuous numerical variable
- **Spirit_servings:** average spirit serving per person, is a continuous numerical variable
- **Wine_servings:** average wine serving per person, is a continuous numerical variable
- **Total_litres_of_pure_alcohol:** total litres of pure alcohol served, is a continuous numerical variable

In order to know some information based on the dataset, we can use a function called summary(); his output is the largest value in data, the least value or mean and median and another similar type of information.

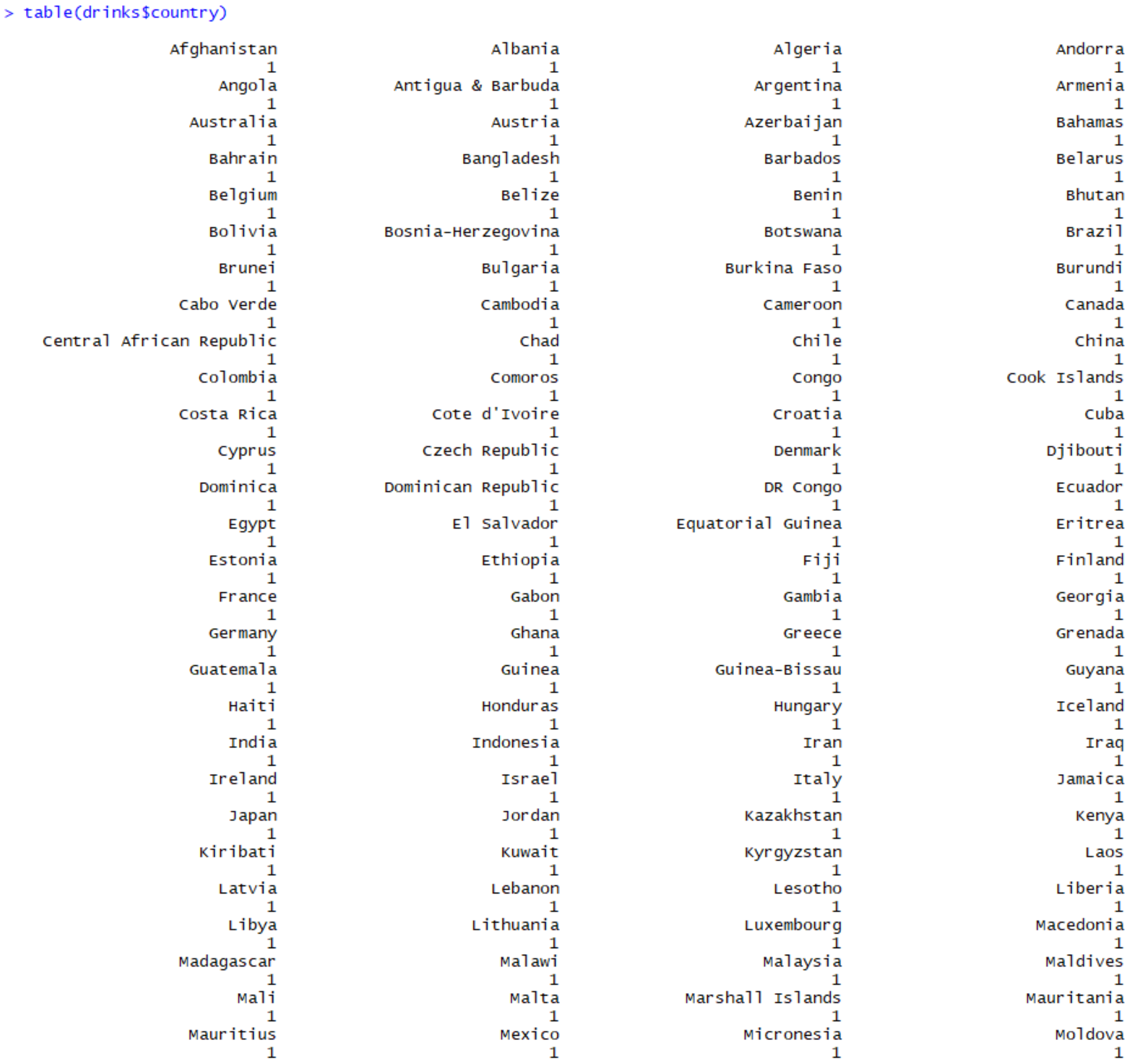


```
> summary(drinks)
               country     beer_servings    spirit_servings  wine_servings    total_litres_of_pure_alcohol
 Afghanistan      :  1   Min.   :  1.0   Min.   :  1.00   Min.   :  1.00   Min.   : 0.100              
 Albania          :  1   1st Qu.: 20.0   1st Qu.:  4.00   1st Qu.:  2.00   1st Qu.: 1.400              
 Algeria          :  1   Median : 76.0   Median : 56.00   Median :  8.00   Median : 4.300              
 Andorra          :  1   Mean   :106.3   Mean   : 81.17   Mean   : 49.76   Mean   : 4.843              
 Angola           :  1   3rd Qu.:188.0   3rd Qu.:128.00   3rd Qu.: 59.00   3rd Qu.: 7.200              
 Antigua & Barbuda:  1   Max.   :376.0   Max.   :438.00   Max.   :370.00   Max.   :14.400              
 (Other)          :187                                                                                 
```

UNIVARIATE ANALYSIS

This chapter aim to analyse the variable of the dataset.

***1.Country*:** as we said before, the Country is a categorical variable, that indicates the country of the world based on our study and dataset.
Of course, due to the type of categorical variable, the table() output will be this because the variable Country has 193 levels:

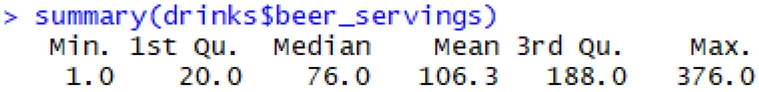


```
> table(drinks$country)

             Afghanistan                  Albania                  Algeria                  Andorra 
                       1                        1                        1                        1 
                  Angola        Antigua & Barbuda                Argentina                  Armenia 
                       1                        1                        1                        1 
               Australia                  Austria               Azerbaijan                  Bahamas 
                       1                        1                        1                        1 
                 Bahrain               Bangladesh                 Barbados                  Belarus 
                       1                        1                        1                        1 
                 Belgium                   Belize                    Benin                   Bhutan 
                       1                        1                        1                        1 
                 Bolivia       Bosnia-Herzegovina                 Botswana                   Brazil 
                       1                        1                        1                        1 
                  Brunei                 Bulgaria             Burkina Faso                  Burundi 
                       1                        1                        1                        1 
              Cabo Verde                 Cambodia                 Cameroon                   Canada 
                       1                        1                        1                        1 
Central African Republic                     Chad                    Chile                    China 
                       1                        1                        1                        1 
                Colombia                  Comoros                    Congo             Cook Islands 
                       1                        1                        1                        1 
              Costa Rica            Cote d'Ivoire                  Croatia                     Cuba 
                       1                        1                        1                        1 
                  Cyprus           Czech Republic                  Denmark                 Djibouti 
                       1                        1                        1                        1 
                Dominica       Dominican Republic                 DR Congo                  Ecuador 
                       1                        1                        1                        1 
                   Egypt              El Salvador        Equatorial Guinea                  Eritrea 
                       1                        1                        1                        1 
                 Estonia                 Ethiopia                     Fiji                  Finland 
                       1                        1                        1                        1 
                  France                    Gabon                   Gambia                  Georgia 
                       1                        1                        1                        1 
                 Germany                    Ghana                   Greece                  Grenada 
                       1                        1                        1                        1 
               Guatemala                   Guinea            Guinea-Bissau                   Guyana 
                       1                        1                        1                        1 
                   Haiti                 Honduras                  Hungary                  Iceland 
                       1                        1                        1                        1 
                   India                Indonesia                     Iran                     Iraq 
                       1                        1                        1                        1 
                 Ireland                   Israel                    Italy                  Jamaica 
                       1                        1                        1                        1 
                   Japan                   Jordan               Kazakhstan                    Kenya 
                       1                        1                        1                        1 
                Kiribati                   Kuwait               Kyrgyzstan                     Laos 
                       1                        1                        1                        1 
                  Latvia                  Lebanon                  Lesotho                  Liberia 
                       1                        1                        1                        1 
                   Libya                Lithuania               Luxembourg                Macedonia 
                       1                        1                        1                        1 
              Madagascar                   Malawi                 Malaysia                 Maldives 
                       1                        1                        1                        1 
                    Mali                    Malta         Marshall Islands               Mauritania 
                       1                        1                        1                        1 
               Mauritius                   Mexico               Micronesia                  Moldova 
                       1                        1                        1                        1 
```

```
                      Monaco                     Mongolia                   Montenegro                      Morocco
                           1                            1                            1                            1
                  Mozambique                      Myanmar                      Namibia                        Nauru
                           1                            1                            1                            1
                       Nepal                  Netherlands                  New Zealand                    Nicaragua
                           1                            1                            1                            1
                       Niger                      Nigeria                         Niue                  North Korea
                           1                            1                            1                            1
                      Norway                         Oman                     Pakistan                        Palau
                           1                            1                            1                            1
                      Panama             Papua New Guinea                     Paraguay                         Peru
                           1                            1                            1                            1
                 Philippines                       Poland                     Portugal                        Qatar
                           1                            1                            1                            1
                     Romania           Russian Federation                       Rwanda                        Samoa
                           1                            1                            1                            1
                  San Marino          Sao Tome & Principe                 Saudi Arabia                      Senegal
                           1                            1                            1                            1
                      Serbia                   Seychelles                 Sierra Leone                    Singapore
                           1                            1                            1                            1
                    Slovakia                     Slovenia              Solomon Islands                      Somalia
                           1                            1                            1                            1
                South Africa                  South Korea                        Spain                    Sri Lanka
                           1                            1                            1                            1
           St. Kitts & Nevis                    St. Lucia St. Vincent & the Grenadines                        Sudan
                           1                            1                            1                            1
                    Suriname                    Swaziland                       Sweden                  Switzerland
                           1                            1                            1                            1
                       Syria                   Tajikistan                     Tanzania                     Thailand
                           1                            1                            1                            1
                 Timor-Leste                         Togo                        Tonga            Trinidad & Tobago
                           1                            1                            1                            1
                     Tunisia                       Turkey                 Turkmenistan                       Tuvalu
                           1                            1                            1                            1
                      Uganda                      Ukraine         United Arab Emirates               United Kingdom
                           1                            1                            1                            1
                     Uruguay                          USA                   Uzbekistan                      Vanuatu
                           1                            1                            1                            1
                   Venezuela                      Vietnam                        Yemen                       Zambia
                           1                            1                            1                            1
                    Zimbabwe
                           1
```

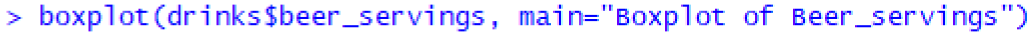
So, the frequency, obviously is set 1 at every country cause of the study.

2.Beer_servings: as we said before, average beer serving per person, is a continuous numerical variable, that assume value included in [1;376]. All the information of the variable Beer_servings can be displayed with the summary(drinks$beer_servings).

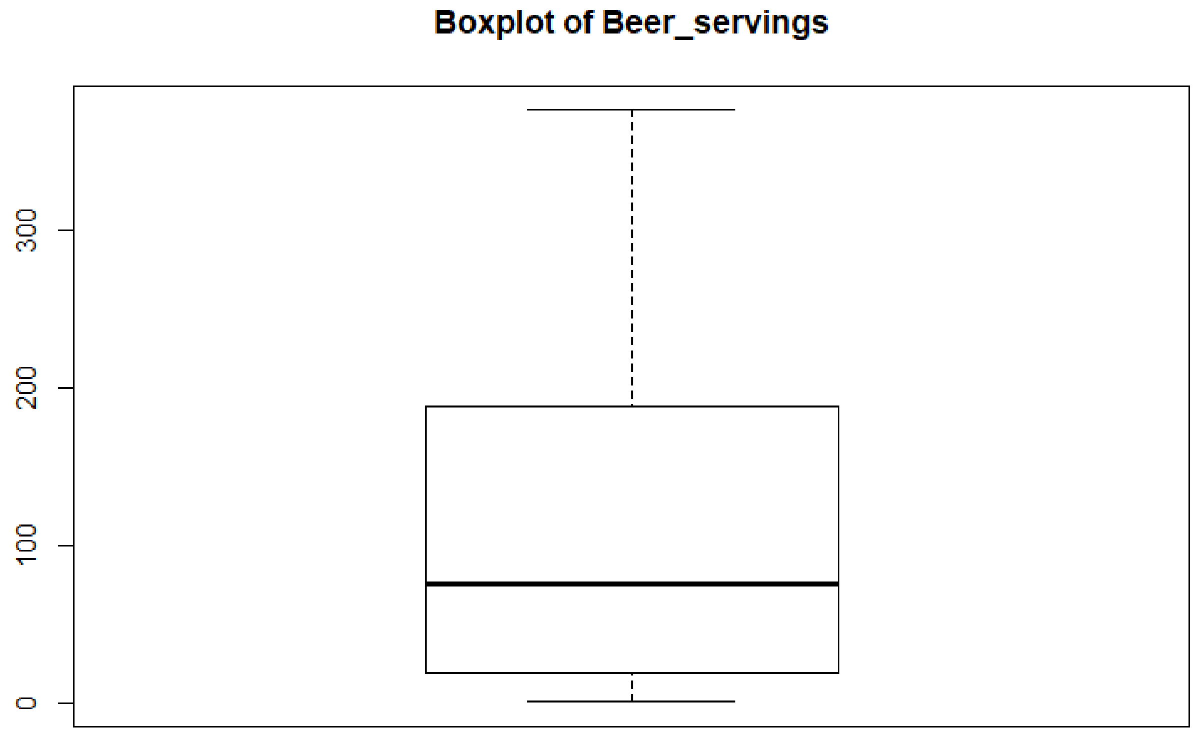


```
> summary(drinks$beer_servings)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    1.0    20.0    76.0   106.3   188.0   376.0 
```

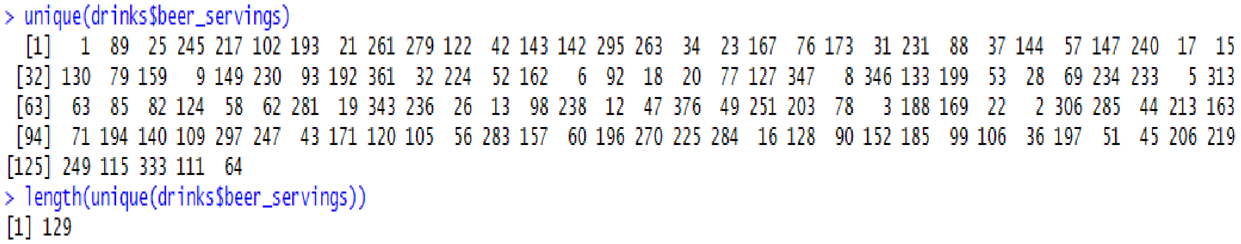
So, we can attest that the minimum average beer serving per person is 1, the maximum is 376; also, the 25% of average beer serving per person is at least 20, the 50% of average beer serving per person is at least 76, and finally, the 75% of average beer serving per person is at least 188. We can also plot a boxplot of Beer_servings.



```
> boxplot(drinks$beer_servings, main="Boxplot of Beer_servings")
```

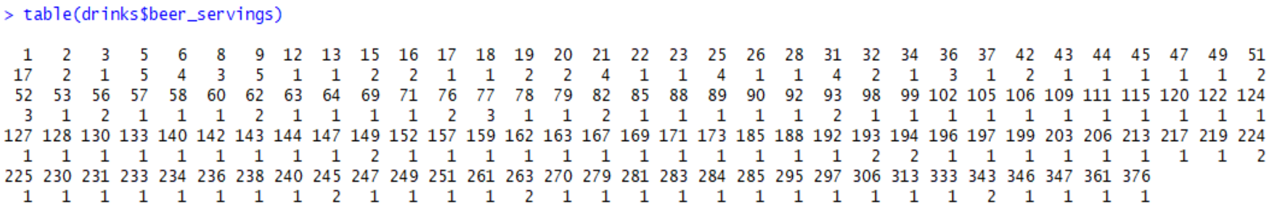


Going on, we can discover more about this variable, for example all the values that can assumes and the absolute frequencies of the variable.



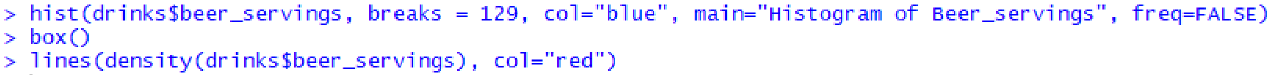
```
> unique(drinks$beer_servings)
  [1]   1  89  25 245 217 102 193  21 261 279 122  42 143 142 295 263  34  23 167  76 173  31 231  88  37 144  57 147 240  17  15
 [32] 130  79 159   9 149 230  93 192 361  32 224  52 162   6  92  18  20  77 127 347   8 346 133 199  53  28  69 234 233   5 313
 [63]  63  85  82 124  58  62 281  19 343 236  26  13  98 238  12  47 376  49 251 203  78   3 188 169  22   2 306 285  44 213 163
 [94]  71 194 140 109 297 247  43 171 120 105  56 283 157  60 196 270 225 284  16 128  90 152 185  99 106  36 197  51  45 206 219
[125] 249 115 333 111  64
> length(unique(drinks$beer_servings))
[1] 129
```

The Beer_servings variable can assume 129 different values.

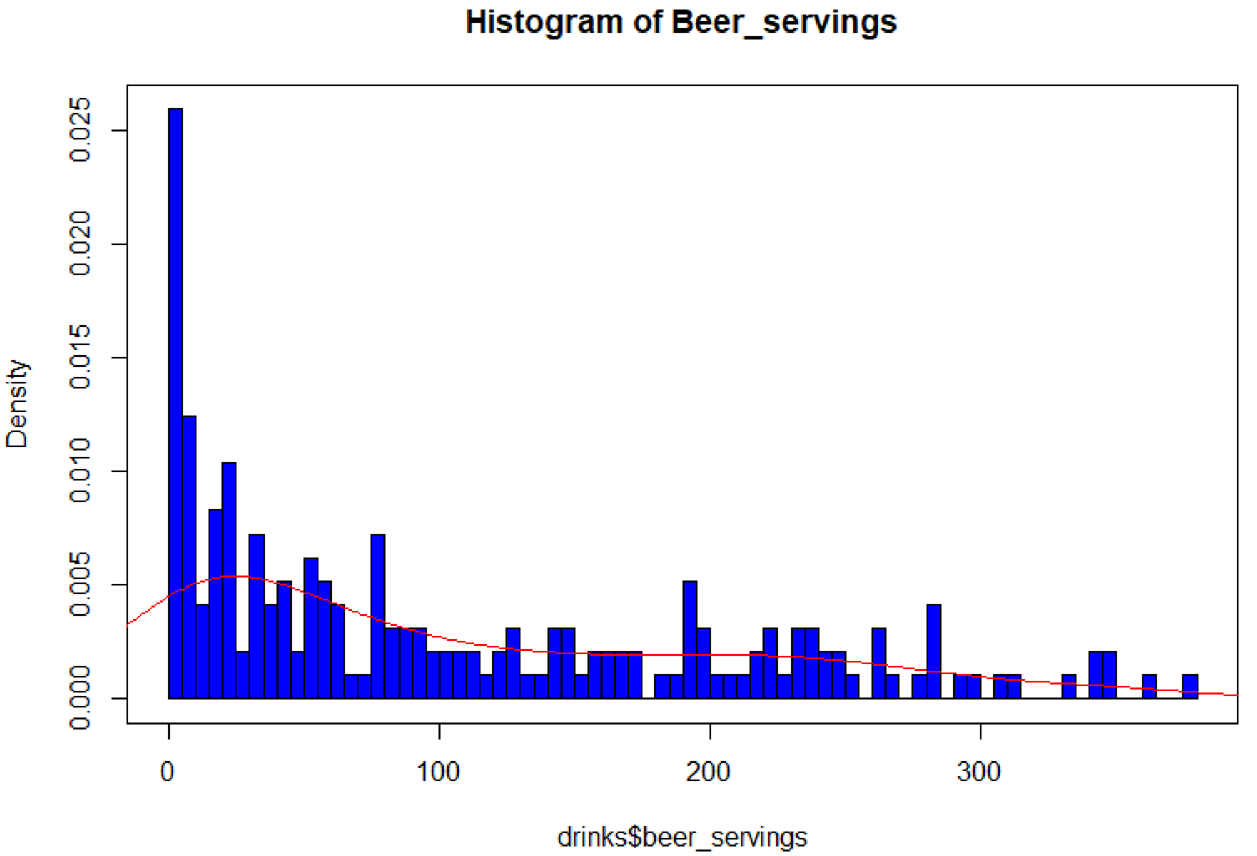


```
> table(drinks$beer_servings)

  1   2   3   5   6   8   9  12  13  15  16  17  18  19  20  21  22  23  25  26  28  31  32  34  36  37  42  43  44  45  47  49  51 
 17   2   1   5   4   3   5   1   1   2   2   1   1   2   2   4   1   1   4   1   1   4   2   1   3   1   2   1   1   1   1   1   2 
 52  53  56  57  58  60  62  63  64  69  71  76  77  78  79  82  85  88  89  90  92  93  98  99 102 105 106 109 111 115 120 122 124 
  3   1   2   1   1   1   2   1   1   1   1   2   3   1   1   2   1   1   1   1   1   2   1   1   1   1   1   1   1   1   1   1   1 
127 128 130 133 140 142 143 144 147 149 152 157 159 162 163 167 169 171 173 185 188 192 193 194 196 197 199 203 206 213 217 219 224 
  1   1   1   1   1   1   1   1   1   2   1   1   1   1   1   1   1   1   1   1   1   1   2   2   1   1   1   1   1   1   1   1   2 
225 230 231 233 234 236 238 240 245 247 249 251 261 263 270 279 281 283 284 285 295 297 306 313 333 343 346 347 361 376 
  1   1   1   1   1   1   1   1   2   1   1   1   1   2   1   1   1   1   1   1   1   1   1   1   1   2   1   1   1   1 
```

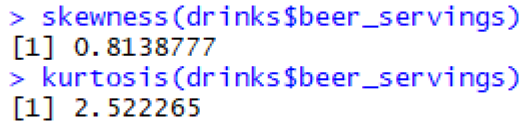
We can see the movement of the variable with the use of a histogram, trough the following commands.



```
> hist(drinks$beer_servings, breaks = 129, col="blue", main="Histogram of Beer_servings", freq=FALSE)
> box()
> lines(density(drinks$beer_servings), col="red")
```

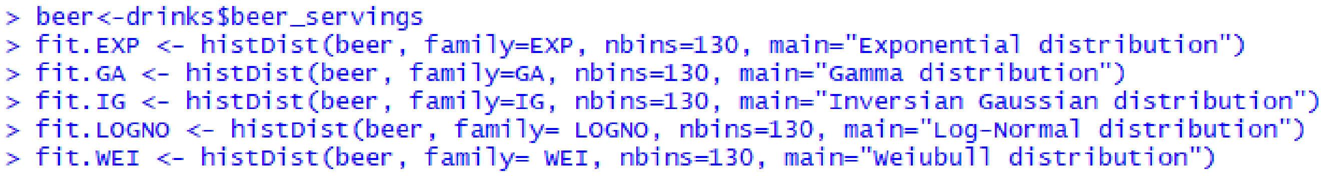


In the picture, the red line represents the density of the distribution of the Beer_servings values. Next the computation of skewness and kurtosis.

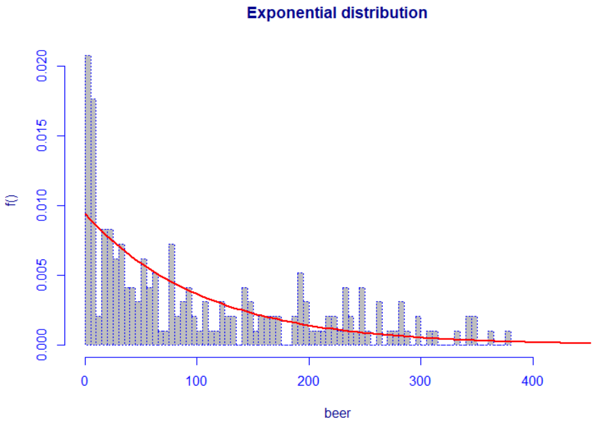


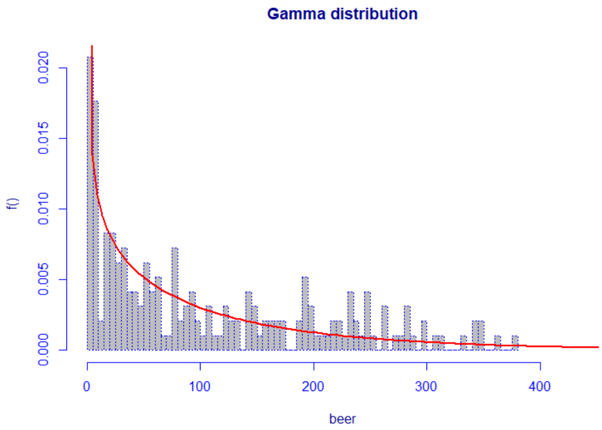
```
> skewness(drinks$beer_servings)
[1] 0.8138777
> kurtosis(drinks$beer_servings)
[1] 2.522265
```

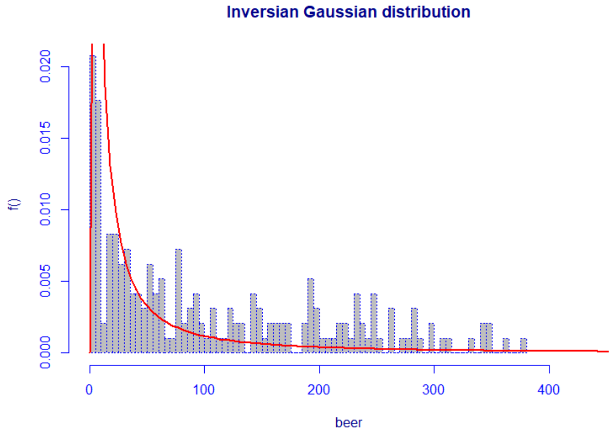
Due to the skewness value (0.8138777) that is near to 1, we can affirm that the Beer_servings distribution is right skewed; means that most of the distribution is at the left of the hist.
The kurtosis value (2.522265) is above 0, so the distribution is leptokurtic.
Next thing to do is the data fitting of Beer_servings distribution.
The goal is to find the best model that fits the Beer_servings distribution.

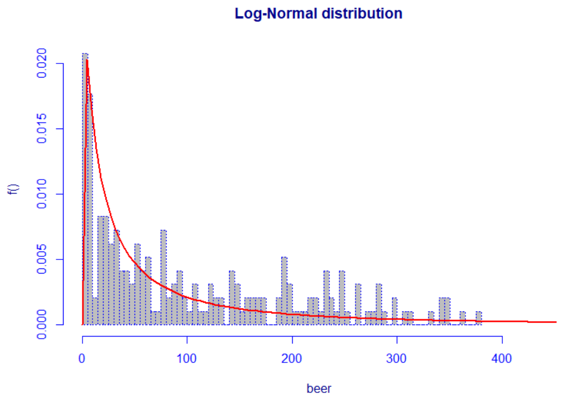


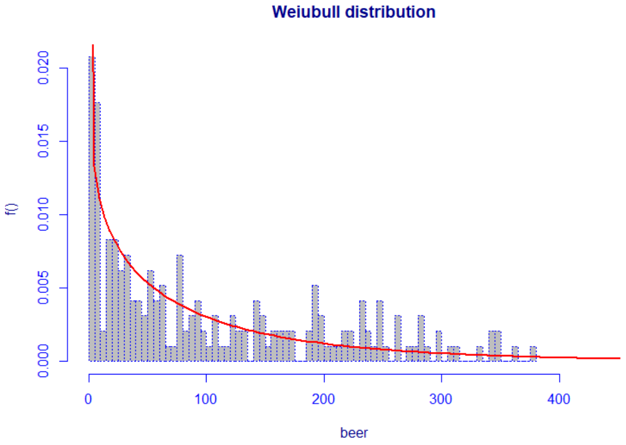
```
> beer<-drinks$beer_servings
> fit.EXP <- histDist(beer, family=EXP, nbins=130, main="Exponential distribution")
> fit.GA <- histDist(beer, family=GA, nbins=130, main="Gamma distribution")
> fit.IG <- histDist(beer, family=IG, nbins=130, main="Inversian Gaussian distribution")
> fit.LOGNO <- histDist(beer, family= LOGNO, nbins=130, main="Log-Normal distribution")
> fit.WEI <- histDist(beer, family= WEI, nbins=130, main="Weiubull distribution")
```



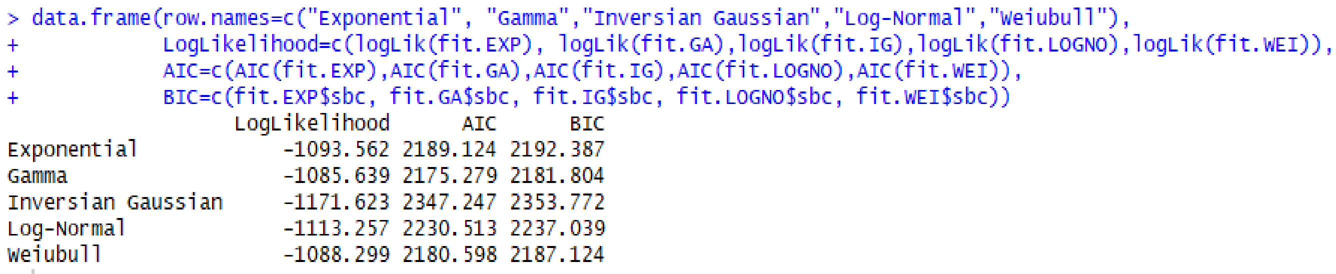








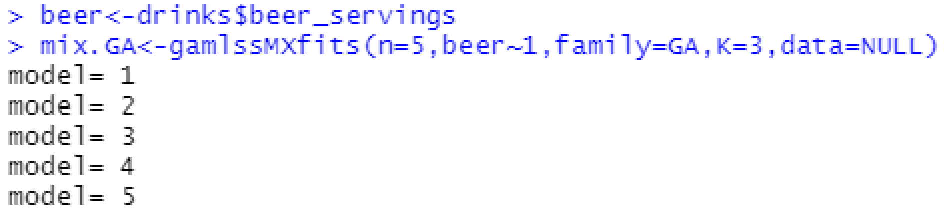
Now, evaluation of the best model that can fit the variable by taking in account both AIC and BIC indexes. The lower are the indexes, the better is the fitting.



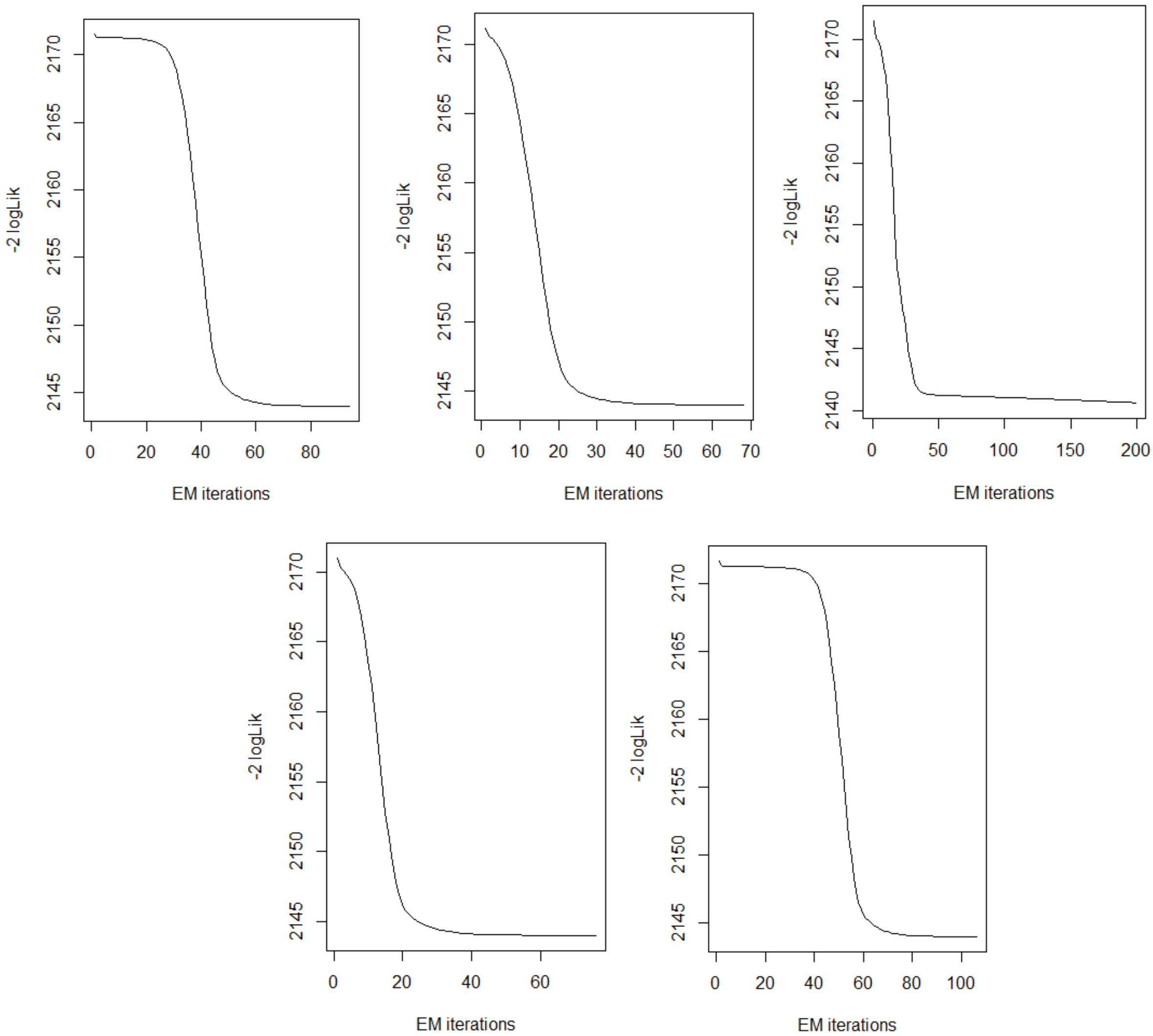
```
> data.frame(row.names=c("Exponential", "Gamma","Inversian Gaussian","Log-Normal","Weiubull"),
+           LogLikelihood=c(logLik(fit.EXP), logLik(fit.GA),logLik(fit.IG),logLik(fit.LOGNO),logLik(fit.WEI)),
+           AIC=c(AIC(fit.EXP),AIC(fit.GA),AIC(fit.IG),AIC(fit.LOGNO),AIC(fit.WEI)),
+           BIC=c(fit.EXP$sbc, fit.GA$sbc, fit.IG$sbc, fit.LOGNO$sbc, fit.WEI$sbc))
                   LogLikelihood      AIC      BIC
Exponential            -1093.562 2189.124 2192.387
Gamma                  -1085.639 2175.279 2181.804
Inversian Gaussian     -1171.623 2347.247 2353.772
Log-Normal             -1113.257 2230.513 2237.039
Weiubull               -1088.299 2180.598 2187.124
```

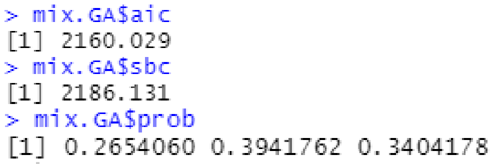
Finally, according with the parameters comes out from the computation, the best model to fit our Beer_servings data is the Gamma distribution.

In the next step we'll try a mixture of 3 Gamma distribution.

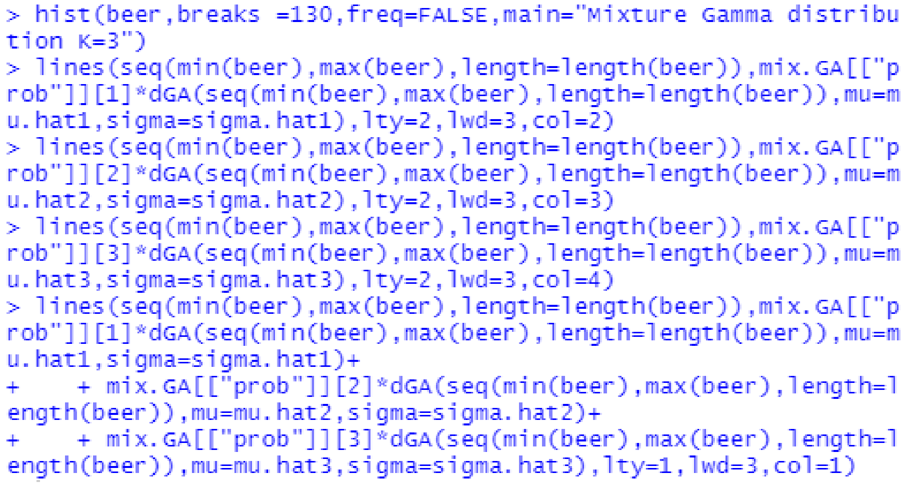


```
> beer<-drinks$beer_servings
> mix.GA<-gamlssMXfits(n=5,beer~1,family=GA,K=3,data=NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
```

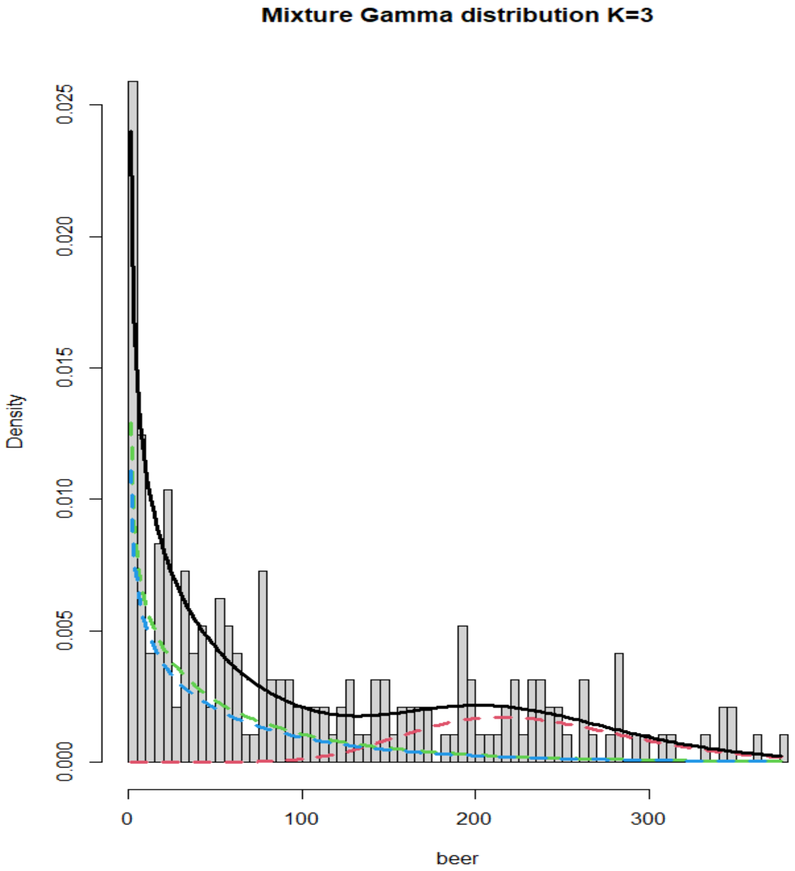




```
> mix.GA$aic
[1] 2160.029
> mix.GA$sbc
[1] 2186.131
> mix.GA$prob
[1] 0.2654060 0.3941762 0.3404178
```

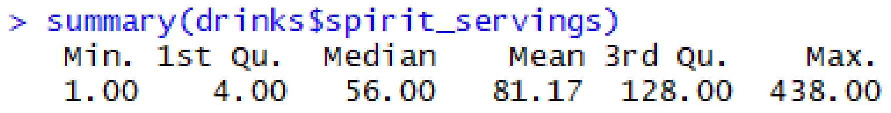


```
> hist(beer,breaks =130,freq=FALSE,main="Mixture Gamma distribu
tion K=3")
> lines(seq(min(beer),max(beer),length=length(beer)),mix.GA[["p
rob"]][1]*dGA(seq(min(beer),max(beer),length=length(beer)),mu=m
u.hat1,sigma=sigma.hat1),lty=2,lwd=3,col=2)
> lines(seq(min(beer),max(beer),length=length(beer)),mix.GA[["p
rob"]][2]*dGA(seq(min(beer),max(beer),length=length(beer)),mu=m
u.hat2,sigma=sigma.hat2),lty=2,lwd=3,col=3)
> lines(seq(min(beer),max(beer),length=length(beer)),mix.GA[["p
rob"]][3]*dGA(seq(min(beer),max(beer),length=length(beer)),mu=m
u.hat3,sigma=sigma.hat3),lty=2,lwd=3,col=4)
> lines(seq(min(beer),max(beer),length=length(beer)),mix.GA[["p
rob"]][1]*dGA(seq(min(beer),max(beer),length=length(beer)),mu=m
u.hat1,sigma=sigma.hat1)+
+     + mix.GA[["prob"]][2]*dGA(seq(min(beer),max(beer),length=l
ength(beer)),mu=mu.hat2,sigma=sigma.hat2)+
+     + mix.GA[["prob"]][3]*dGA(seq(min(beer),max(beer),length=l
ength(beer)),mu=mu.hat3,sigma=sigma.hat3),lty=1,lwd=3,col=1)
```



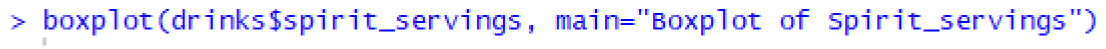
According to the image above, we have the black line that is the overall mixture of K=3, and then the other distribution, the red, blue and green.

3.Spirit_servings: as we said before, this variable indicates average spirit serving per person, and it's a continuous numerical variable. It assumes values included in [1;438].

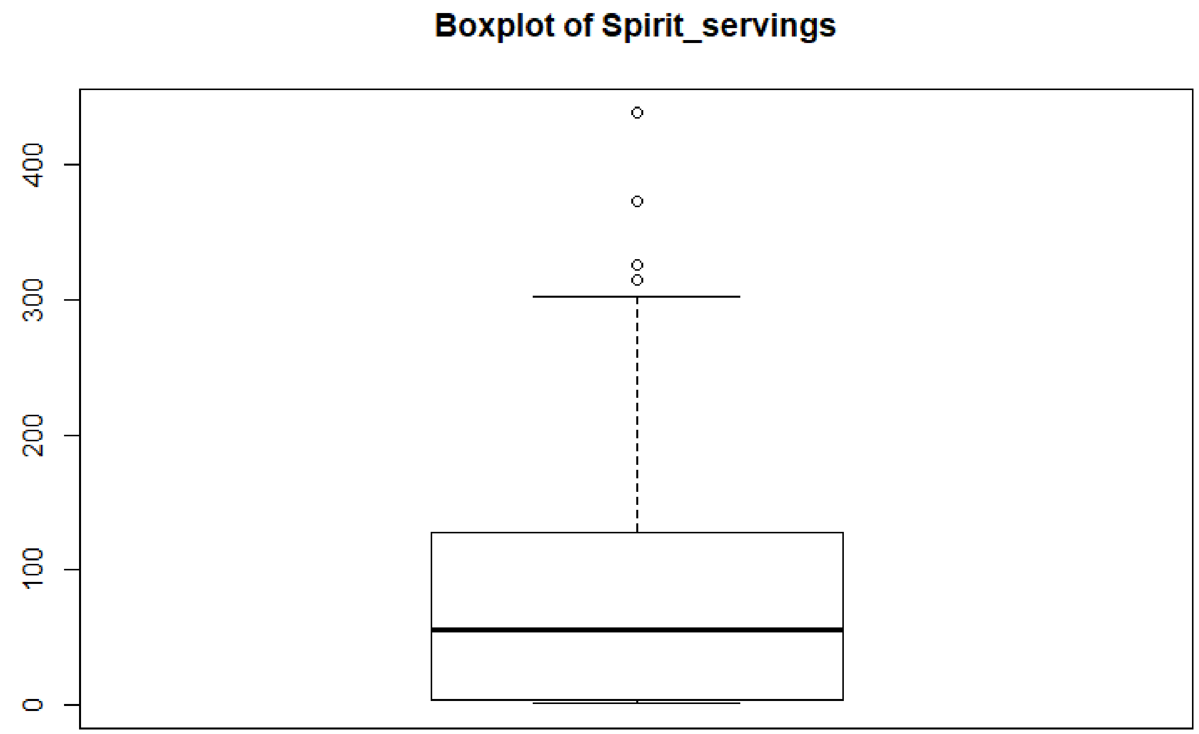


```
> summary(drinks$spirit_servings)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   1.00    4.00   56.00   81.17  128.00  438.00 
```

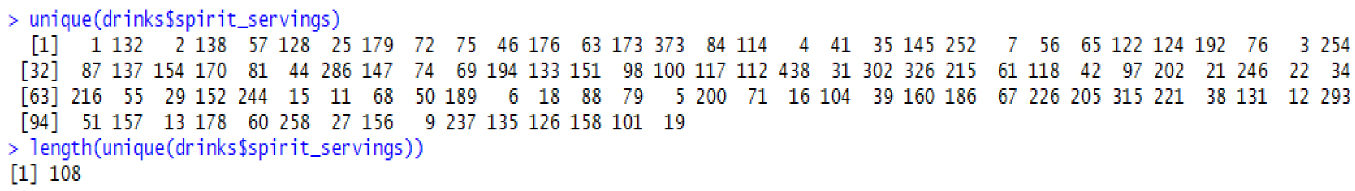
The minimum average spirit serving per person is 1, and the maximum is 438. According to this code, the 25% of average spirit serving per person is at least 4, the 50% of average spirit serving per person is at least 56, and the 75% of average spirit serving per person is at least 128.
Next, we are going to plot the boxplot of the variable.



```
> boxplot(drinks$spirit_servings, main="Boxplot of Spirit_servings")
```

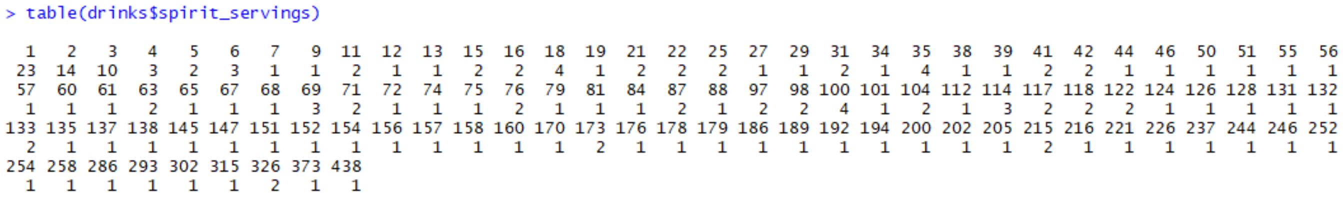


Now we can compute all the values that Spirit_serving assumes and the absolute frequencies of the variable.



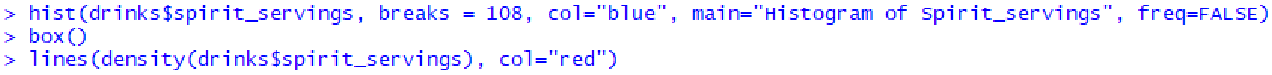
```
> unique(drinks$spirit_servings)
  [1]   1 132   2 138  57 128  25 179  72  75  46 176  63 173 373  84 114   4  41  35 145 252   7  56  65 122 124 192  76   3 254
 [32]  87 137 154 170  81  44 286 147  74  69 194 133 151  98 100 117 112 438  31 302 326 215  61 118  42  97 202  21 246  22  34
 [63] 216  55  29 152 244  15  11  68  50 189   6  18  88  79   5 200  71  16 104  39 160 186  67 226 205 315 221  38 131  12 293
 [94]  51 157  13 178  60 258  27 156   9 237 135 126 158 101  19
> length(unique(drinks$spirit_servings))
[1] 108
```

This variable can assume 108 different values.

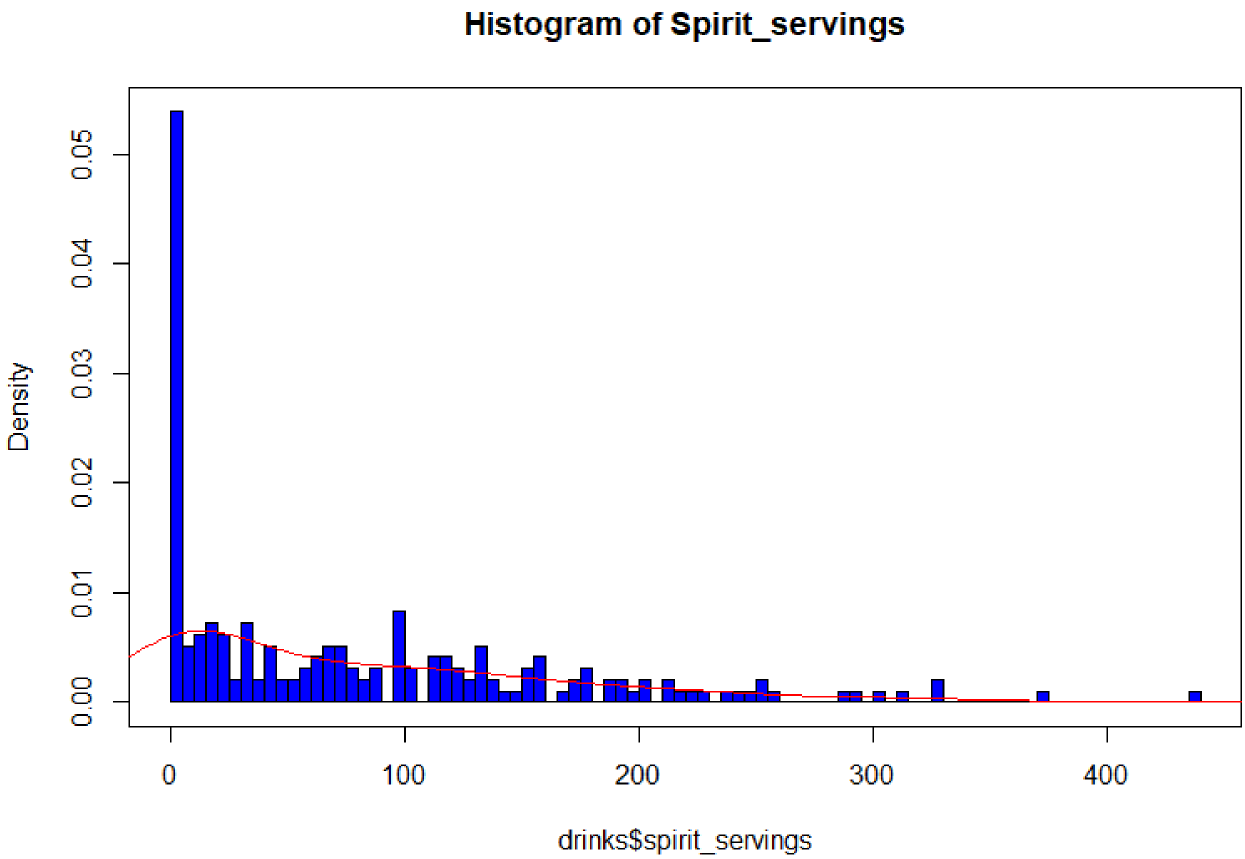


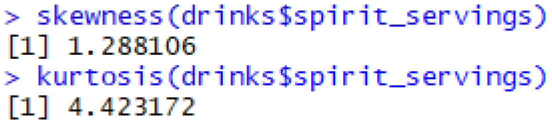
```
> table(drinks$spirit_servings)

  1   2   3   4   5   6   7   9  11  12  13  15  16  18  19  21  22  25  27  29  31  34  35  38  39  41  42  44  46  50  51  55  56 
 23  14  10   3   2   3   1   1   2   1   1   2   2   4   1   2   2   2   1   1   2   1   4   1   1   2   2   1   1   1   1   1   1 
 57  60  61  63  65  67  68  69  71  72  74  75  76  79  81  84  87  88  97  98 100 101 104 112 114 117 118 122 124 126 128 131 132 
  1   1   1   2   1   1   1   3   2   1   1   1   2   1   1   1   2   1   2   2   4   1   2   1   3   2   2   2   1   1   1   1   1 
133 135 137 138 145 147 151 152 154 156 157 158 160 170 173 176 178 179 186 189 192 194 200 202 205 215 216 221 226 237 244 246 252 
  2   1   1   1   1   1   1   1   1   1   1   1   1   1   2   1   1   1   1   1   1   1   1   1   1   2   1   1   1   1   1   1   1 
254 258 286 293 302 315 326 373 438 
  1   1   1   1   1   1   2   1   1 
```

Remember that the table() function gives as output the variable name and the frequency.
Now compute the histogram and see the information about the absolute frequencies.



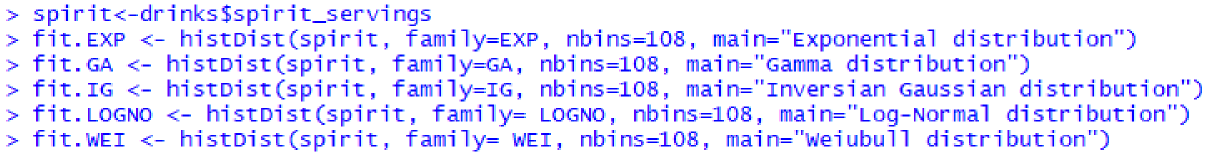
```
> hist(drinks$spirit_servings, breaks = 108, col="blue", main="Histogram of Spirit_servings", freq=FALSE)
> box()
> lines(density(drinks$spirit_servings), col="red")
```



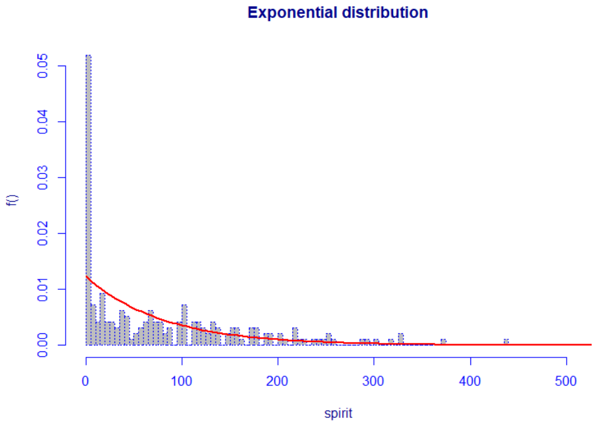


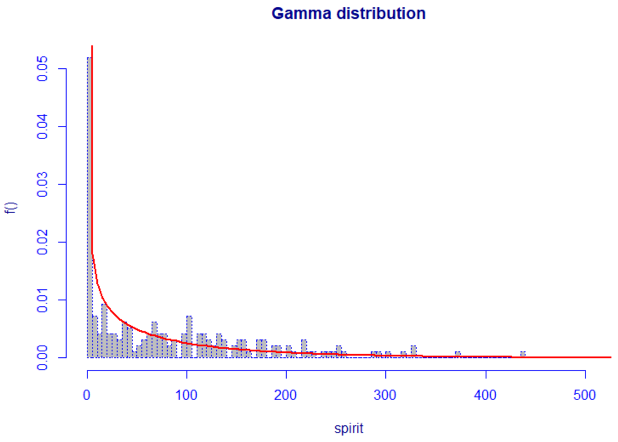
```
> skewness(drinks$spirit_servings)
[1] 1.288106
> kurtosis(drinks$spirit_servings)
[1] 4.423172
```

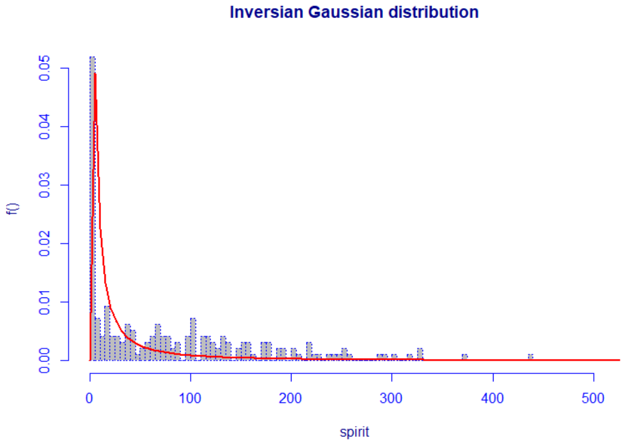
Based on the computation, we can affirm that the skewness value (1.288106) is above one, so it is right skewed, and it's also leptokurtic, by the kurtosis value (4.423172).
Next step, data fitting.

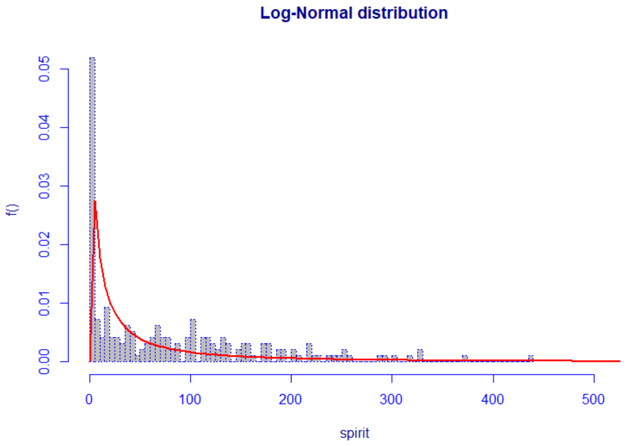


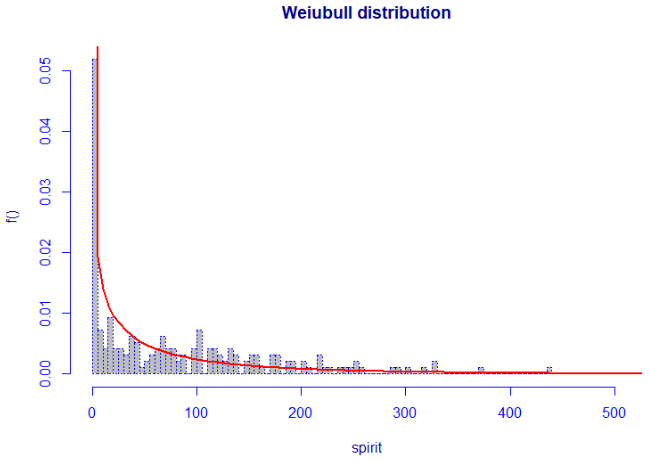
```
> spirit<-drinks$spirit_servings
> fit.EXP <- histDist(spirit, family=EXP, nbins=108, main="Exponential distribution")
> fit.GA <- histDist(spirit, family=GA, nbins=108, main="Gamma distribution")
> fit.IG <- histDist(spirit, family=IG, nbins=108, main="Inversian Gaussian distribution")
> fit.LOGNO <- histDist(spirit, family= LOGNO, nbins=108, main="Log-Normal distribution")
> fit.WEI <- histDist(spirit, family= WEI, nbins=108, main="Weiubull distribution")
```









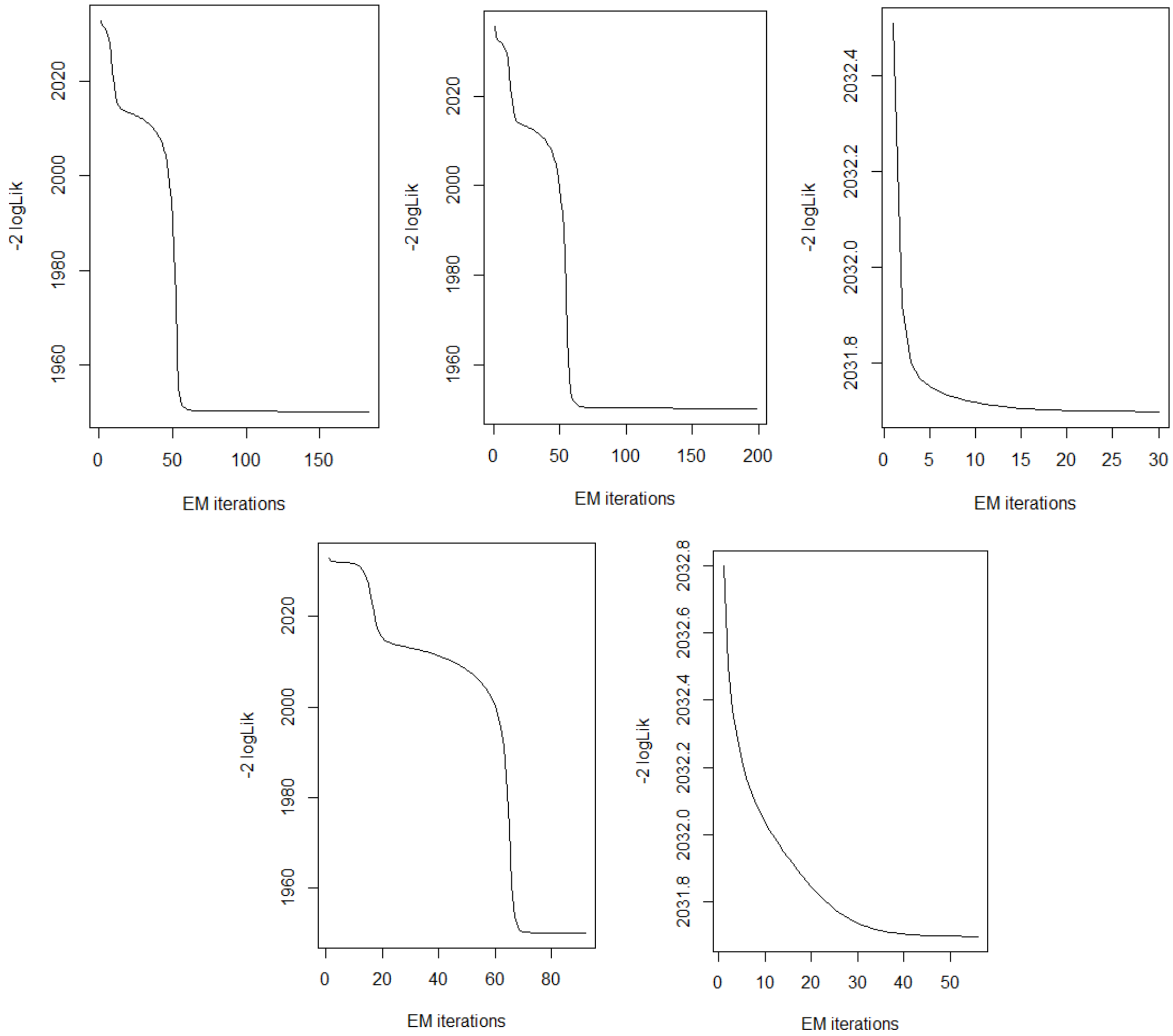


```
> data.frame(row.names=c("Exponential", "Gamma","Inversian Gaussian","Log-Normal","Weiubull"),
+           LogLikelihood=c(logLik(fit.EXP), logLik(fit.GA),logLik(fit.IG),logLik(fit.LOGNO),logLik(fit.WEI)),
+           AIC=c(AIC(fit.EXP),AIC(fit.GA),AIC(fit.IG),AIC(fit.LOGNO),AIC(fit.WEI)),
+           BIC=c(fit.EXP$sbc, fit.GA$sbc, fit.IG$sbc, fit.LOGNO$sbc, fit.WEI$sbc))
                   LogLikelihood      AIC      BIC
Exponential             -1041.536 2085.071 2088.334
Gamma                   -1015.848 2035.697 2042.222
Inversian Gaussian      -1066.698 2137.397 2143.922
Log-Normal              -1035.074 2074.148 2080.674
Weiubull                -1019.623 2043.246 2049.772
```

As suggested from the computation, the best model that fit better the Spirit_servings variable is the Gamma distribution.

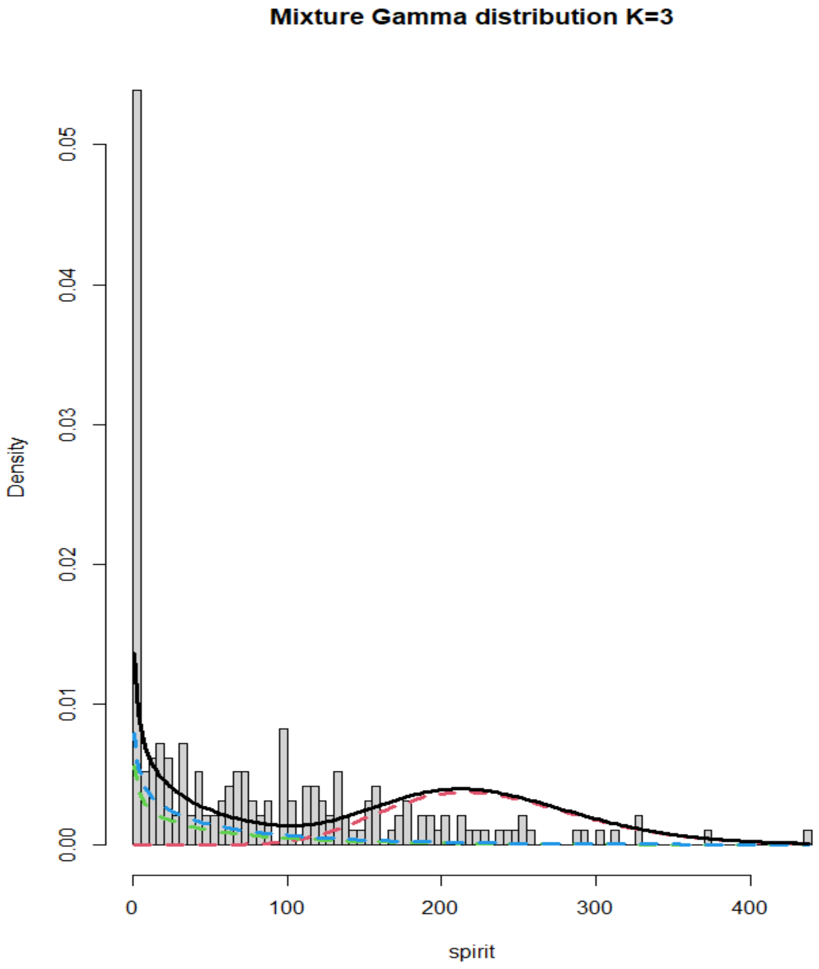
Now we'll try a mixture of 3 Gamma distribution.

```
> spirit<-drinks$spirit_servings
> mix.GA<-gamlssMXfits(n=5,spirit~1,family=GA,K=3,data=NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
```

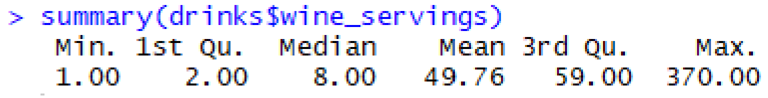


```
> mix.GA$aic
[1] 1966.239
> mix.GA$sbc
[1] 1992.341
> mix.GA$prob
[1] 0.5824842 0.1723920 0.2451238
```

```
> hist(spirit,breaks =108,freq=FALSE,main="Mixture Gamma distri
bution K=3")
> lines(seq(min(spirit),max(spirit),length=length(spirit)),mix.
GA[["prob"]][1]*dGA(seq(min(spirit),max(spirit),length=length(s
pirit)),mu=mu.hat1,sigma=sigma.hat1),lty=2,lwd=3,col=2)
> lines(seq(min(spirit),max(spirit),length=length(spirit)),mix.
GA[["prob"]][2]*dGA(seq(min(spirit),max(spirit),length=length(s
pirit)),mu=mu.hat2,sigma=sigma.hat2),lty=2,lwd=3,col=3)
> lines(seq(min(spirit),max(spirit),length=length(spirit)),mix.
GA[["prob"]][3]*dGA(seq(min(spirit),max(spirit),length=length(s
pirit)),mu=mu.hat3,sigma=sigma.hat3),lty=2,lwd=3,col=4)
> lines(seq(min(spirit),max(spirit),length=length(spirit)),mix.
GA[["prob"]][1]*dGA(seq(min(spirit),max(spirit),length=length(s
pirit)),mu=mu.hat1,sigma=sigma.hat1)+
+     + mix.GA[["prob"]][2]*dGA(seq(min(spirit),max(spirit),leng
th=length(spirit)),mu=mu.hat2,sigma=sigma.hat2)+
+     + mix.GA[["prob"]][3]*dGA(seq(min(spirit),max(spirit),leng
th=length(spirit)),mu=mu.hat3,sigma=sigma.hat3),lty=1,lwd=3,col
=1)
```

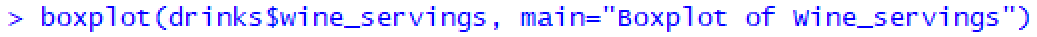


4.Wine_servings: indicates the average wine serving per person. It's a continuous numerical variable defined in the interval [1;370].

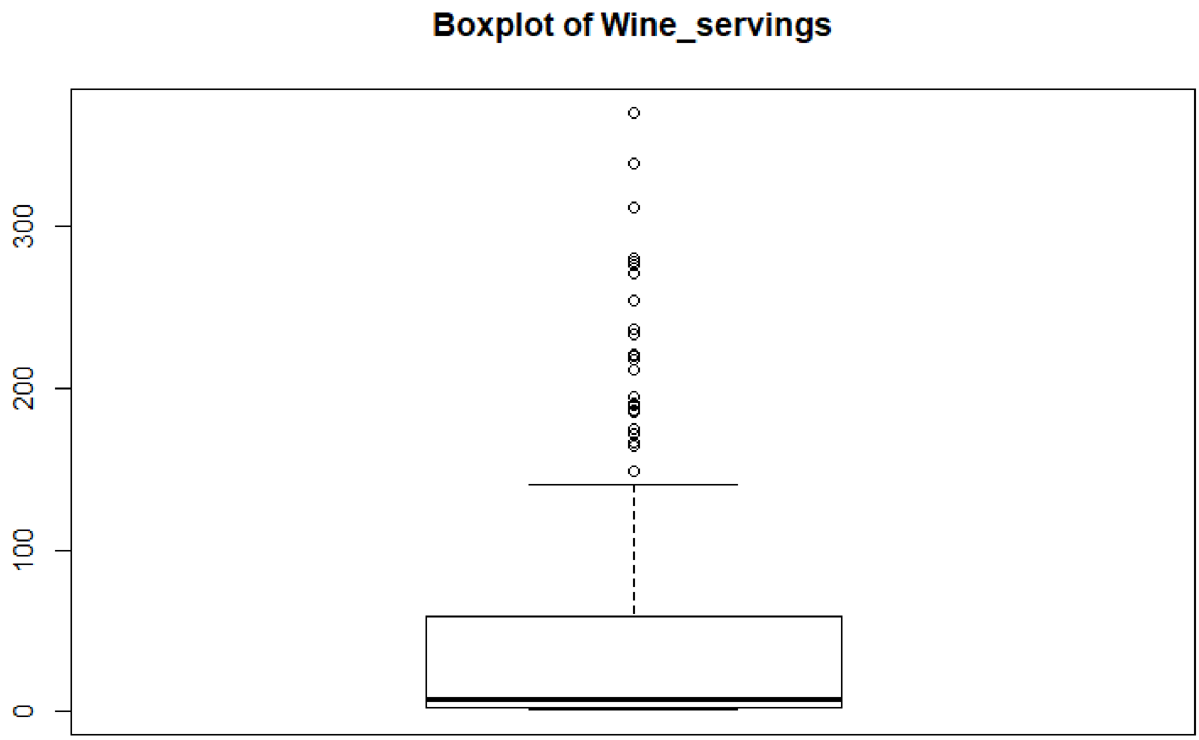


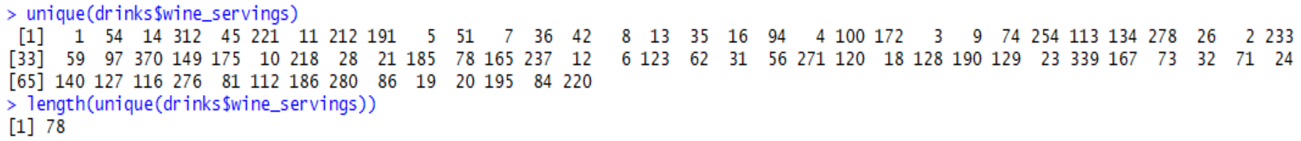
```
> summary(drinks$wine_servings)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   1.00    2.00    8.00   49.76   59.00  370.00 
```

The minimum average wine serving per person is 1, and the maximum is 370. The 25% of average wine serving per person is at least 2, the 50% of average wine serving per person is at least 8, and the 75% of average wine serving per person is at least 59.



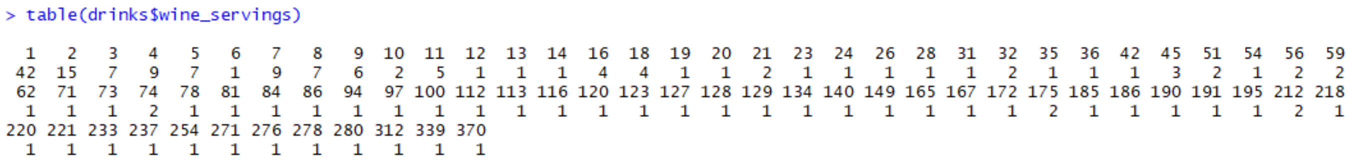
```
> boxplot(drinks$wine_servings, main="Boxplot of Wine_servings")
```





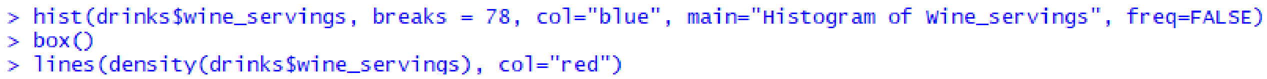
```
> unique(drinks$wine_servings)
 [1]   1  54  14 312  45 221  11 212 191   5  51   7  36  42   8  13  35  16  94   4 100 172   3   9  74 254 113 134 278  26   2 233
[33]  59  97 370 149 175  10 218  28  21 185  78 165 237  12   6 123  62  31  56 271 120  18 128 190 129  23 339 167  73  32  71  24
[65] 140 127 116 276  81 112 186 280  86  19  20 195  84 220
> length(unique(drinks$wine_servings))
[1] 78
```

The Wine_servings variable can assume 78 different values.

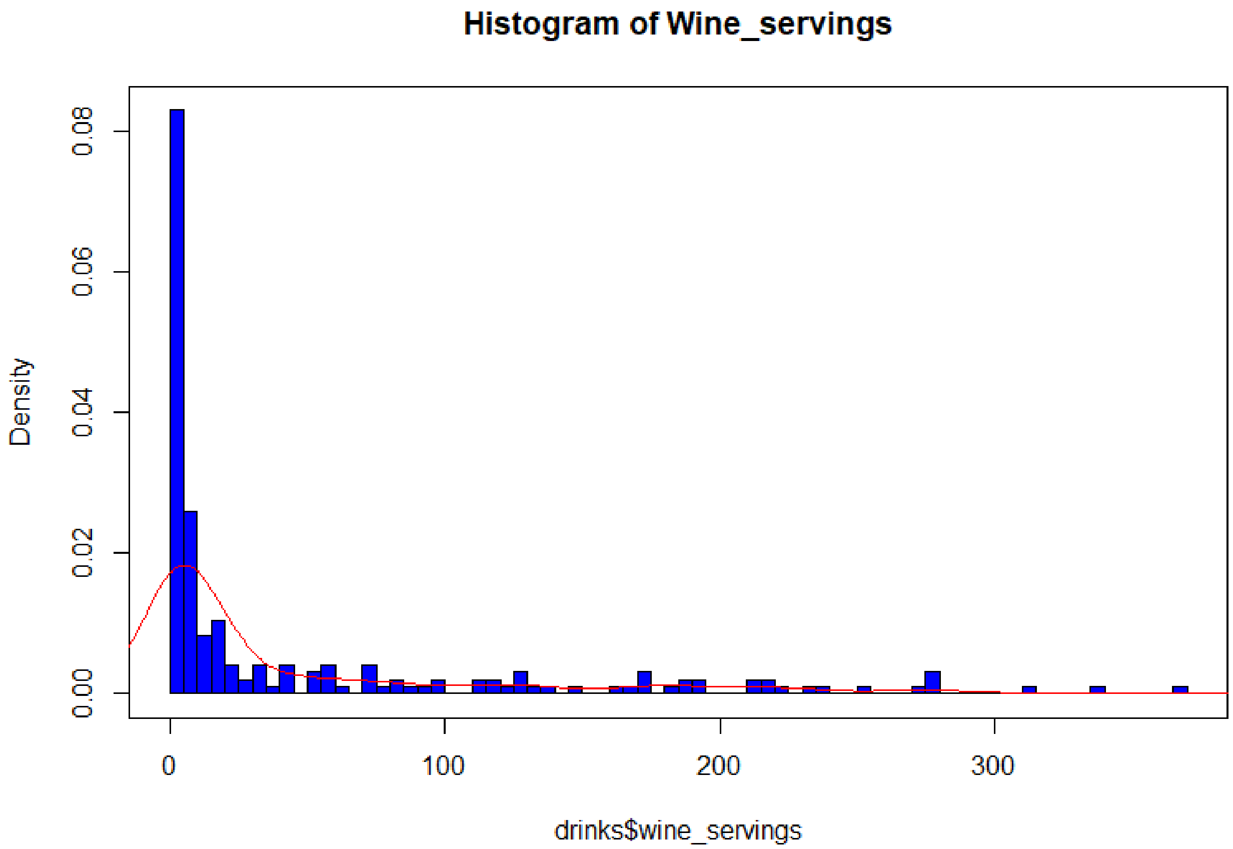


```
> table(drinks$wine_servings)

  1   2   3   4   5   6   7   8   9  10  11  12  13  14  16  18  19  20  21  23  24  26  28  31  32  35  36  42  45  51  54  56  59 
 42  15   7   9   7   1   9   7   6   2   5   1   1   1   4   4   1   1   2   1   1   1   1   1   2   1   1   1   3   2   1   2   2 
 62  71  73  74  78  81  84  86  94  97 100 112 113 116 120 123 127 128 129 134 140 149 165 167 172 175 185 186 190 191 195 212 218 
  1   1   1   2   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   2   1   1   1   1   1   2   1 
220 221 233 237 254 271 276 278 280 312 339 370 
  1   1   1   1   1   1   1   1   1   1   1   1 
```

We can plot a histogram for this variable, and then compute the skewness and the kurtosis.



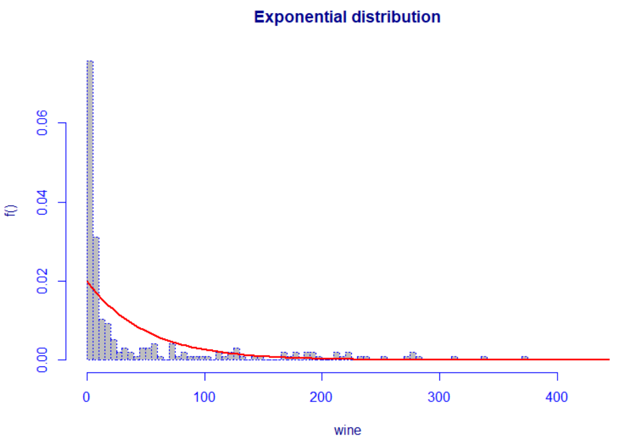
```
> hist(drinks$wine_servings, breaks = 78, col="blue", main="Histogram of Wine_servings", freq=FALSE)
> box()
> lines(density(drinks$wine_servings), col="red")
```

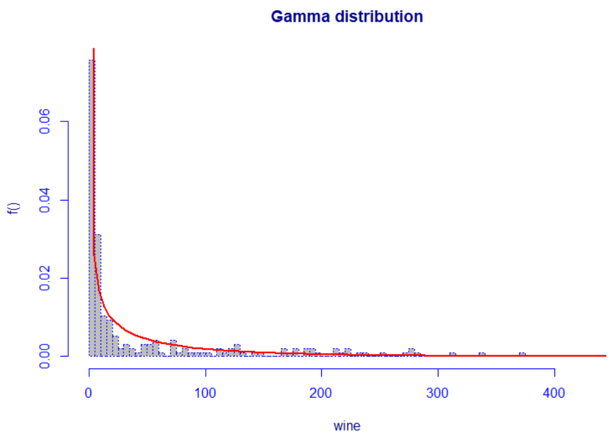


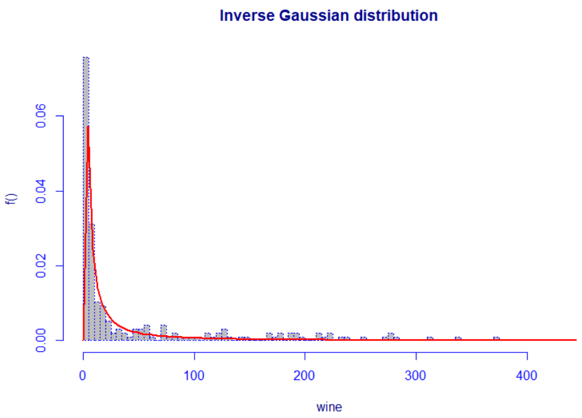
```
> skewness(drinks$wine_servings)
[1] 1.900648
> kurtosis(drinks$wine_servings)
[1] 5.859856
```

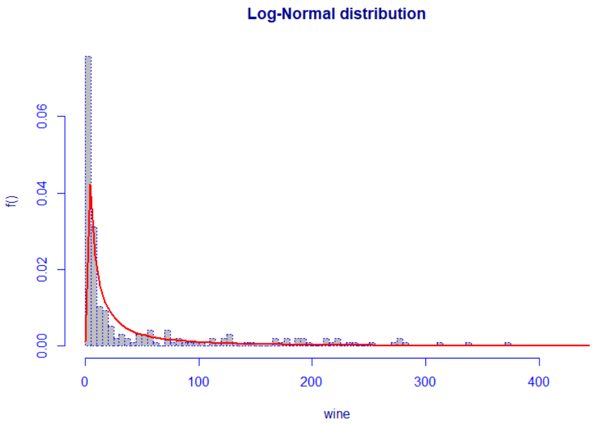
The result of the computation is, in term of skewness, the value (1.900648) is above 1, so the distribution is right skewed. It’s also leptokurtic due to the kurtosis value (5.859856).
To fit this distribution, we are going to compare some models.

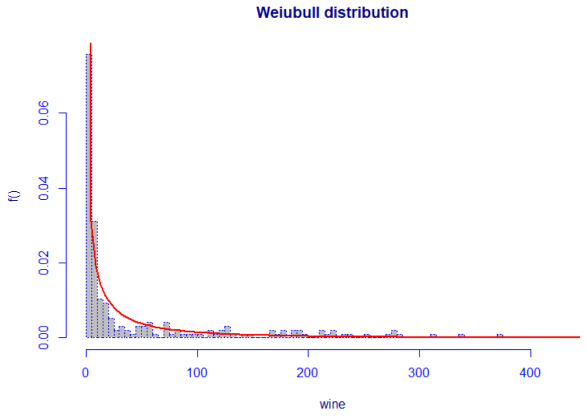
```
> wine<-drinks$wine_servings
> fit.EXP <- histDist(wine, family=EXP, nbins=78, main="Exponential distribution")
> fit.GA <- histDist(wine, family=GA, nbins=78, main="Gamma distribution")
> fit.IG <- histDist(wine, family=IG, nbins=78, main="Inverse Gaussian distribution")
> fit.LOGNO <- histDist(wine, family= LOGNO, nbins=78, main="Log-Normal distribution")
> fit.WEI <- histDist(wine, family= WEI, nbins=78, main="Weiubull distribution")
```



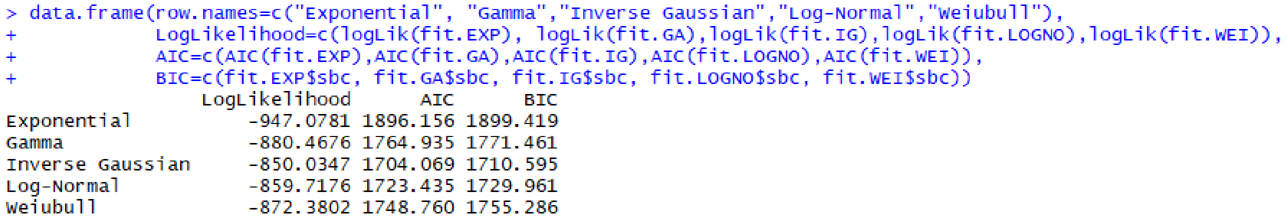








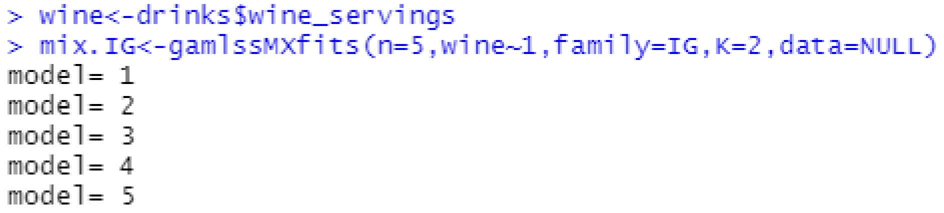
After that we are going to evaluate which models fits better the distribution.



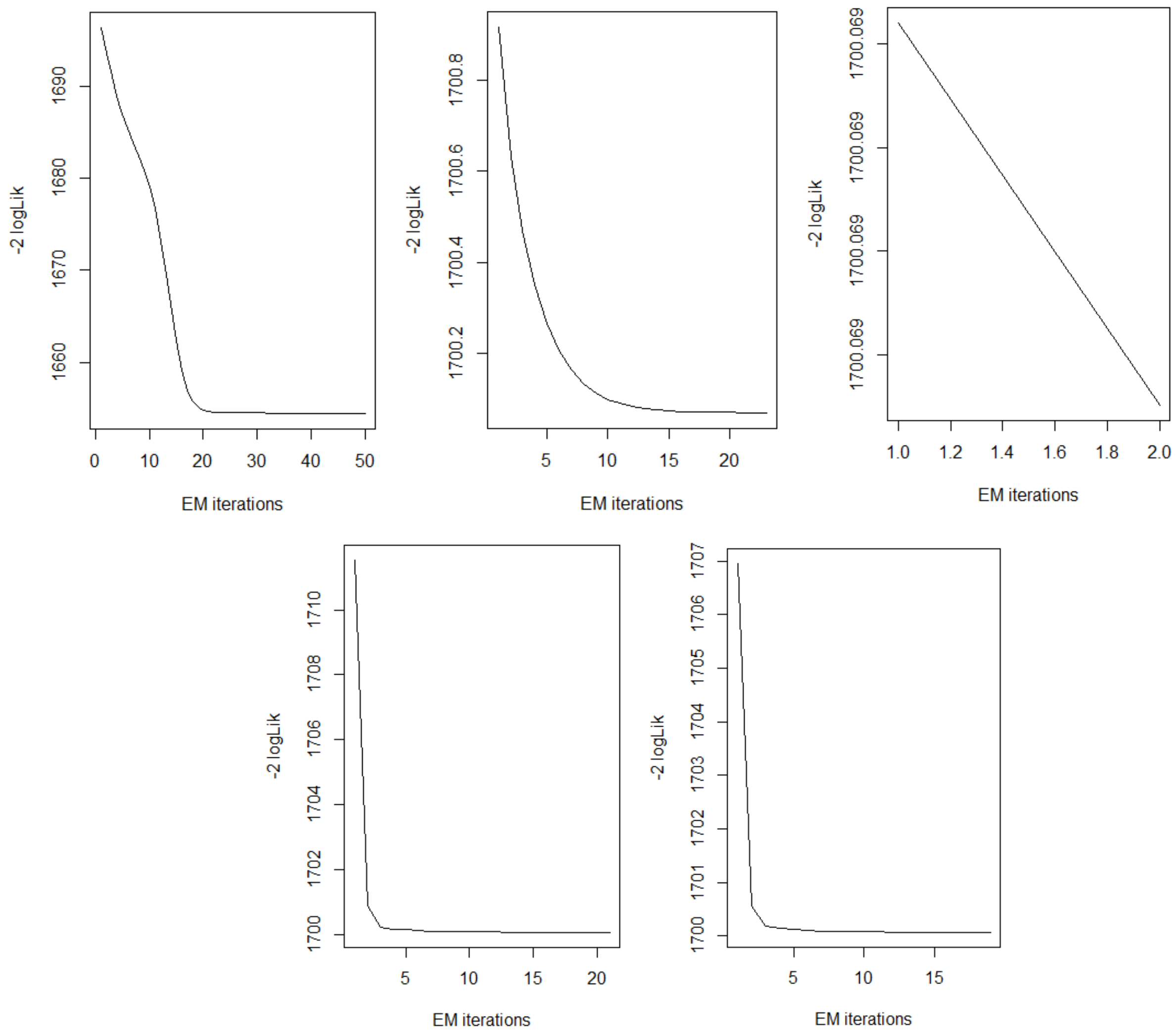
```
> data.frame(row.names=c("Exponential", "Gamma","Inverse Gaussian","Log-Normal","Weiubull"),
+           LogLikelihood=c(logLik(fit.EXP), logLik(fit.GA),logLik(fit.IG),logLik(fit.LOGNO),logLik(fit.WEI)),
+           AIC=c(AIC(fit.EXP),AIC(fit.GA),AIC(fit.IG),AIC(fit.LOGNO),AIC(fit.WEI)),
+           BIC=c(fit.EXP$sbc, fit.GA$sbc, fit.IG$sbc, fit.LOGNO$sbc, fit.WEI$sbc))
                 LogLikelihood      AIC      BIC
Exponential           -947.0781 1896.156 1899.419
Gamma                 -880.4676 1764.935 1771.461
Inverse Gaussian      -850.0347 1704.069 1710.595
Log-Normal            -859.7176 1723.435 1729.961
Weiubull              -872.3802 1748.760 1755.286
```

Based on the image above, the model that fits better the Wine_servings variable is the Inverse Gaussian.

Next, we’ll compute a mixture of 2 Inverse Gaussian distribution.

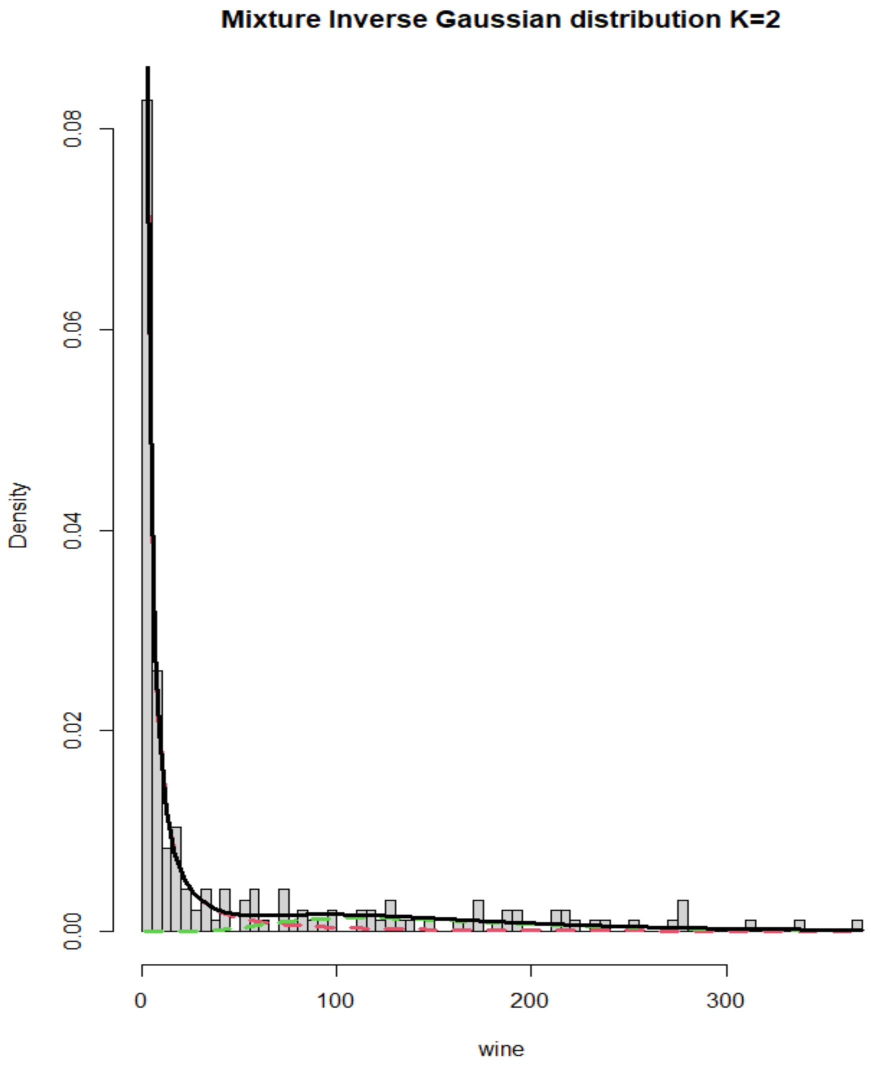


```
> wine<-drinks$wine_servings
> mix.IG<-gamlssMXfits(n=5,wine~1,family=IG,K=2,data=NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
```

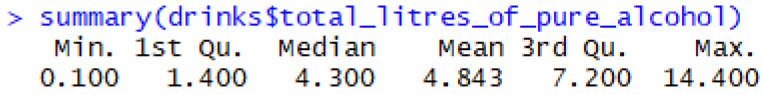


```
> mix.IG$aic
[1] 1664.495
> mix.IG$sbc
[1] 1680.809
> mix.IG$prob
[1] 0.7812411 0.2187589
```

```
> hist(wine,breaks =78,freq=FALSE,main="Mixture Inverse Gaussia
n distribution K=2")
> lines(seq(min(wine),max(wine),length=length(wine)),mix.IG[["p
rob"]][1]*dIG(seq(min(wine),max(wine),length=length(wine)),mu=m
u.hat1,sigma=sigma.hat1),lty=2,lwd=3,col=2)
> lines(seq(min(wine),max(wine),length=length(wine)),mix.IG[["p
rob"]][2]*dIG(seq(min(wine),max(wine),length=length(wine)),mu=m
u.hat2,sigma=sigma.hat2),lty=2,lwd=3,col=3)
> lines(seq(min(wine),max(wine),length=length(wine)),mix.IG[["p
rob"]][1]*dIG(seq(min(wine),max(wine),length=length(wine)),mu=m
u.hat1,sigma=sigma.hat1)+
+          + mix.IG[["prob"]][2]*dIG(seq(min(wine),max(wine),len
gth=length(wine)),mu=mu.hat2,sigma=sigma.hat2),lty=1,lwd=3,col=
1)
```

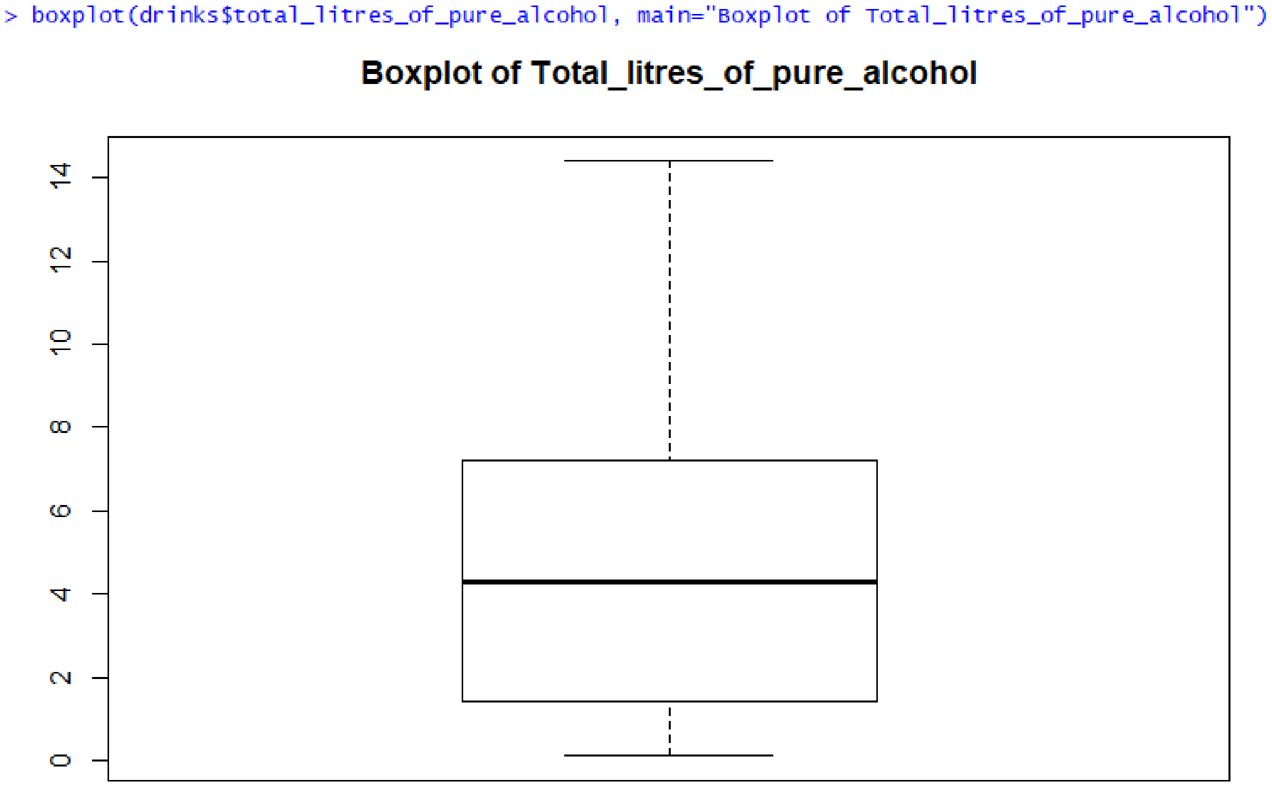

Mixture Inverse Gaussian distribution K=2
Density
0.00
0.02
0.04
0.06
0.08
0
100
200
300
wine

5.Total_litres_of_pure_alcohol: the last variable to be analysed, stand for total litres of pure alcohol served, is a continuous numerical variable. His values are included in the interval [0.100,14.400].
With the function summary() we can see some information about that.



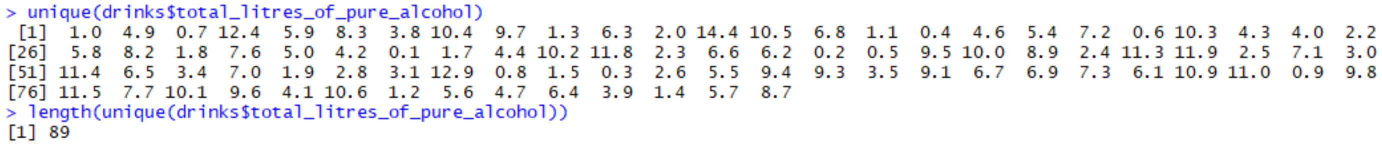
```
> summary(drinks$total_litres_of_pure_alcohol)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.100   1.400   4.300   4.843   7.200  14.400 
```

The minimum value of Total_litres_of_pure_alcohol is 0.100, and the maximum is 14.400. The 25% of Total_litres_of_pure_alcohol is at least 1.400; the 50% of Total_litres_of_pure_alcohol is at least 4.300; finally, the 75% of Total_litres_of_pure_alcohol is at least about 7.200.



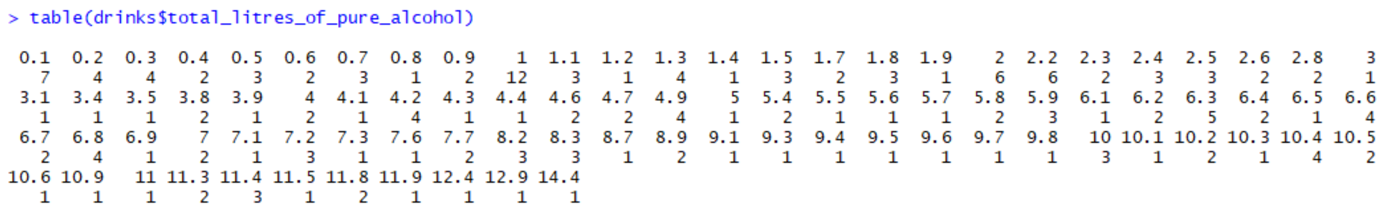
```
> boxplot(drinks$total_litres_of_pure_alcohol, main="Boxplot of Total_litres_of_pure_alcohol")
```

Now we can compute all the values that Total_litres_of_pure_alcohol assumes and the absolute frequencies of the variable.



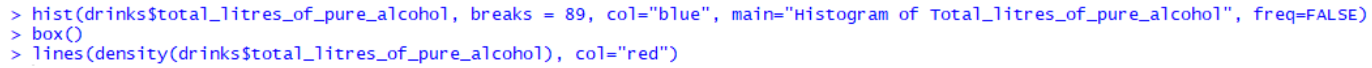
```
> unique(drinks$total_litres_of_pure_alcohol)
 [1]  1.0  4.9  0.7 12.4  5.9  8.3  3.8 10.4  9.7  1.3  6.3  2.0 14.4 10.5  6.8  1.1  0.4  4.6  5.4  7.2  0.6 10.3  4.3  4.0  2.2
[26]  5.8  8.2  1.8  7.6  5.0  4.2  0.1  1.7  4.4 10.2 11.8  2.3  6.6  6.2  0.2  0.5  9.5 10.0  8.9  2.4 11.3 11.9  2.5  7.1  3.0
[51] 11.4  6.5  3.4  7.0  1.9  2.8  3.1 12.9  0.8  1.5  0.3  2.6  5.5  9.4  9.3  3.5  9.1  6.7  6.9  7.3  6.1 10.9 11.0  0.9  9.8
[76] 11.5  7.7 10.1  9.6  4.1 10.6  1.2  5.6  4.7  6.4  3.9  1.4  5.7  8.7
> length(unique(drinks$total_litres_of_pure_alcohol))
[1] 89
```

As we can see from the image, the variable assumes 89 different values.
Now with the function table() we are going to see the frequency.

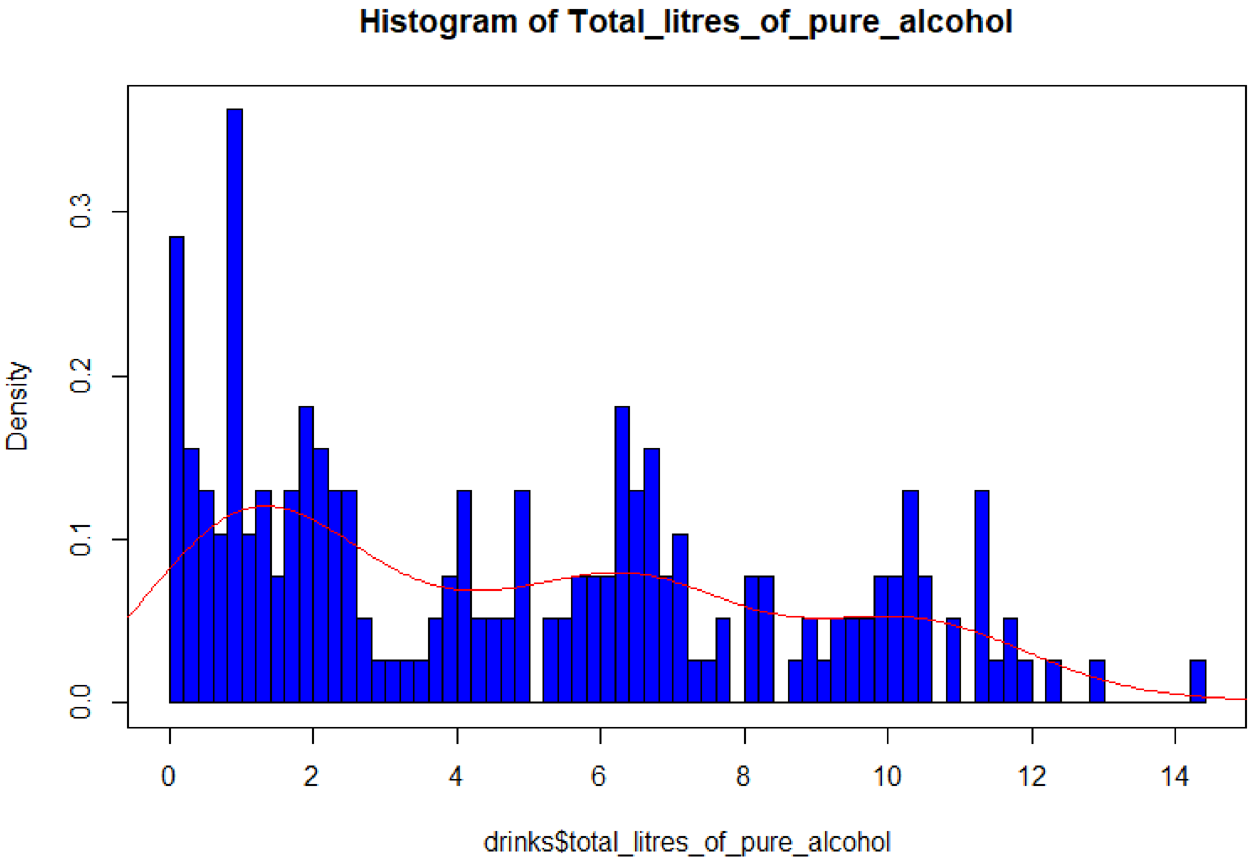


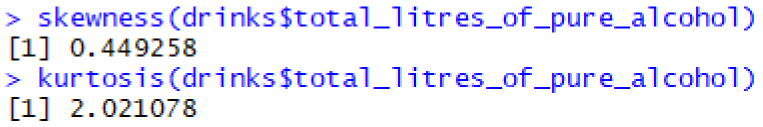
```
> table(drinks$total_litres_of_pure_alcohol)

 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  1.7  1.8  1.9    2  2.2  2.3  2.4  2.5  2.6  2.8    3
   7    4    4    2    3    2    3    1    2   12    3    1    4    1    3    2    3    1    6    6    2    3    3    2    2    1
 3.1  3.4  3.5  3.8  3.9    4  4.1  4.2  4.3  4.4  4.6  4.7  4.9    5  5.4  5.5  5.6  5.7  5.8  5.9  6.1  6.2  6.3  6.4  6.5  6.6
   1    1    1    2    1    2    1    4    1    1    2    2    4    1    2    1    1    1    2    3    1    2    5    2    1    4
 6.7  6.8  6.9    7  7.1  7.2  7.3  7.6  7.7  8.2  8.3  8.7  8.9  9.1  9.3  9.4  9.5  9.6  9.7  9.8   10 10.1 10.2 10.3 10.4 10.5
   2    4    1    2    1    3    1    1    2    3    3    1    2    1    1    1    1    1    1    1    3    1    2    1    4    2
10.6 10.9   11 11.3 11.4 11.5 11.8 11.9 12.4 12.9 14.4
   1    1    1    2    3    1    2    1    1    1    1
```

Computation of the histogram for the variable Total_litres_of_pure_alcohol.



```
> hist(drinks$total_litres_of_pure_alcohol, breaks = 89, col="blue", main="Histogram of Total_litres_of_pure_alcohol", freq=FALSE)
> box()
> lines(density(drinks$total_litres_of_pure_alcohol), col="red")
```



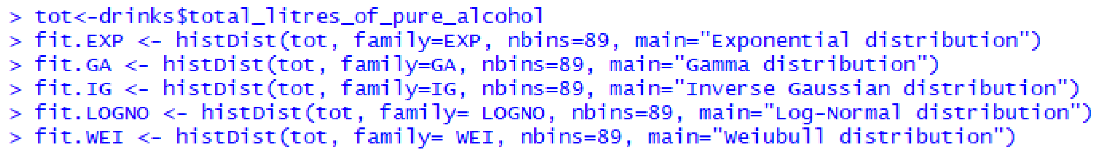


```
> skewness(drinks$total_litres_of_pure_alcohol)
[1] 0.449258
> kurtosis(drinks$total_litres_of_pure_alcohol)
[1] 2.021078
```

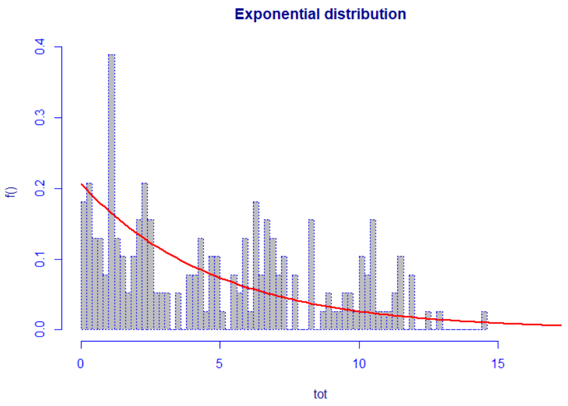
As the results say, the skewness value (0.449258) is smaller than 1, so we can attest that the distribution is fairly symmetric.

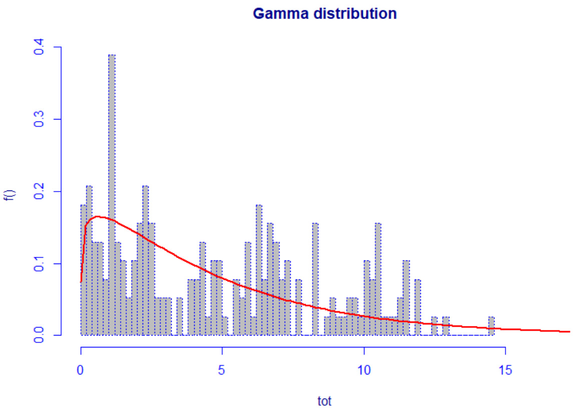
The kurtosis value (2.021078) is above 0, so the distribution is leptokurtic.

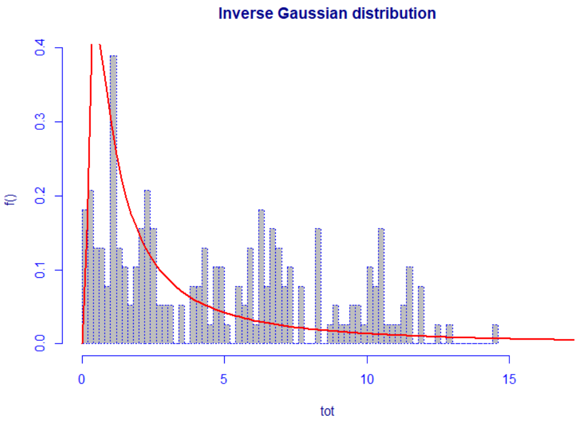
After that, we can go through the data fitting of Total_litres_of_pure_Alcohol.

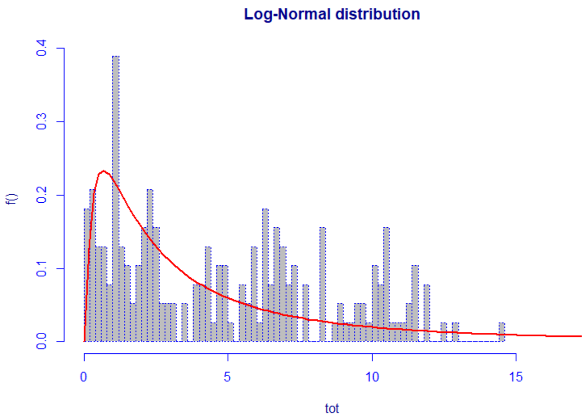


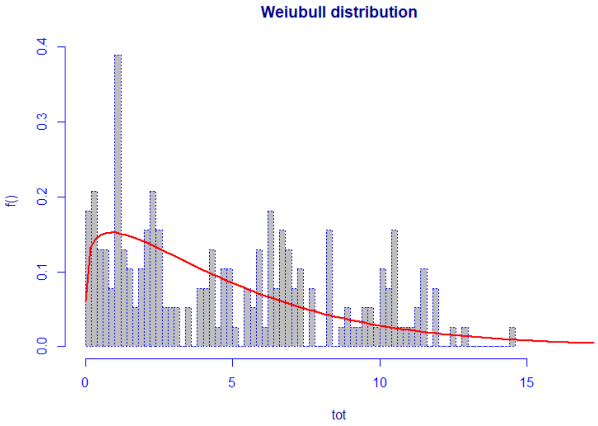
```
> tot<-drinks$total_litres_of_pure_alcohol
> fit.EXP <- histDist(tot, family=EXP, nbins=89, main="Exponential distribution")
> fit.GA <- histDist(tot, family=GA, nbins=89, main="Gamma distribution")
> fit.IG <- histDist(tot, family=IG, nbins=89, main="Inverse Gaussian distribution")
> fit.LOGNO <- histDist(tot, family= LOGNO, nbins=89, main="Log-Normal distribution")
> fit.WEI <- histDist(tot, family= WEI, nbins=89, main="Weiubull distribution")
```


Exponential distribution


Gamma distribution


Inverse Gaussian distribution


Log-Normal distribution


Weiubull distribution

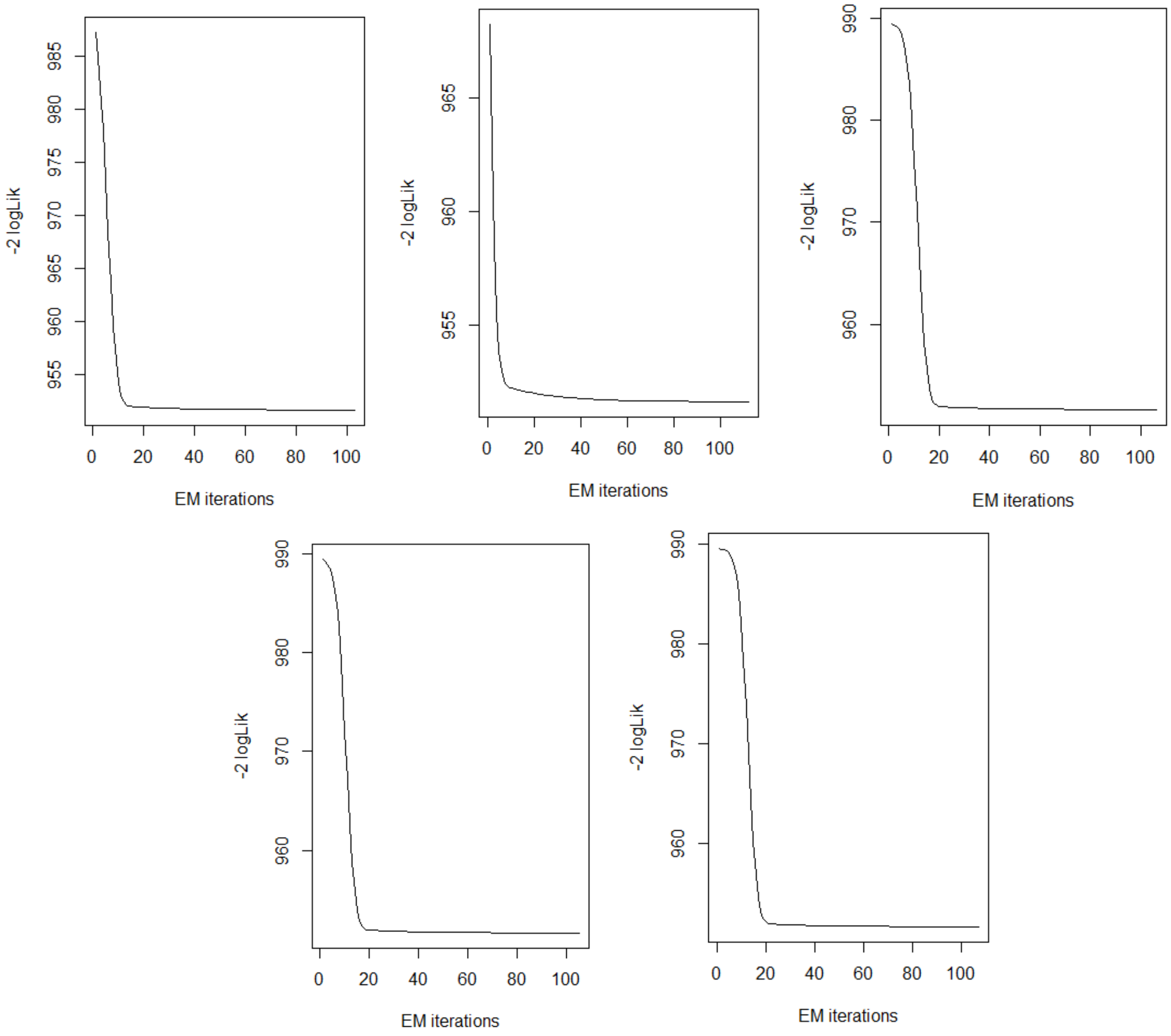
Next step is the evaluation of the model that fits better the distribution.

```
> data.frame(row.names=c("Exponential", "Gamma","Inverse Gaussian","Log-Normal","Weiubull"),
+           LogLikelihood=c(logLik(fit.EXP), logLik(fit.GA),logLik(fit.IG),logLik(fit.LOGNO),logLik(fit.WEI)),
+           AIC=c(AIC(fit.EXP),AIC(fit.GA),AIC(fit.IG),AIC(fit.LOGNO),AIC(fit.WEI)),
+           BIC=c(fit.EXP$sbc, fit.GA$sbc, fit.IG$sbc, fit.LOGNO$sbc, fit.WEI$sbc))
                 LogLikelihood       AIC      BIC
Exponential          -497.4643  996.9287 1000.191
Gamma                -496.3525  996.7051 1003.230
Inverse Gaussian     -554.6597 1113.3193 1119.845
Log-Normal           -521.0677 1046.1355 1052.661
Weiubull             -494.7744  993.5489 1000.074
```

For the variable Total_litres_of_pure_alcohol, the best model, according to the values above, is the Weibull distribution.

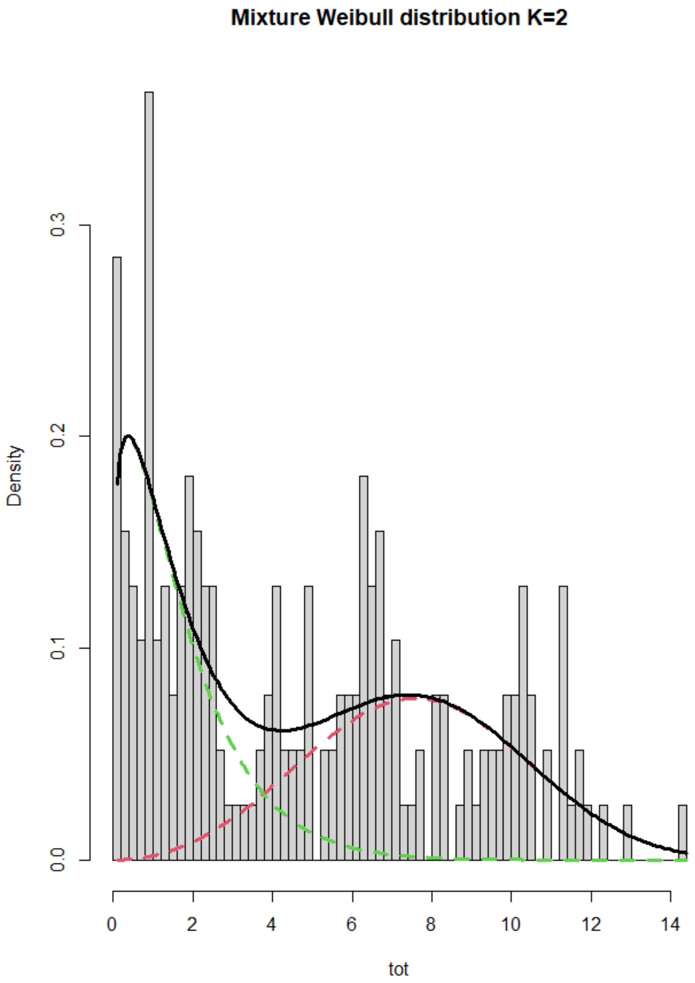
Finally, we'll compute the mixture of 2 Weibull distribution.

```
> tot<-drinks$total_litres_of_pure_alcohol
> mix.WEI<-gamlssMXfits(n=5,tot~1,family=WEI,K=2,data=NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
```



```
> mix.WEI$aic
[1] 961.6383
> mix.WEI$sbc
[1] 977.9518
> mix.WEI$prob
[1] 0.5302933 0.4697067
```

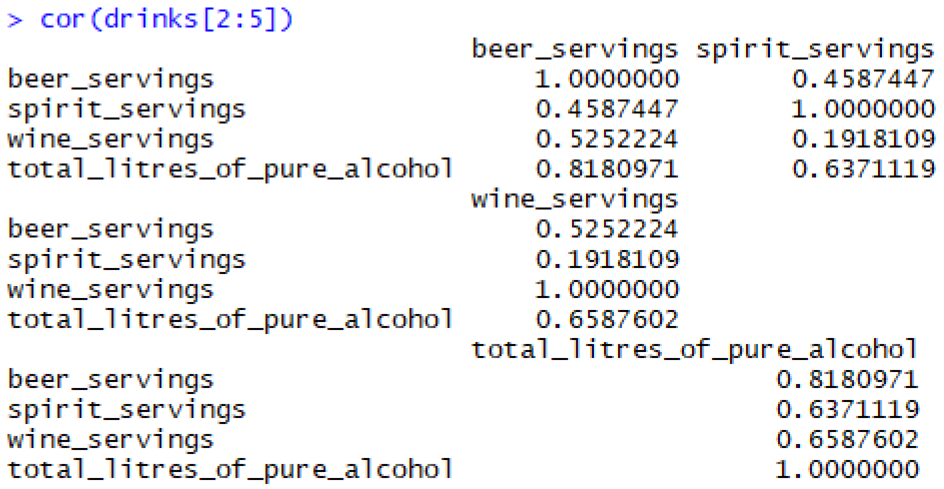
```
> hist(tot,breaks =89,freq=FALSE,main="Mixture Weibull distribu
tion K=2")
> lines(seq(min(tot),max(tot),length=length(tot)),mix.WEI[["pro
b"]][1]*dWEI(seq(min(tot),max(tot),length=length(tot)),mu=mu.ha
t1,sigma=sigma.hat1),lty=2,lwd=3,col=2)
> lines(seq(min(tot),max(tot),length=length(tot)),mix.WEI[["pro
b"]][2]*dWEI(seq(min(tot),max(tot),length=length(tot)),mu=mu.ha
t2,sigma=sigma.hat2),lty=2,lwd=3,col=3)
> lines(seq(min(tot),max(tot),length=length(tot)),mix.WEI[["pro
b"]][1]*dWEI(seq(min(tot),max(tot),length=length(tot)),mu=mu.ha
t1,sigma=sigma.hat1)+
+          + mix.WEI[["prob"]][2]*dWEI(seq(min(tot),max(tot),len
gth=length(tot)),mu=mu.hat2,sigma=sigma.hat2),lty=1,lwd=3,col=
1)
```



PRINCIPAL COMPONENT ANALYSIS

This chapter aims to analyse the dataset by take in consideration the PCA, Principal Component Analysis. The PCA is an instrument that allows us to summarize the information in the dataset. In other words, the PCA method reduces the dimensionality of multivariate data with minimal loss of information. An important aspect is that each variable, from a geometrical point of view, represent a different dimension.

Take in consideration our dataset, there are 4 numerical variable, and 1 categorical, that is the first one, Country. We are going to exclude this variable from the correlation matrix cor() because is categorical with the following code.

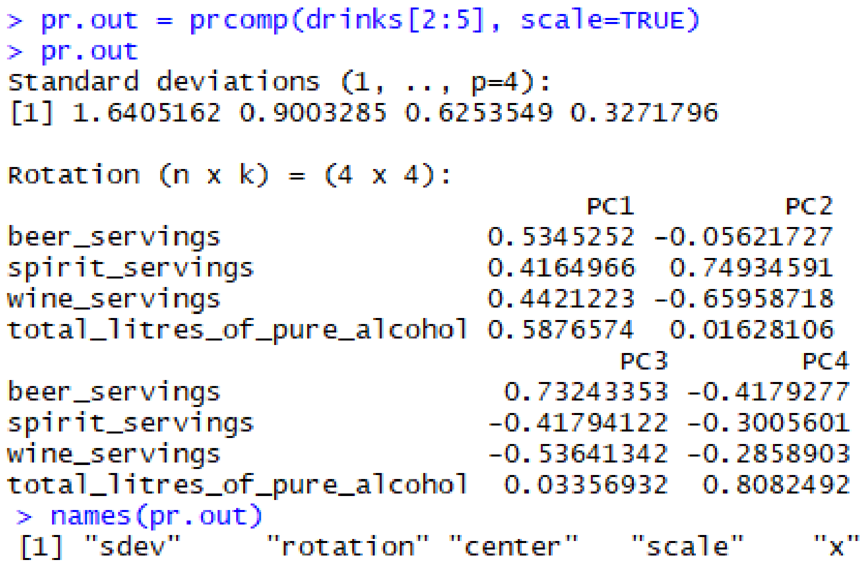


```
> cor(drinks[2:5])
                             beer_servings spirit_servings
beer_servings                    1.0000000       0.4587447
spirit_servings                  0.4587447       1.0000000
wine_servings                    0.5252224       0.1918109
total_litres_of_pure_alcohol     0.8180971       0.6371119
                             wine_servings
beer_servings                    0.5252224
spirit_servings                  0.1918109
wine_servings                    1.0000000
total_litres_of_pure_alcohol     0.6587602
                             total_litres_of_pure_alcohol
beer_servings                                   0.8180971
spirit_servings                                 0.6371119
wine_servings                                   0.6587602
total_litres_of_pure_alcohol                    1.0000000
```

We can see that there is a positive correlation between spirit_servings (0.4587447) and wine_servings (0.5252224).

The next step is to find the PCs (Principal components) with the use of the command prcomp() that use singular value decomposition, which examines the covariance / correlation between individuals and centres the variable to have mean zero.

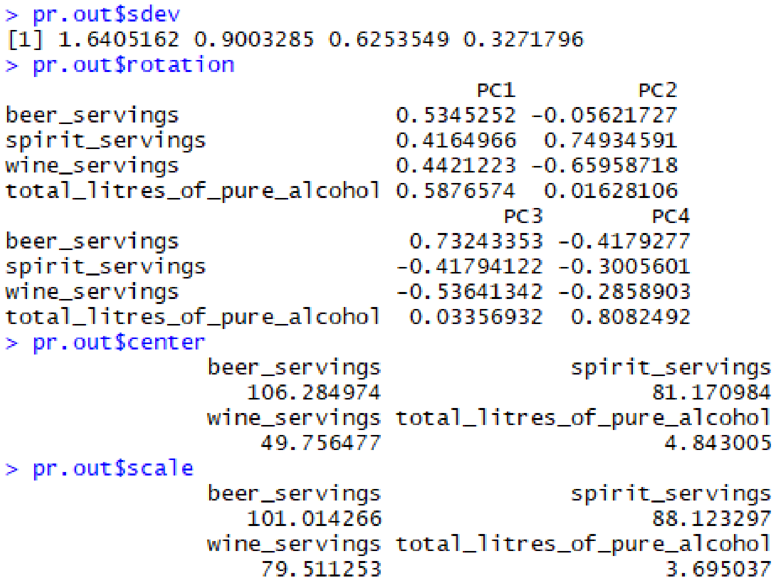
As we can see from the code below, the option scale=TRUE allows us to scale the variable to have standard deviation one.



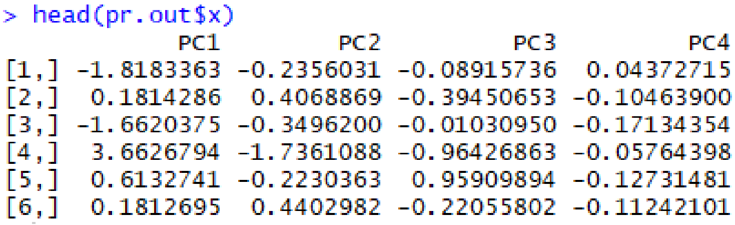
```
> pr.out = prcomp(drinks[2:5], scale=TRUE)
> pr.out
Standard deviations (1, .., p=4):
[1] 1.6405162 0.9003285 0.6253549 0.3271796

Rotation (n x k) = (4 x 4):
                                   PC1         PC2
beer_servings                0.5345252 -0.05621727
spirit_servings              0.4164966  0.74934591
wine_servings                0.4421223 -0.65958718
total_litres_of_pure_alcohol 0.5876574  0.01628106
                                     PC3        PC4
beer_servings                 0.73243353 -0.4179277
spirit_servings              -0.41794122 -0.3005601
wine_servings                -0.53641342 -0.2858903
total_litres_of_pure_alcohol  0.03356932  0.8082492
 > names(pr.out)
 [1] "sdev"      "rotation" "center"    "scale"     "x"
```

So, the prcomp will return a list that have, as the computation above suggests, some components:

- "sdev" indicates the standard deviation of the PCs
- "rotation" will give a matrix of variable loadings and the columns are the eigenvectors
- "center" indicates the variable means
- "scale" indicates the variable standard deviation
- "x" indicates the coordinates of the individuals (observations) on the principal components, so the score of the PC.

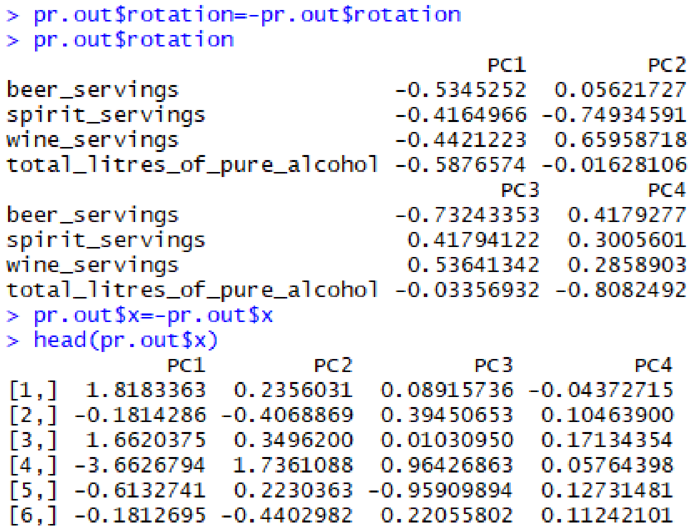


```
> pr.out$sdev
[1] 1.6405162 0.9003285 0.6253549 0.3271796
> pr.out$rotation
                                PC1         PC2
beer_servings             0.5345252 -0.05621727
spirit_servings           0.4164966  0.74934591
wine_servings             0.4421223 -0.65958718
total_litres_of_pure_alcohol 0.5876574  0.01628106
                                 PC3        PC4
beer_servings              0.73243353 -0.4179277
spirit_servings           -0.41794122 -0.3005601
wine_servings             -0.53641342 -0.2858903
total_litres_of_pure_alcohol  0.03356932  0.8082492
> pr.out$center
               beer_servings              spirit_servings
                  106.284974                    81.170984
               wine_servings total_litres_of_pure_alcohol
                   49.756477                     4.843005
> pr.out$scale
               beer_servings              spirit_servings
                  101.014266                    88.123297
               wine_servings total_litres_of_pure_alcohol
                   79.511253                     3.695037
```



```
> head(pr.out$x)
            PC1        PC2         PC3         PC4
[1,] -1.8183363 -0.2356031 -0.08915736  0.04372715
[2,]  0.1814286  0.4068869 -0.39450653 -0.10463900
[3,] -1.6620375 -0.3496200 -0.01030950 -0.17134354
[4,]  3.6626794 -1.7361088 -0.96426863 -0.05764398
[5,]  0.6132741 -0.2230363  0.95909894 -0.12731481
[6,]  0.1812695  0.4402982 -0.22055802 -0.11242101
```

For a better interpretation, in the output of "rotation" and "x" we can do a sign flip.

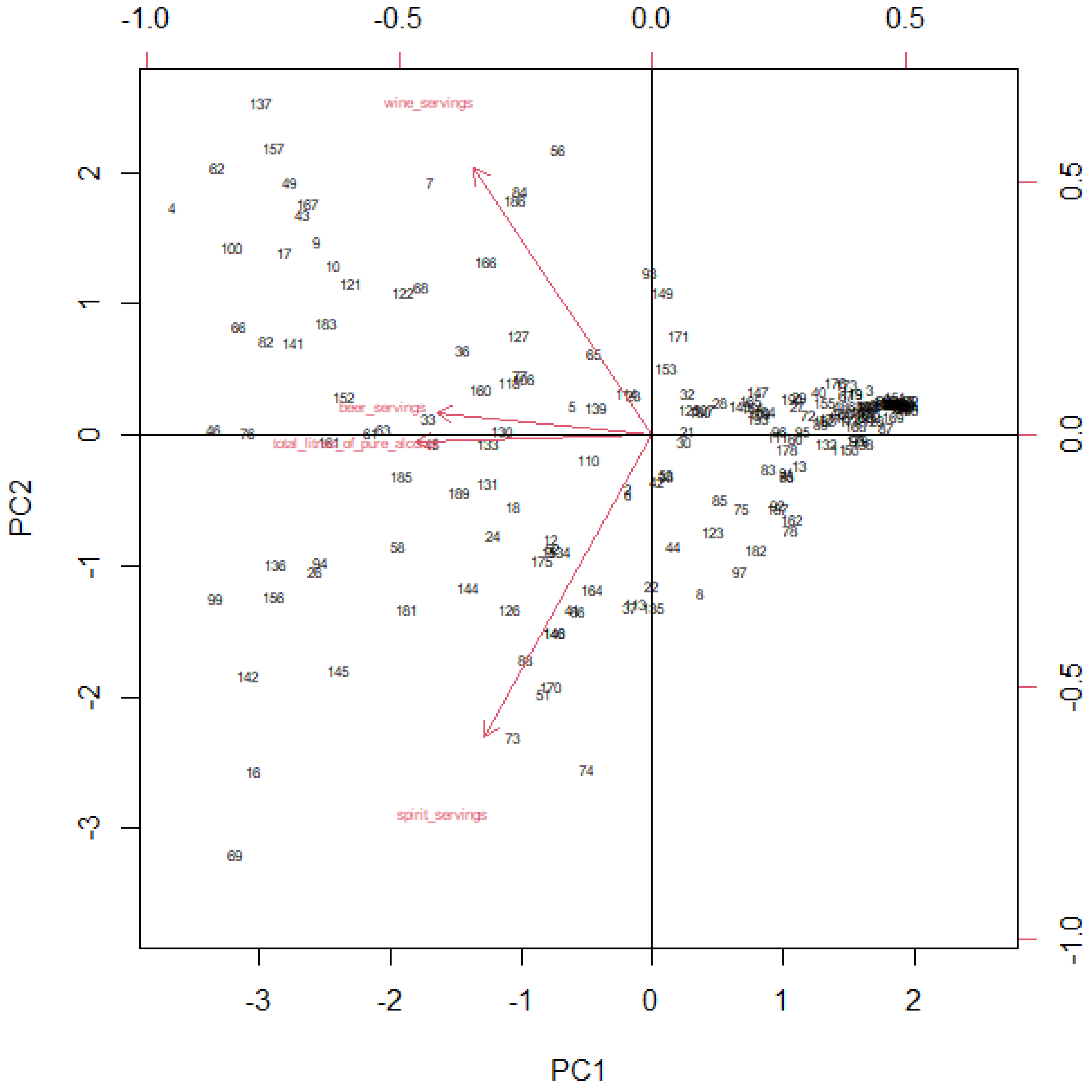


```
> pr.out$rotation=-pr.out$rotation
> pr.out$rotation
                                 PC1         PC2
beer_servings             -0.5345252  0.05621727
spirit_servings           -0.4164966 -0.74934591
wine_servings             -0.4421223  0.65958718
total_litres_of_pure_alcohol -0.5876574 -0.01628106
                                  PC3        PC4
beer_servings             -0.73243353  0.4179277
spirit_servings            0.41794122  0.3005601
wine_servings              0.53641342  0.2858903
total_litres_of_pure_alcohol -0.03356932 -0.8082492
> pr.out$x=-pr.out$x
> head(pr.out$x)
            PC1        PC2         PC3         PC4
[1,]  1.8183363  0.2356031  0.08915736 -0.04372715
[2,] -0.1814286 -0.4068869  0.39450653  0.10463900
[3,]  1.6620375  0.3496200  0.01030950  0.17134354
[4,] -3.6626794  1.7361088  0.96426863  0.05764398
[5,] -0.6132741  0.2230363 -0.95909894  0.12731481
[6,] -0.1812695 -0.4402982  0.22055802  0.11242101
```

Biplot: it is an exploratory plot which aims to represent both the observations and variables of a matrix of multivariate data on the same plot. In this case, we will be able to visualize the scores and the original variable in the first two PCs'space.

```
> biplot(pr.out, scale=0, cex=0.5)
> abline(h=0)
> abline(v=0)
```

As the code above, the PCs will be the axes, the points are the computed scores, and the variables are the arrows.



To better interpretate the biplot, we can do some examples, like there is a positive correlation between the variable beer_servings and total_litres_of_pure_alcohol, because their arrows are almost near, so the angle is close to 0. From the theory study, we know that the points (PCs score) close to the origin has values of PCs close to the mean.

We can choose the number of PCs based on three important factors.

- **(Cumulative) The proportion of variance explained (PVE):**

```
> pve=pr.var/sum(pr.var)
> pve
[1] 0.67282334 0.20264786 0.09776718 0.02676162
```

As the definition of PVE attest, the first PC describes 67.3% of the variability, the second PC describes 20.2% of the variability, the third PC describes the 9.7% of the variability and the fourth one describes 2.6% of the variability.

```
> cumsum(pve)
[1] 0.6728233 0.8754712 0.9732384 1.0000000
```

So, according with the definition of the PVE, we must choose as many PCs as needed to explain at least the 80% of the total variance. In this case, the first two principal components explain the 87.5% of the variability.

- **Kaiser's rule:**

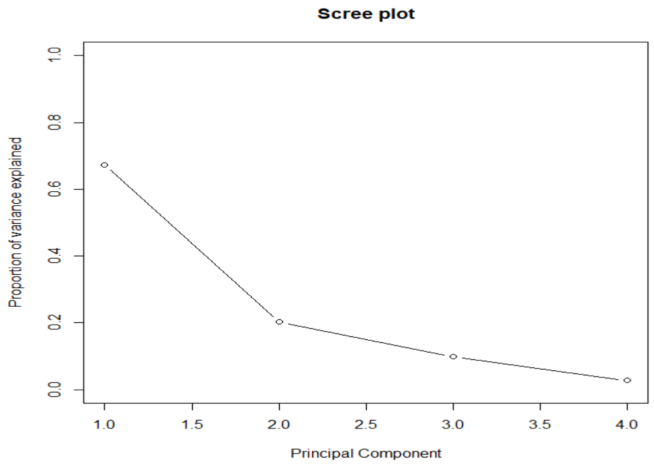
```
> pr.var=pr.out$sdev^2
> pr.var
[1] 2.6912934 0.8105914 0.3910687 0.1070465
```

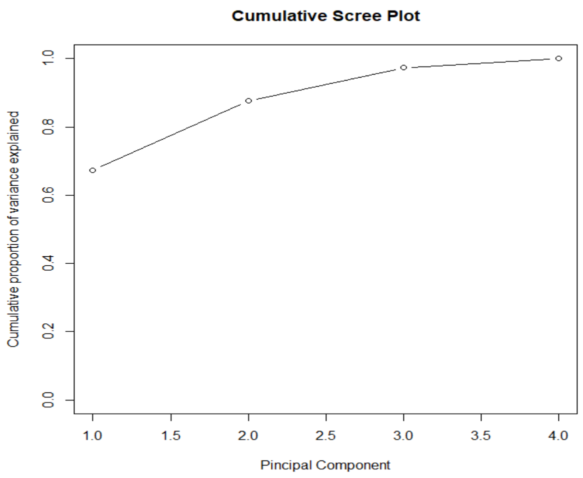
This rule suggests to retaining as many PCs as are the eigenvalues of R (variance) greater than 1.

- **Scree plot:**

This rule says that to determine the number of principal components we must look for the point in the plot that produce a jump.

```
> plot(pve,main="Scree plot", xlab="Principal Component",
+       ylab="Proportion of variance explained", ylim=c(0,1),
 type = 'b'
+ )

> plot(cumsum(pve), main="Cumulative Scree Plot", xlab="Pinc
ipal Component",
+       ylab = "Cumulative proportion of variance explained",
 ylim=c(0,1),type='b'
+       )
```





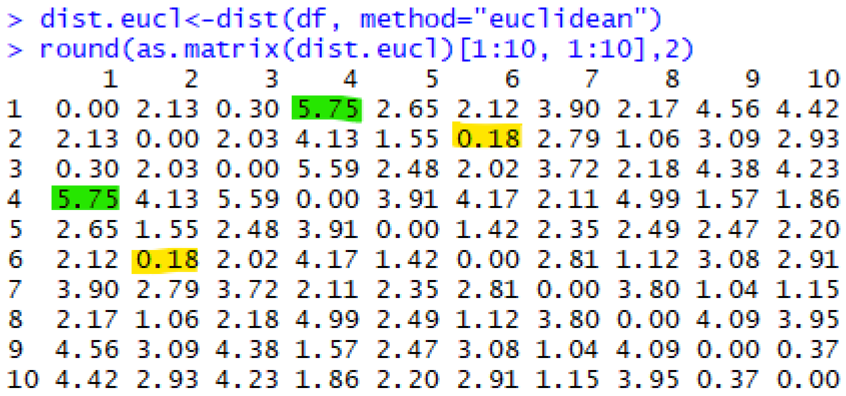
CLUSTER ANALYSIS

Cluster Analysis (CA), simply said clustering, is one of the most important statistical methods for discovering knowledge in multidimensional data. The goal of CA is to identify patterns (or groups, or clusters) of similar units within a data set.

There are different methods. The first step, in our case, is to calculate the distance between pair of units and build the distance matrix.

Important! We are analysing only a subset of data.

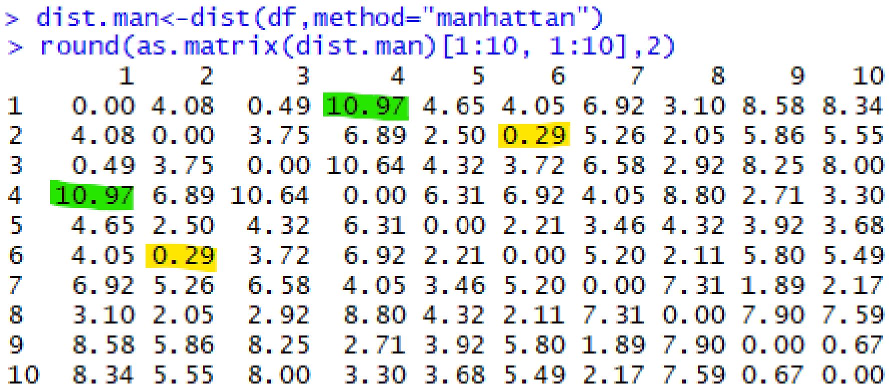
First distance to be computed is Euclidean.



```
> dist.eucl<-dist(df, method="euclidean")
> round(as.matrix(dist.eucl)[1:10, 1:10],2)
      1    2    3    4    5    6    7    8    9   10
1  0.00 2.13 0.30 5.75 2.65 2.12 3.90 2.17 4.56 4.42
2  2.13 0.00 2.03 4.13 1.55 0.18 2.79 1.06 3.09 2.93
3  0.30 2.03 0.00 5.59 2.48 2.02 3.72 2.18 4.38 4.23
4  5.75 4.13 5.59 0.00 3.91 4.17 2.11 4.99 1.57 1.86
5  2.65 1.55 2.48 3.91 0.00 1.42 2.35 2.49 2.47 2.20
6  2.12 0.18 2.02 4.17 1.42 0.00 2.81 1.12 3.08 2.91
7  3.90 2.79 3.72 2.11 2.35 2.81 0.00 3.80 1.04 1.15
8  2.17 1.06 2.18 4.99 2.49 1.12 3.80 0.00 4.09 3.95
9  4.56 3.09 4.38 1.57 2.47 3.08 1.04 4.09 0.00 0.37
10 4.42 2.93 4.23 1.86 2.20 2.91 1.15 3.95 0.37 0.00
```

So, the result, based on this matrix of Euclidean distance, suggests that the most distant observations are the 2 and 6, and the most similar observation are 1 and 4.

After the Euclidian distance, we can try to code the Manhattan distance.

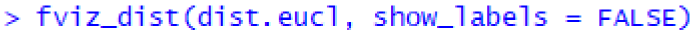


```
> dist.man<-dist(df,method="manhattan")
> round(as.matrix(dist.man)[1:10, 1:10],2)
       1    2     3     4    5    6    7    8    9   10
1   0.00 4.08  0.49 10.97 4.65 4.05 6.92 3.10 8.58 8.34
2   4.08 0.00  3.75  6.89 2.50 0.29 5.26 2.05 5.86 5.55
3   0.49 3.75  0.00 10.64 4.32 3.72 6.58 2.92 8.25 8.00
4  10.97 6.89 10.64  0.00 6.31 6.92 4.05 8.80 2.71 3.30
5   4.65 2.50  4.32  6.31 0.00 2.21 3.46 4.32 3.92 3.68
6   4.05 0.29  3.72  6.92 2.21 0.00 5.20 2.11 5.80 5.49
7   6.92 5.26  6.58  4.05 3.46 5.20 0.00 7.31 1.89 2.17
8   3.10 2.05  2.92  8.80 4.32 2.11 7.31 0.00 7.90 7.59
9   8.58 5.86  8.25  2.71 3.92 5.80 1.89 7.90 0.00 0.67
10  8.34 5.55  8.00  3.30 3.68 5.49 2.17 7.59 0.67 0.00
```

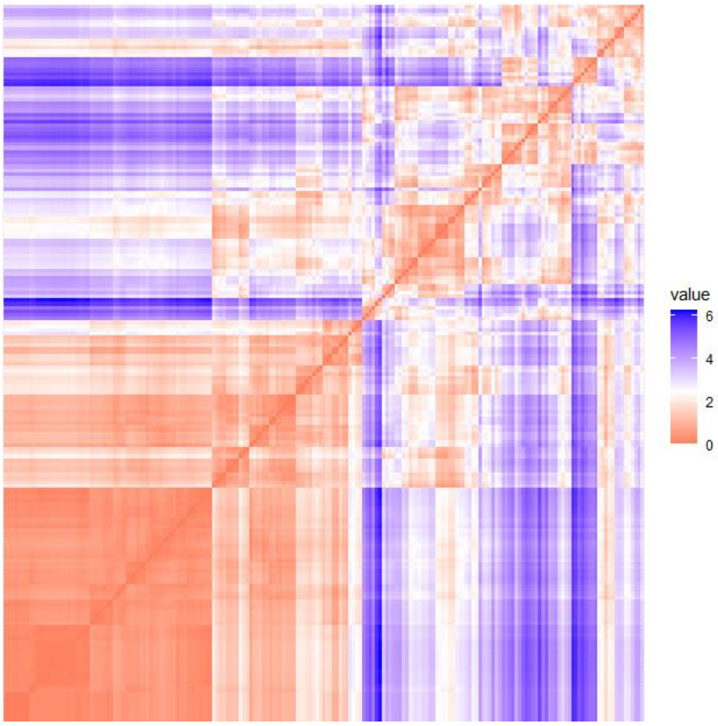
As we can see for this distance matrix, the result in terms of distance and similarity are the same of the previous distance matrix based on Euclidian distance.

Remember that we have used a subset of data.

Now we can experiment the graphical visualization of these distance matrices.

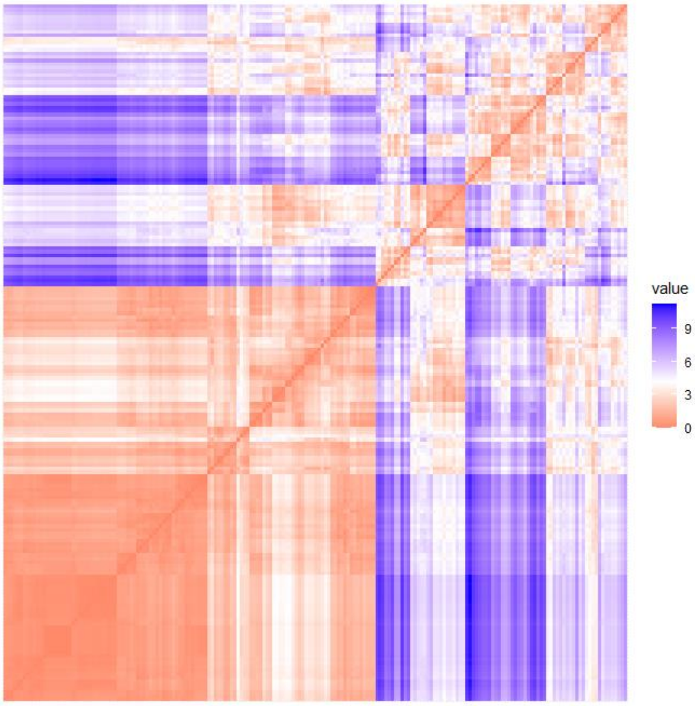


```
> fviz_dist(dist.eucl, show_labels = FALSE)
```



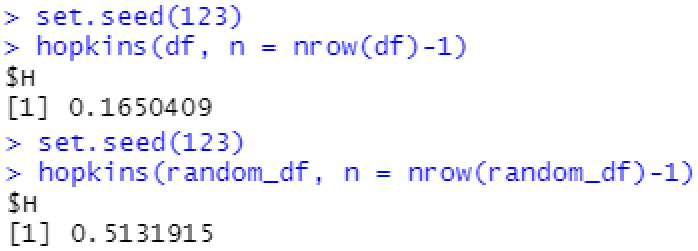
Next the Manhattan distance that is the same.

```
> fviz_dist(dist.man, show_labels = FALSE)
```



According with the images, a level of colour red indicates a higher similarity between the observations, and a level of colour blue indicates a lower similarity between observations.

Another approach that gives us important information about clustering is the ***Hopkins method***, that is a measure of clustering tendency that include the values in the interval [0;1].



```
> set.seed(123)
> hopkins(df, n = nrow(df)-1)
$H
[1] 0.1650409
> set.seed(123)
> hopkins(random_df, n = nrow(random_df)-1)
$H
[1] 0.5131915
```

As we can see from the computation above, the drinks dataset has a hopkins value (0.1650409) close to 0, so it is clusterable. Instead, the randomly generated has a hopinks value (0.5131915) that is above 0.50, so it is not clusterable.

CLUSTER ALGORITHMS

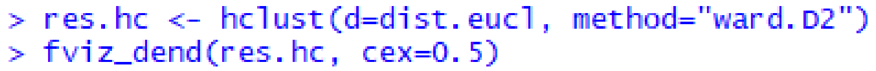
In order to determine the optimal number of clusters, there are two possible methods:

- One consists of using direct methods (Elbow and Average silhouette)
- The second is about the use of statistical testing methods, the gap statistic

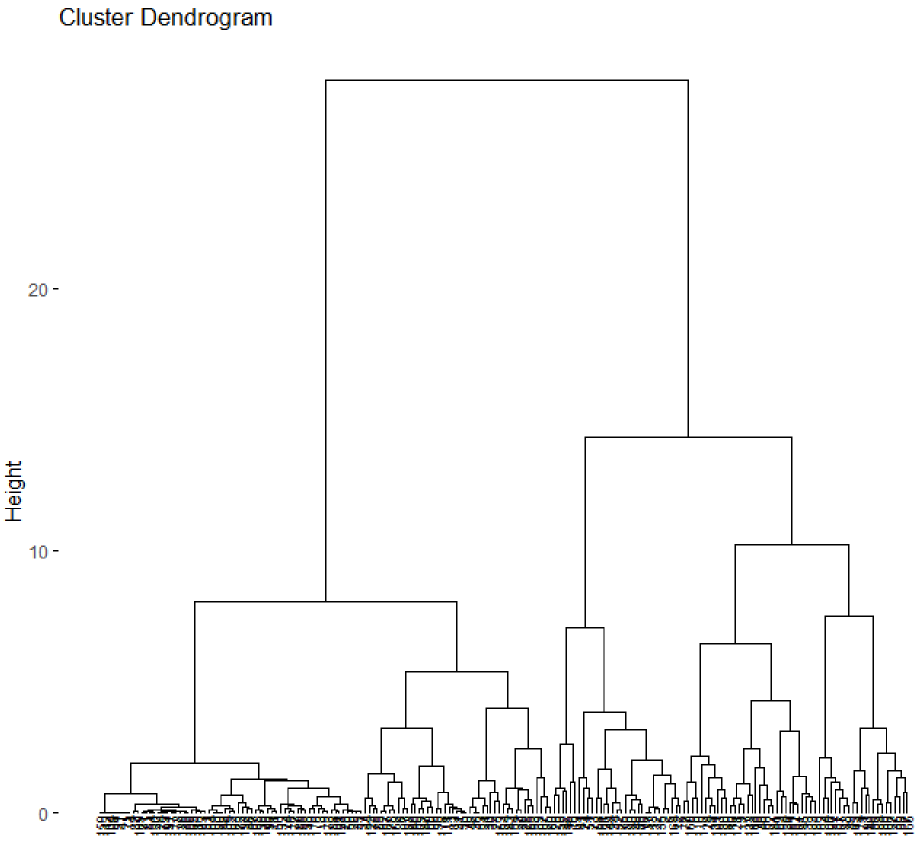
Agglomerative hierarchical clustering approach: is a bottom-up method in which we start with a different cluster for each observation K=n, and then we'll reach a situation where there will be one cluster of all the observations K=1.

1.1 AHC based Ward's linkage method and Euclidean distance

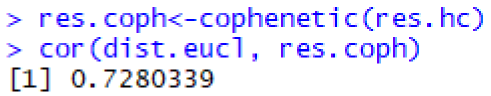
We already computed the Euclidean distance, so we will perform the dendrogram using the Ward's linkage method in the function.



```
> res.hc <- hclust(d=dist.eucl, method="ward.D2")
> fviz_dend(res.hc, cex=0.5)
```



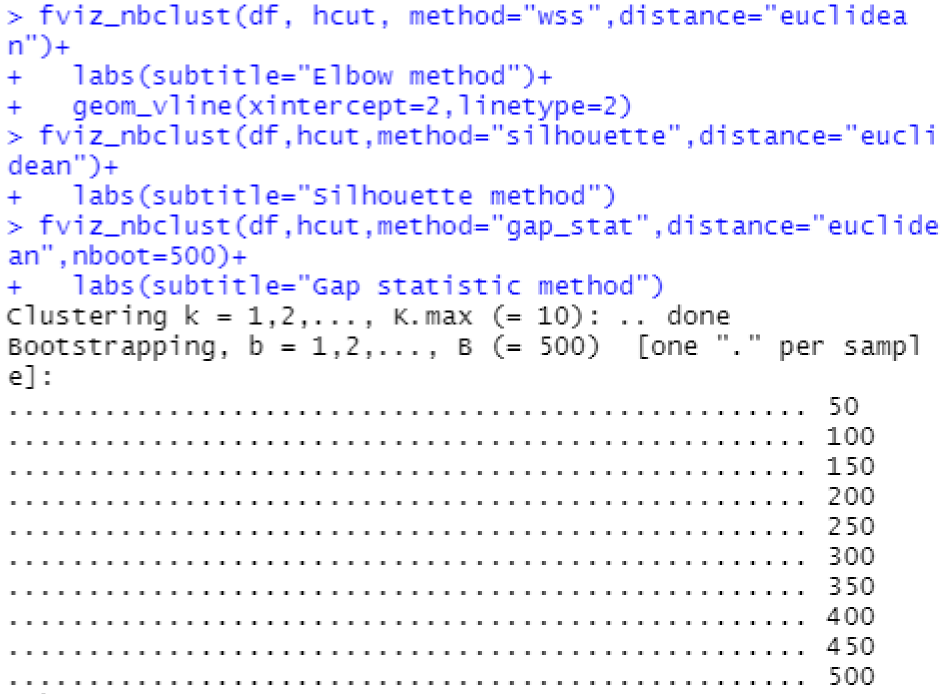
Once we have the dendrogram, we can see which units are close or not, but we can't use the closeness criterion to say that to units are similar. So, to say if that clustering approach is good or not, we must compute the cophenetic distance, that will tell us if the computed value is close to 1. So, this is an accuracy problem.



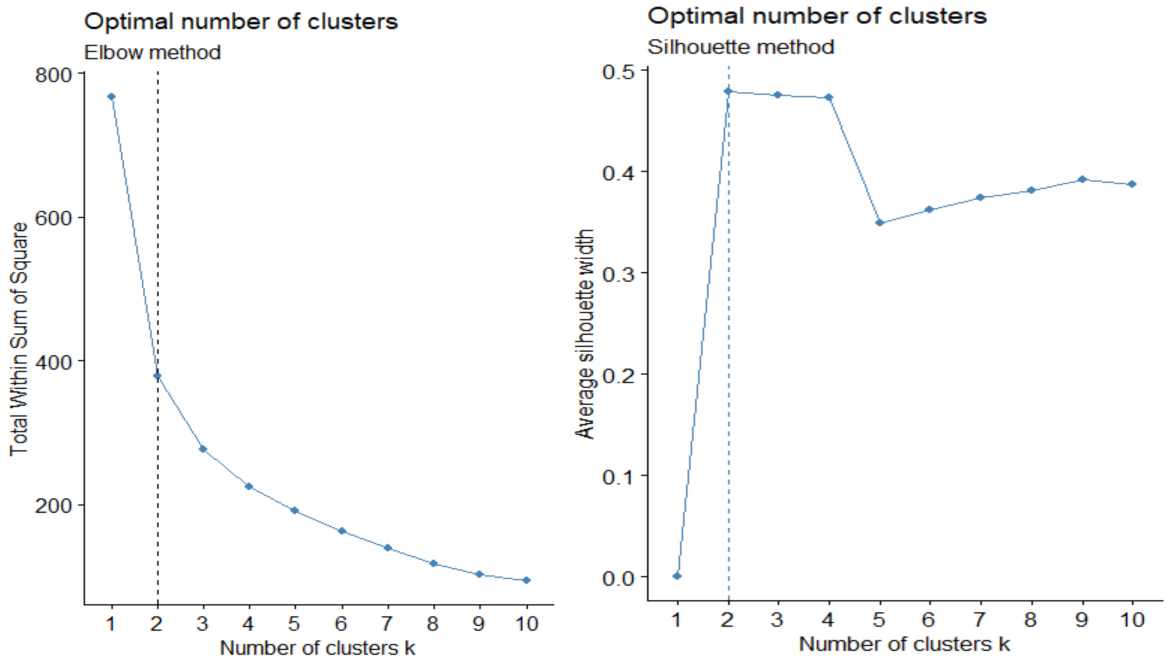
```
> res.coph<-cophenetic(res.hc)
> cor(dist.eucl, res.coph)
[1] 0.7280339
```

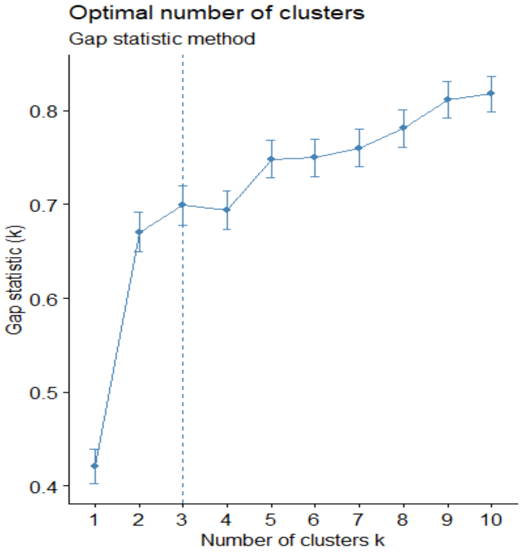
The result is almost 0.73, so isn't good, because the values above 0.75 are felt to be good. This means that the clustering approach that we used does not preserves the original distances between units.

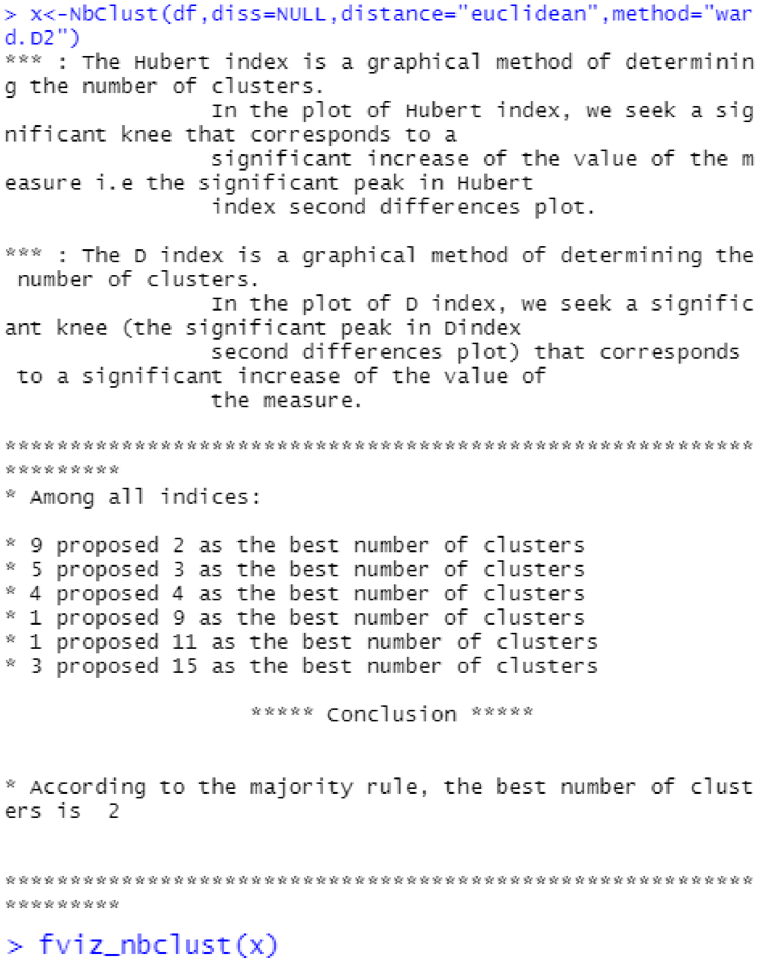
Next, the computation of the optimal number of clusters K.



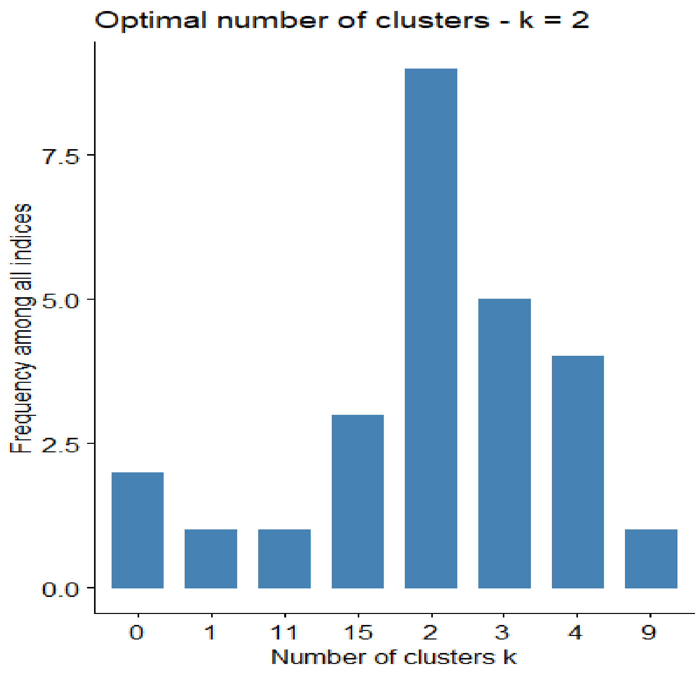
```
> fviz_nbclust(df, hcut, method="wss",distance="euclidea
n")+
+   labs(subtitle="Elbow method")+
+   geom_vline(xintercept=2,linetype=2)
> fviz_nbclust(df,hcut,method="silhouette",distance="eucli
dean")+
+   labs(subtitle="Silhouette method")
> fviz_nbclust(df,hcut,method="gap_stat",distance="euclide
an",nboot=500)+
+   labs(subtitle="Gap statistic method")
Clustering k = 1,2,..., K.max (= 10): .. done
Bootstrapping, b = 1,2,..., B (= 500)  [one "." per sampl
e]:
.................................................. 50
.................................................. 100
.................................................. 150
.................................................. 200
.................................................. 250
.................................................. 300
.................................................. 350
.................................................. 400
.................................................. 450
.................................................. 500
```





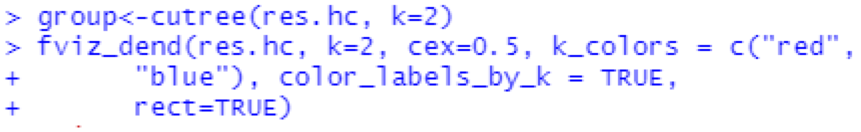


```
> x<-NbClust(df,diss=NULL,distance="euclidean",method="war
d.D2")
*** : The Hubert index is a graphical method of determinin
g the number of clusters.
                In the plot of Hubert index, we seek a sig
nificant knee that corresponds to a
                significant increase of the value of the m
easure 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 signific
ant 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
* 4 proposed 4 as the best number of clusters
* 1 proposed 9 as the best number of clusters
* 1 proposed 11 as the best number of clusters
* 3 proposed 15 as the best number of clusters

                   ***** Conclusion *****


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


***********************************************************
*********
> fviz_nbclust(x)
```

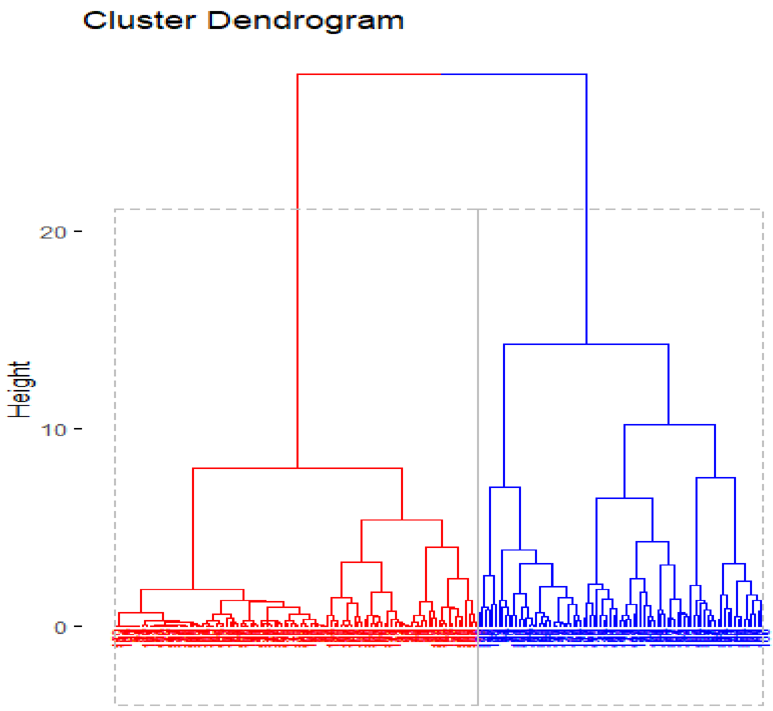


The optimal number of clusters K=2.

After that conclusion, we can see also the clusters by specifying the number of groups, in this case 2.



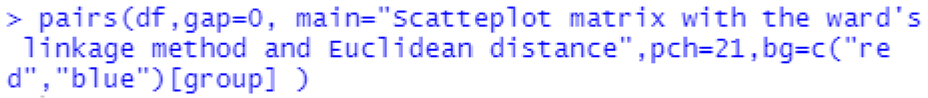
```
> group<-cutree(res.hc, k=2)
> fviz_dend(res.hc, k=2, cex=0.5, k_colors = c("red",
+         "blue"), color_labels_by_k = TRUE,
+         rect=TRUE)
```



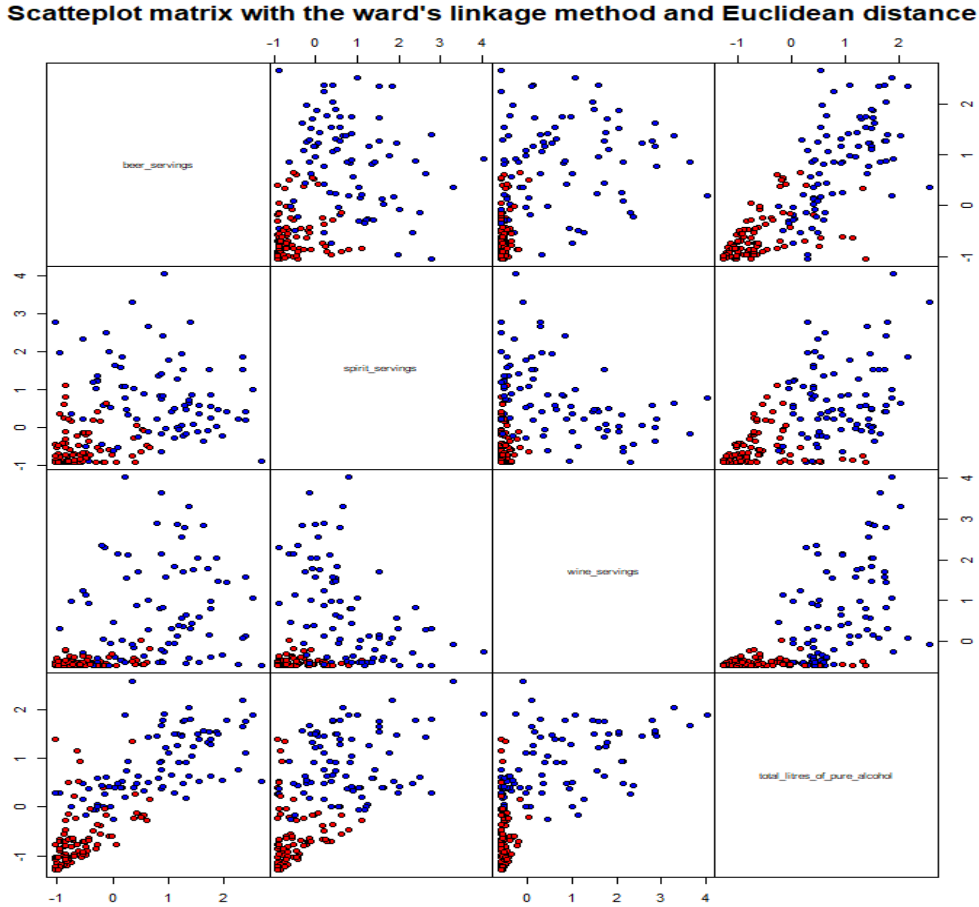
So, the red ones are in the cluster 1, the blue units in cluster 2.

We can also affirm that the cluster in red, so the first one, is the biggest.

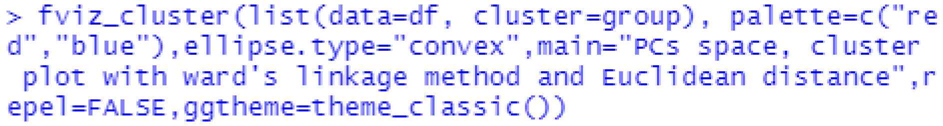
Visualization of the clustering result in the original space:



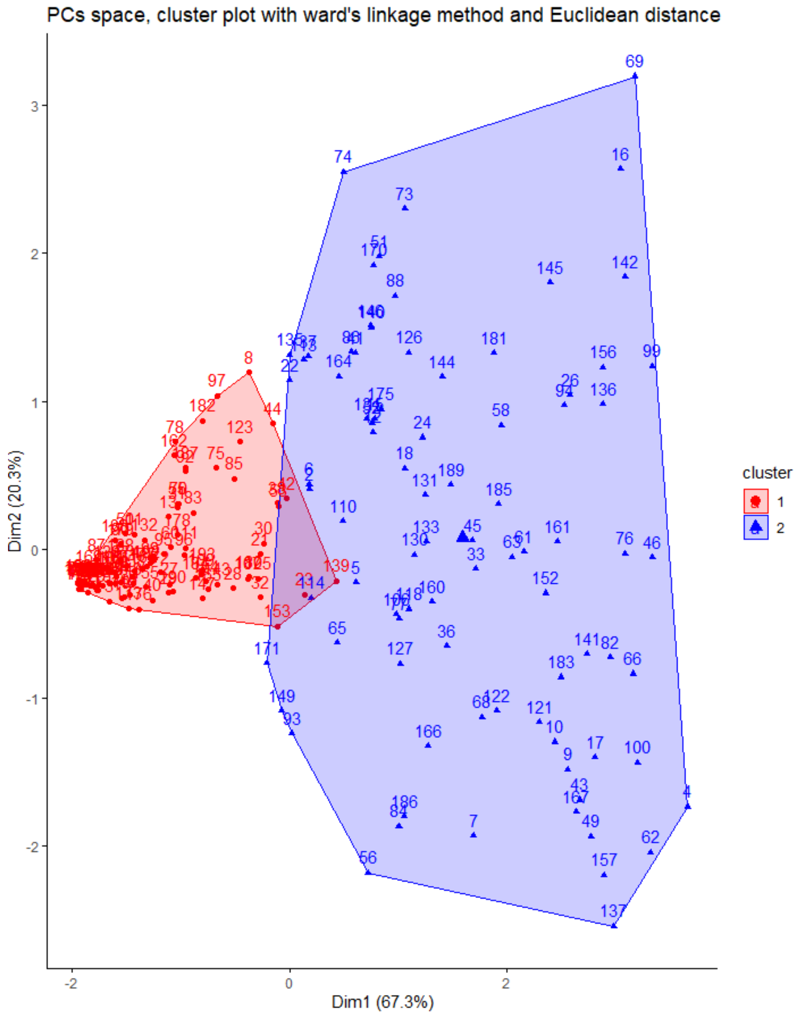
```
> pairs(df,gap=0, main="Scatteplot matrix with the ward's
 linkage method and Euclidean distance",pch=21,bg=c("re
d","blue")[group] )
```



After that, we are going to compute the scatterplot trough the Ward’s linkage method and Euclidean distance in the PCs space.



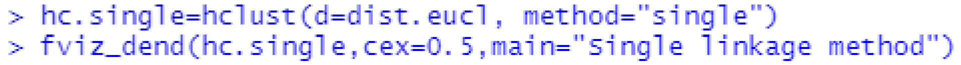
```
> fviz_cluster(list(data=df, cluster=group), palette=c("re
d","blue"),ellipse.type="convex",main="PCs space, cluster
 plot with ward's linkage method and Euclidean distance",r
epel=FALSE,ggtheme=theme_classic())
```



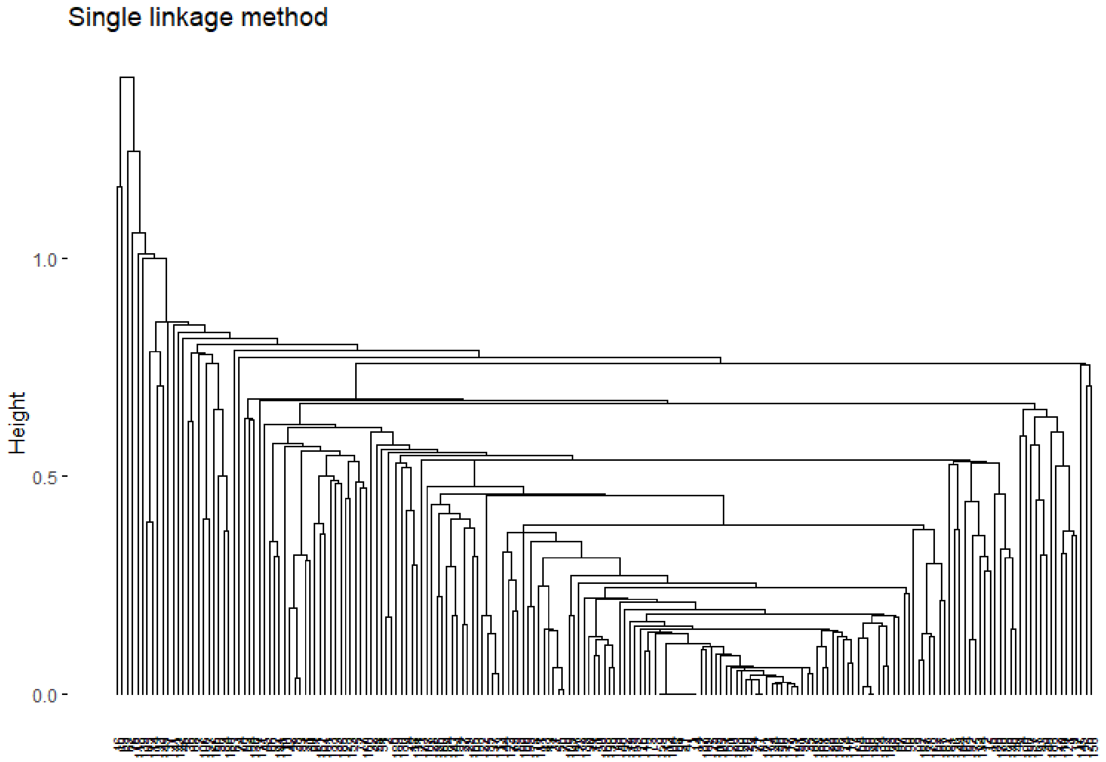
Both clusters are not well separated.

1.2 AHC based on single linkage method and Euclidian distance

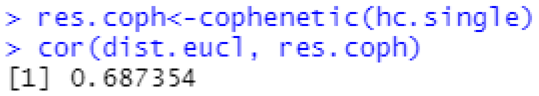
Performing the dendrogram using the single linkage method and Euclidean distance.



```
> hc.single=hclust(d=dist.eucl, method="single")
> fviz_dend(hc.single,cex=0.5,main="Single linkage method")
```



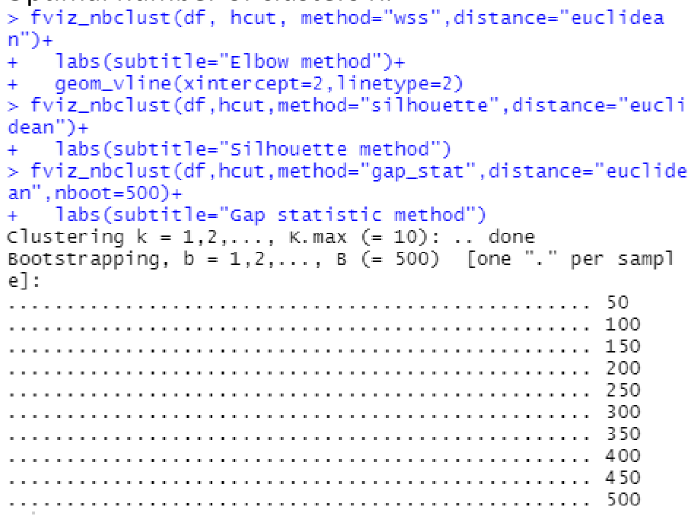
To establish if this approach is good or not, we must compute the cophenetic distance and check for a correlation with the original one.



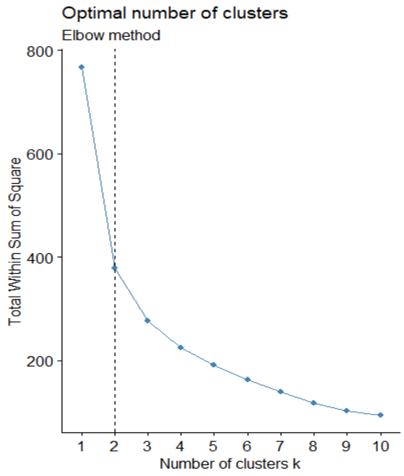
```
> res.coph<-cophenetic(hc.single)
> cor(dist.eucl, res.coph)
[1] 0.687354
```

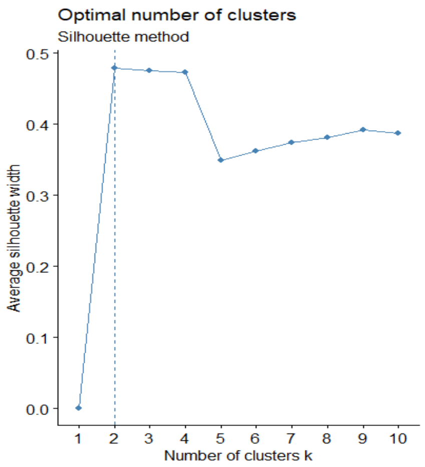
The result of the cophenetic distance (0.687354) is not good because the value should be above the 0.75. This value indicates that this clustering method doesn't preserve the true original distance between units.

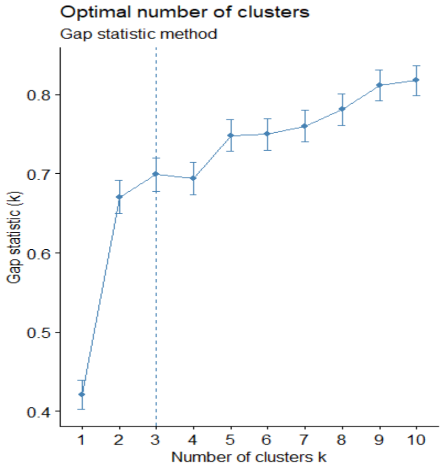
Optimal number of clusters K:

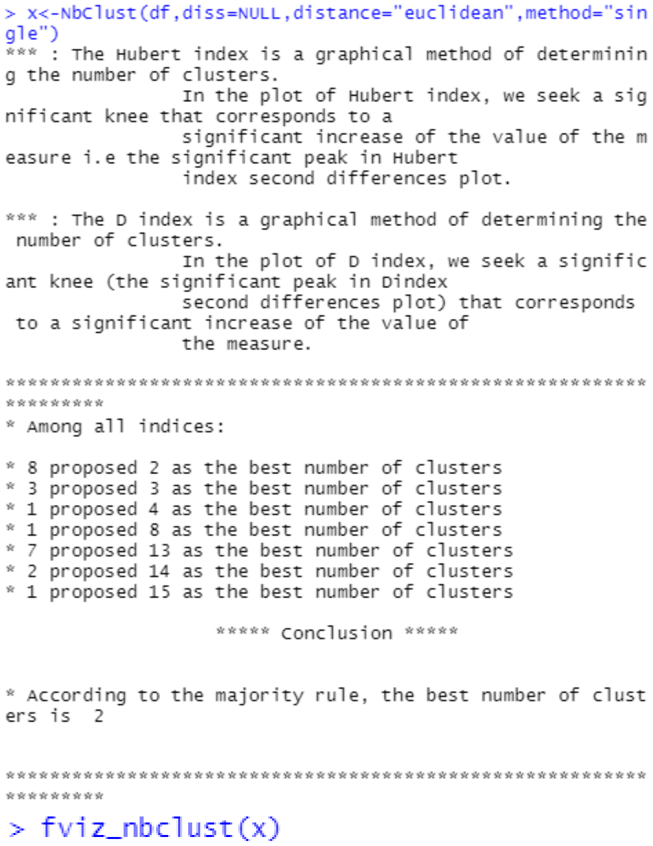


```
> fviz_nbclust(df, hcut, method="wss",distance="euclidea
n")+
+   labs(subtitle="Elbow method")+
+   geom_vline(xintercept=2,linetype=2)
> fviz_nbclust(df,hcut,method="silhouette",distance="eucli
dean")+
+   labs(subtitle="Silhouette method")
> fviz_nbclust(df,hcut,method="gap_stat",distance="euclide
an",nboot=500)+
+   labs(subtitle="Gap statistic method")
Clustering k = 1,2,..., K.max (= 10): .. done
Bootstrapping, b = 1,2,..., B (= 500)  [one "." per sampl
e]:
..................................................  50
.................................................. 100
.................................................. 150
.................................................. 200
.................................................. 250
.................................................. 300
.................................................. 350
.................................................. 400
.................................................. 450
.................................................. 500
```

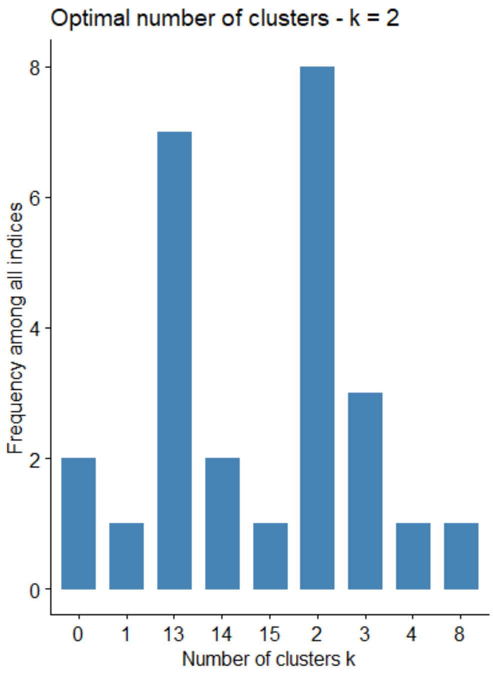




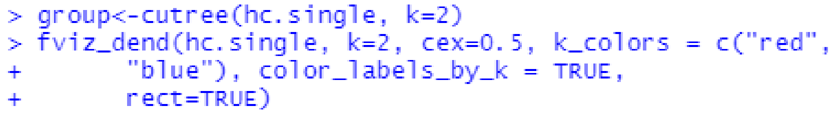




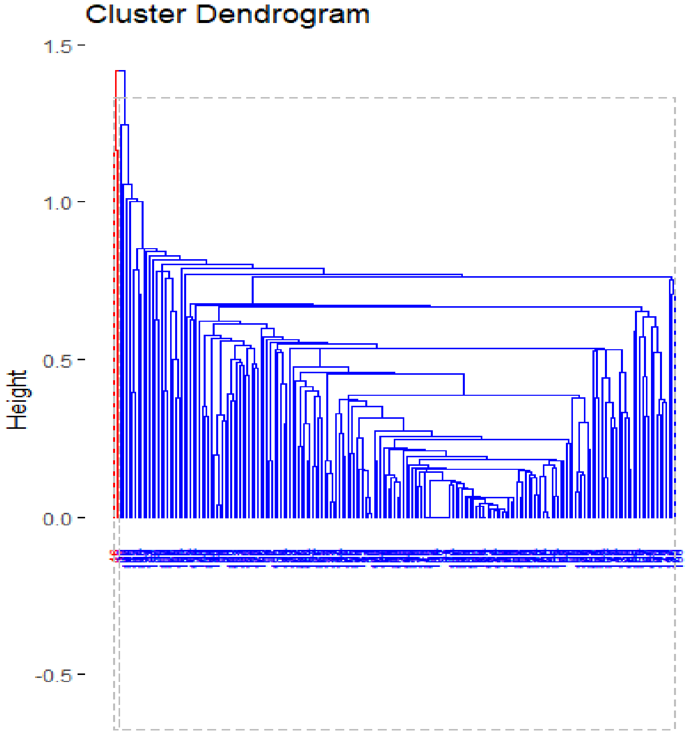
```
> x<-NbClust(df,diss=NULL,distance="euclidean",method="sin
gle")
*** : The Hubert index is a graphical method of determinin
g the number of clusters.
                In the plot of Hubert index, we seek a sig
nificant knee that corresponds to a
                significant increase of the value of the m
easure 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 signific
ant 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
* 3 proposed 3 as the best number of clusters
* 1 proposed 4 as the best number of clusters
* 1 proposed 8 as the best number of clusters
* 7 proposed 13 as the best number of clusters
* 2 proposed 14 as the best number of clusters
* 1 proposed 15 as the best number of clusters

                   ***** Conclusion *****


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

***********************************************************
*********
> fviz_nbclust(x)
```



So, according with the operations, the best number of clusters is K=2. Next, we are going to see the dendrogram cut by the K value.



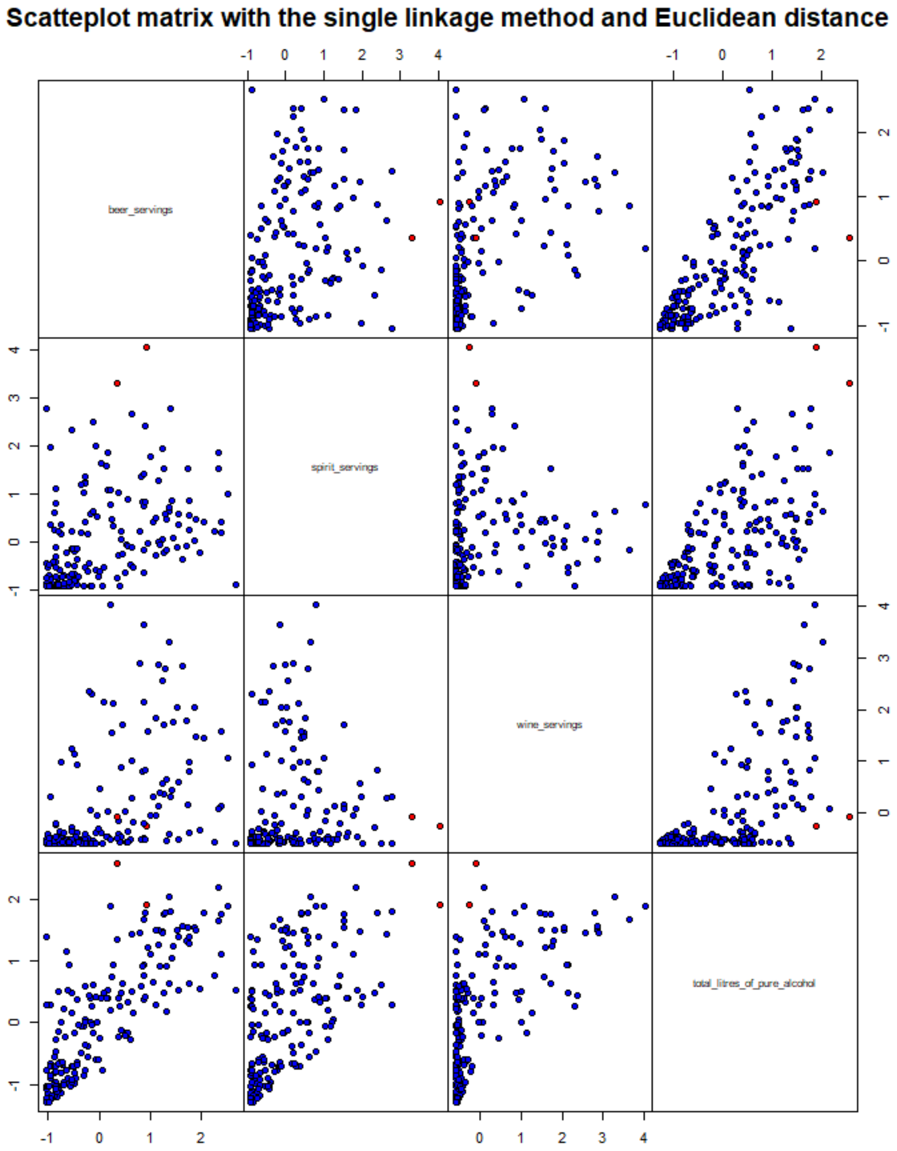
```
> group<-cutree(hc.single, k=2)
> fviz_dend(hc.single, k=2, cex=0.5, k_colors = c("red",
+        "blue"), color_labels_by_k = TRUE,
+        rect=TRUE)
```



We can see from the image above, that the data partition computed is not good, this because almost the entire units are grouped in a single cluster.

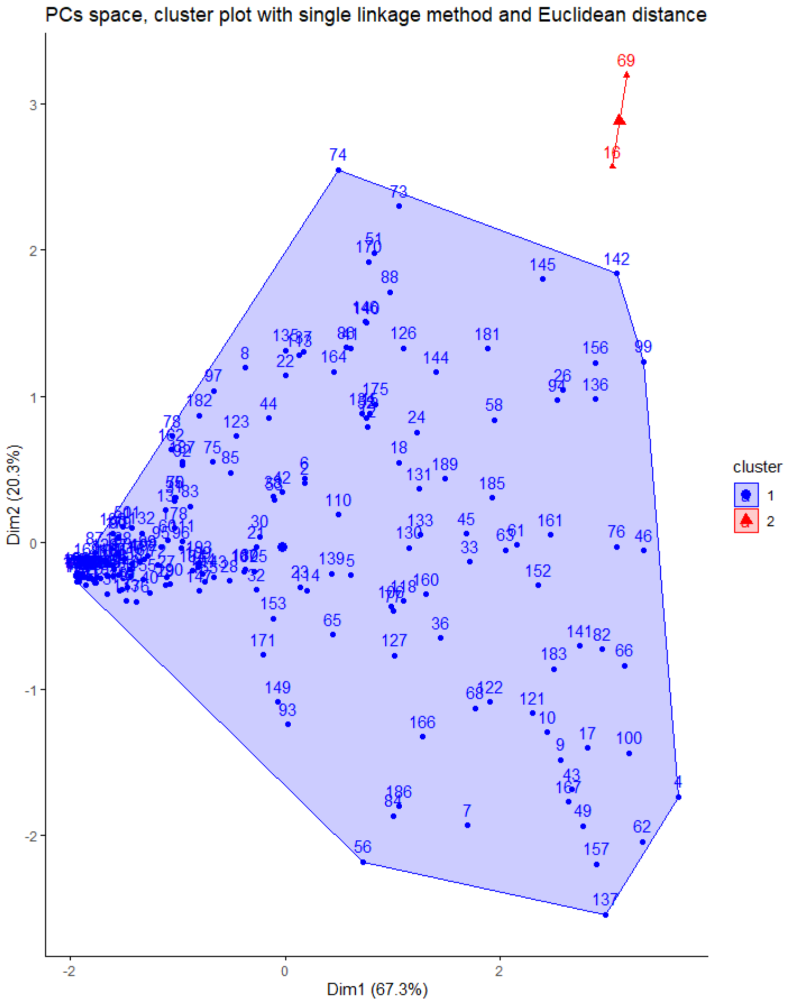
Visualization of the clustering results in the original space:

```
> pairs(df,gap=0, main="Scatteplot matrix with the single
 linkage method and Euclidean distance",pch=21,bg=c("blu
e","red")[group] )
```



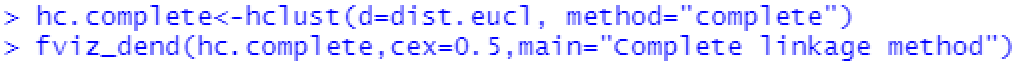
Visualization in the PCs space.

```
> fviz_cluster(list(data=df, cluster=group), palette=c("bl
ue","red"),ellipse.type="convex",main="PCs space, cluster
 plot with single linkage method and Euclidean distance",r
epel=FALSE,ggtheme=theme_classic())
```

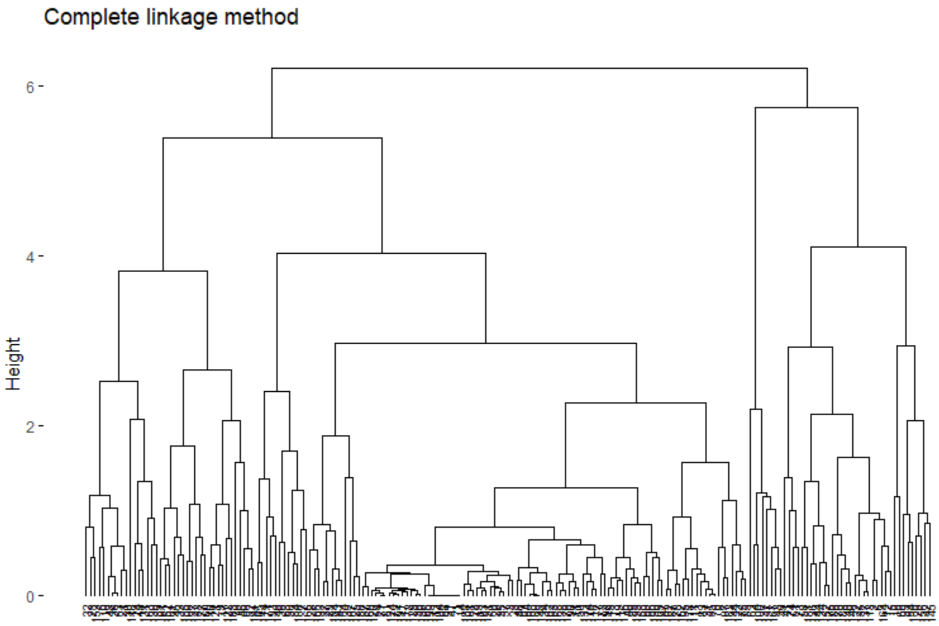

PCs space, cluster plot with single linkage method and Euclidean distance
Dim2 (20.3%)
Dim1 (67.3%)
3
2
1
0
-1
-2
-2
0
2
cluster
1
2
69
16
74
73
51
170
145
142
88
140
126
181
156
99
135
137
118
89
164
144
26
8
22
97
94
136
182
175
58
44
24
123
78
162
187
75
18
189
85
6
2
131
185
34
42
110
133
45
161
30
130
63
61
76
46
33
139
5
152
23
114
160
153
65
36
141
182
171
127
183
66
149
93
68
122
121
10
166
17
100
9
43
167
49
186
84
7
62
56
157
137
4

1.3 AHC based on complete linkage method and Euclidean distance

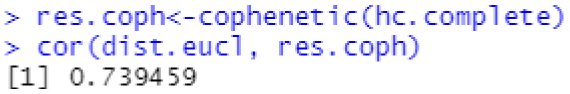
Performing the dendrogram using the complete linkage method and Euclidean distance.



```
> hc.complete<-hclust(d=dist.eucl, method="complete")
> fviz_dend(hc.complete,cex=0.5,main="Complete linkage method")
```

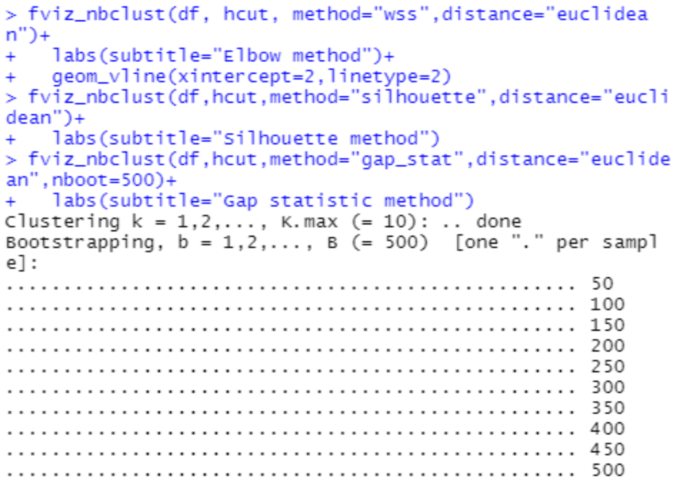


To establish if this approach is good or not, we must compute the cophenetic distance and check for a correlation with the original one.

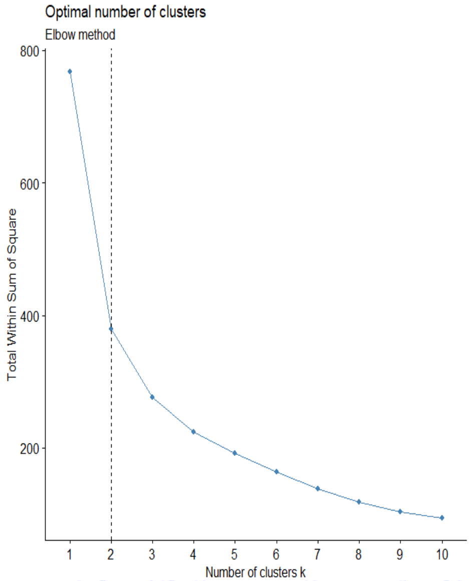


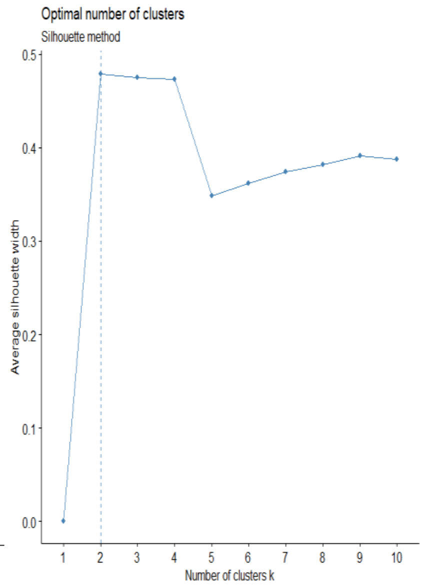
```
> res.coph<-cophenetic(hc.complete)
> cor(dist.eucl, res.coph)
[1] 0.739459
```

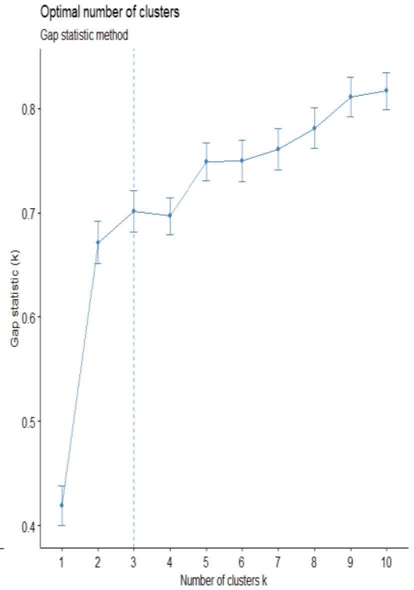
As we can see above, the cophenetic distance's value (0.739459) not reach the threshold value 0.75. So, this method doesn't preserve the true original distance between units.
Find the optimal number of clusters K:

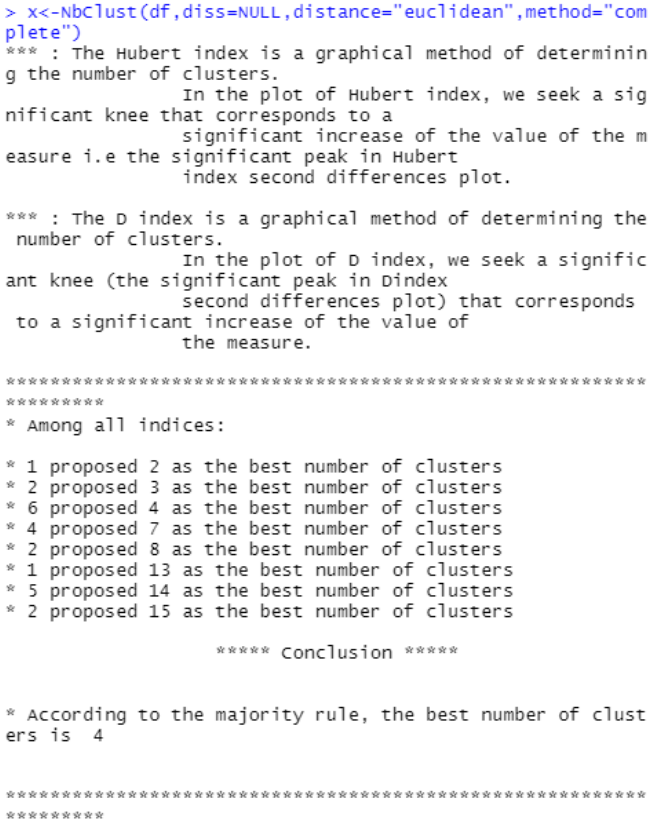


```
> fviz_nbclust(df, hcut, method="wss",distance="euclidea
n")+
+   labs(subtitle="Elbow method")+
+   geom_vline(xintercept=2,linetype=2)
> fviz_nbclust(df,hcut,method="silhouette",distance="eucli
dean")+
+   labs(subtitle="Silhouette method")
> fviz_nbclust(df,hcut,method="gap_stat",distance="euclide
an",nboot=500)+
+   labs(subtitle="Gap statistic method")
Clustering k = 1,2,..., K.max (= 10): .. done
Bootstrapping, b = 1,2,..., B (= 500)  [one "." per sampl
e]:
.................................................. 50
.................................................. 100
.................................................. 150
.................................................. 200
.................................................. 250
.................................................. 300
.................................................. 350
.................................................. 400
.................................................. 450
.................................................. 500
```



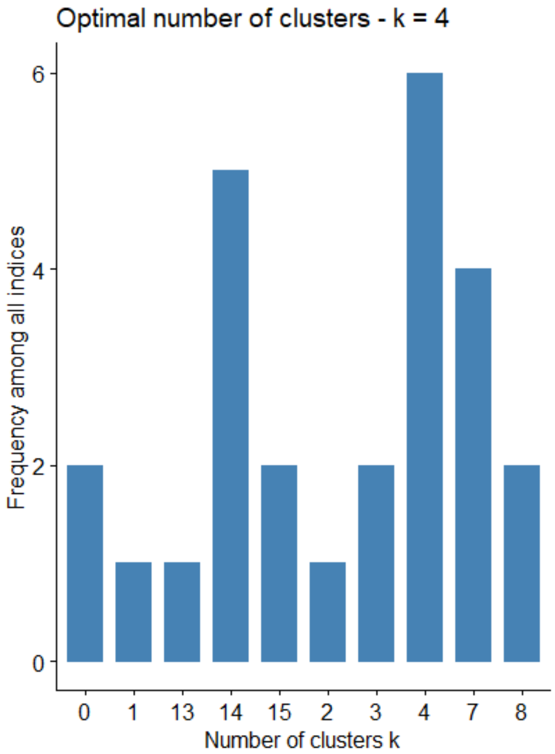






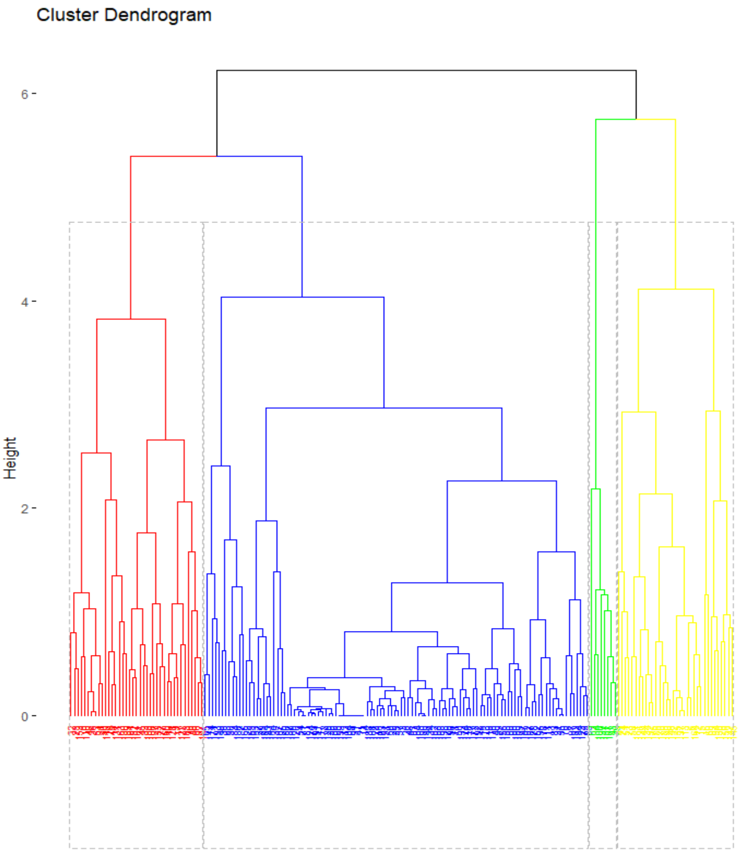
```
> x<-NbClust(df,diss=NULL,distance="euclidean",method="com
plete")
*** : The Hubert index is a graphical method of determinin
g the number of clusters.
                In the plot of Hubert index, we seek a sig
nificant knee that corresponds to a
                significant increase of the value of the m
easure 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 signific
ant knee (the significant peak in Dindex
                second differences plot) that corresponds
 to a significant increase of the value of
                the measure.

***********************************************************
*********
* Among all indices:

* 1 proposed 2 as the best number of clusters
* 2 proposed 3 as the best number of clusters
* 6 proposed 4 as the best number of clusters
* 4 proposed 7 as the best number of clusters
* 2 proposed 8 as the best number of clusters
* 1 proposed 13 as the best number of clusters
* 5 proposed 14 as the best number of clusters
* 2 proposed 15 as the best number of clusters

                   ***** Conclusion *****


* According to the majority rule, the best number of clust
ers is  4


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

```
> fviz_nbclust(x)
```



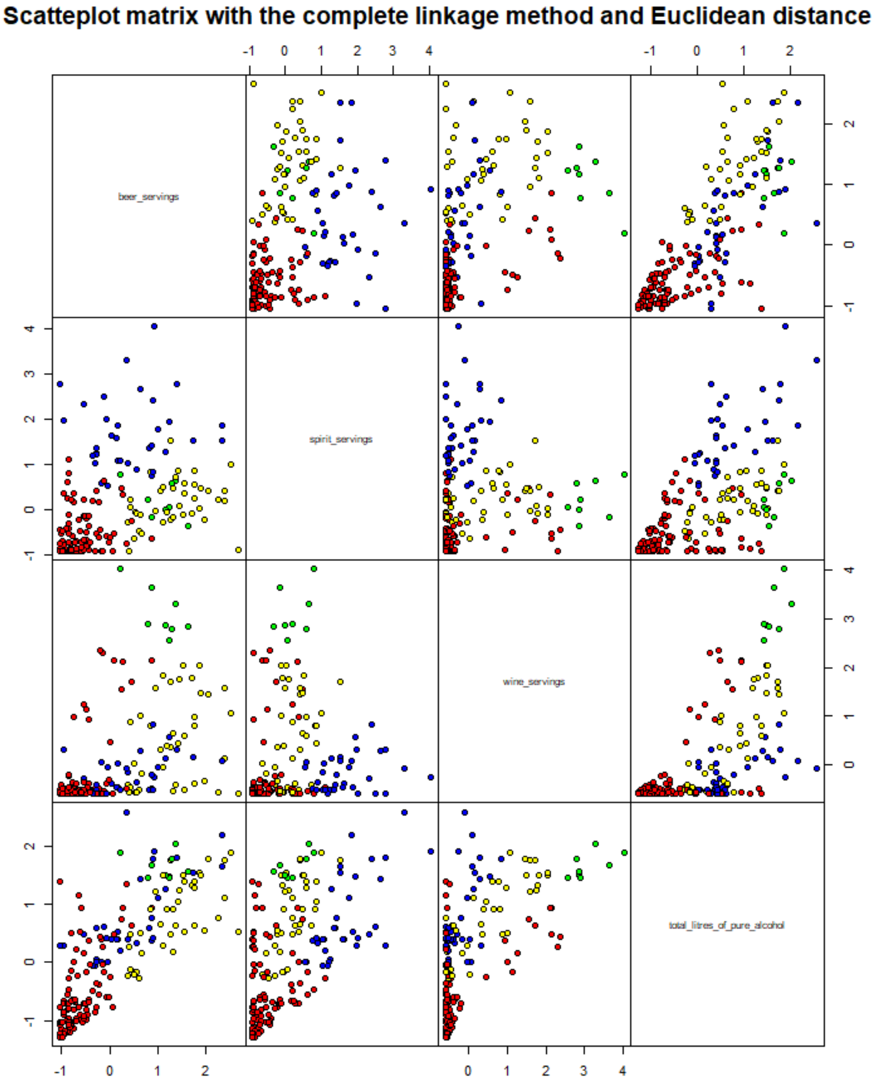
According to the graph, the number of clusters is K=4.

```
> group<-cutree(hc.complete, k=4)
> fviz_dend(hc.complete, k=4, cex=0.5, k_colors = c("red",
+         "blue","green","yellow"), color_labels_by_k = TRU
E,
+         rect=TRUE)
```



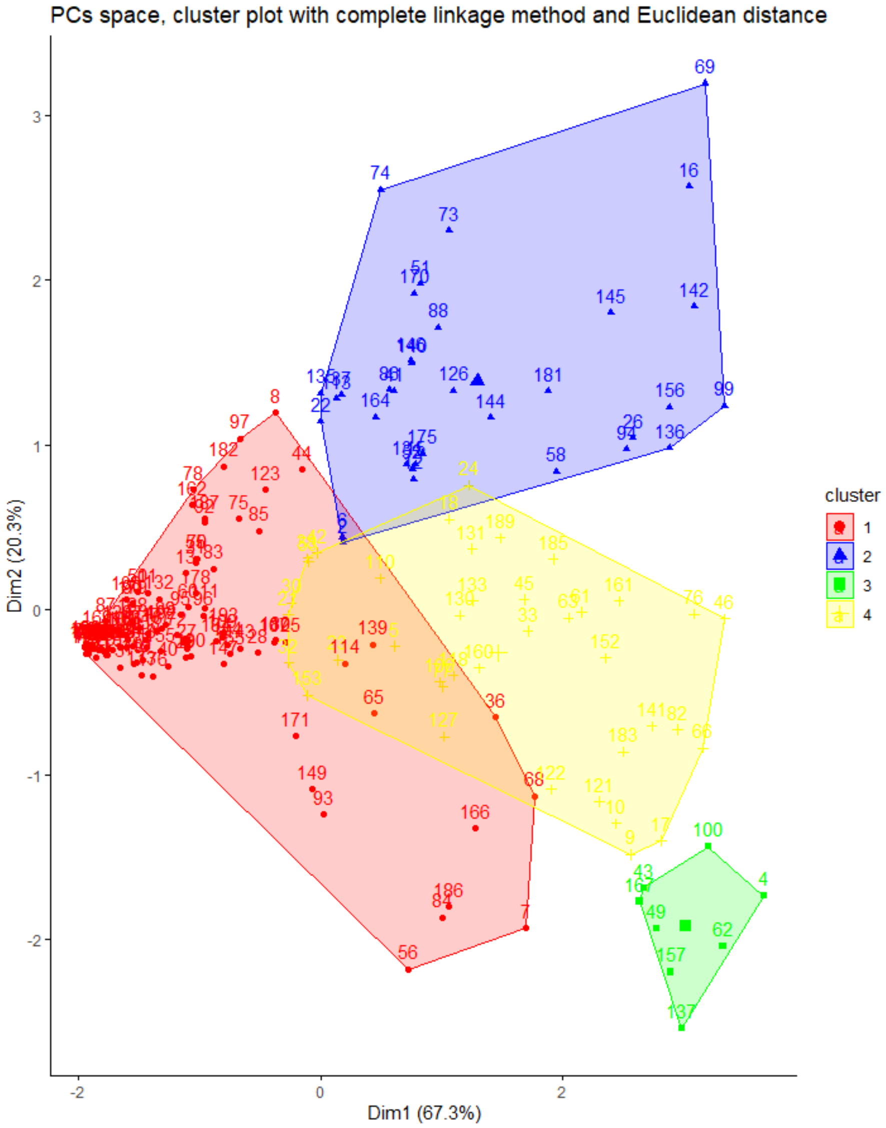
Visualization of the clustering results in the original space:

```
> pairs(df,gap=0, main="Scatteplot matrix with the complet
e linkage method and Euclidean distance",pch=21,bg=c("re
d","blue","green","yellow")[group] )
```



Visualization in the PCs space.

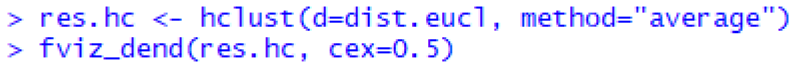
```
> fviz_cluster(list(data=df, cluster=group), palette=c("re
d","blue","green","yellow"),ellipse.type="convex",main="PC
s space, cluster plot with complete linkage method and Euc
lidean distance",repel=FALSE,ggtheme=theme_classic())
```


PCs space, cluster plot with complete linkage method and Euclidean distance
Dim2 (20.3%)
Dim1 (67.3%)
cluster
1
2
3
4
-2
0
2
-2
-1
0
1
2
3
69
74
16
73
145
142
88
126
181
144
164
156
99
58
136
24
189
185
131
161
76
46
45
33
152
183
66
121
10
9
17
100
4
43
49
62
157
137
8
97
182
44
123
78
75
85
139
114
65
171
36
149
93
166
68
186
84
7
56

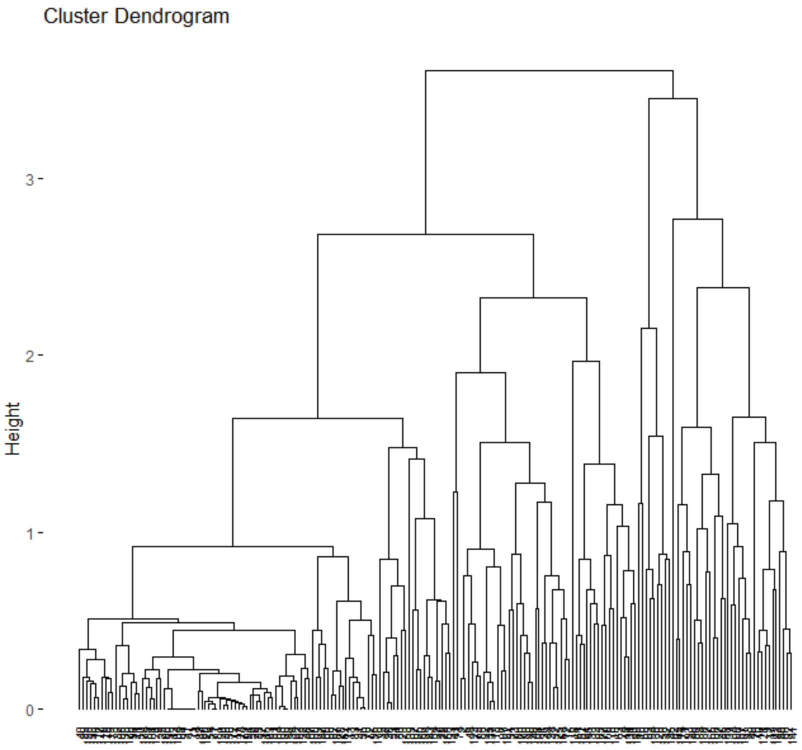
1.4 AHC based on average linkage method and Euclidean distance

As we proceeded before, we have to compute another approach.

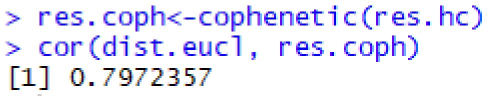
With the following code, there will be the computation of dendrogram, this time with the average linkage method.



```
> res.hc <- hclust(d=dist.eucl, method="average")
> fviz_dend(res.hc, cex=0.5)
```



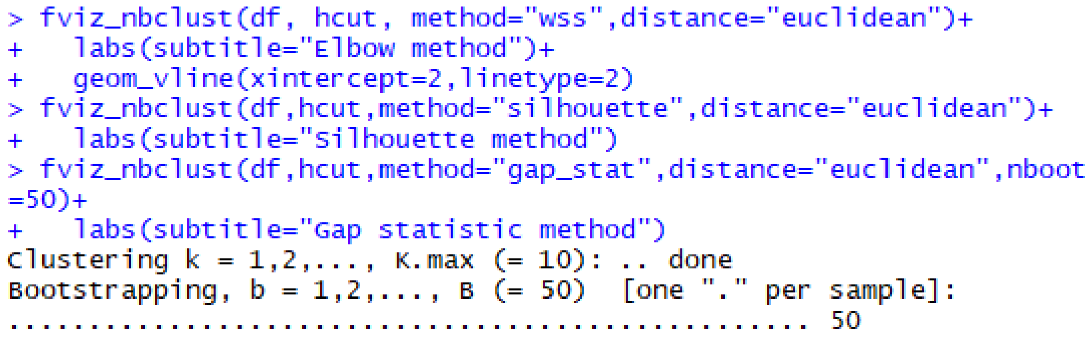
As done in the previous method, in order to decide if this approach is good or not, we'll compute the cophenetic distance.



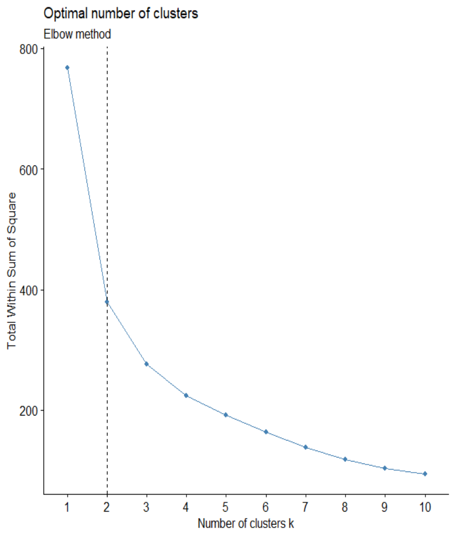
```
> res.coph<-cophenetic(res.hc)
> cor(dist.eucl, res.coph)
[1] 0.7972357
```

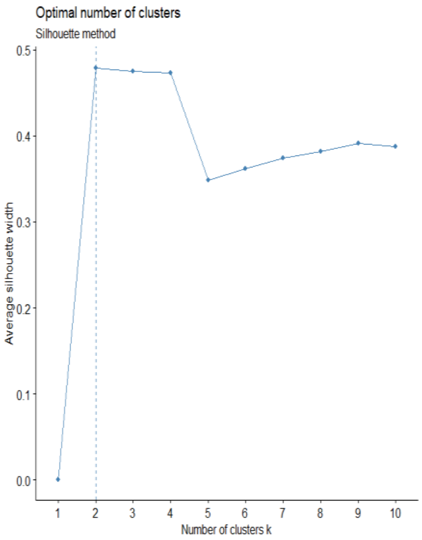
This time, as we can see, the result is almost 0.80, that is greater than 0.75 that is the threshold value for the cophenetic distance. This result tells us that the approach used in this case preserves the true original distance between units.

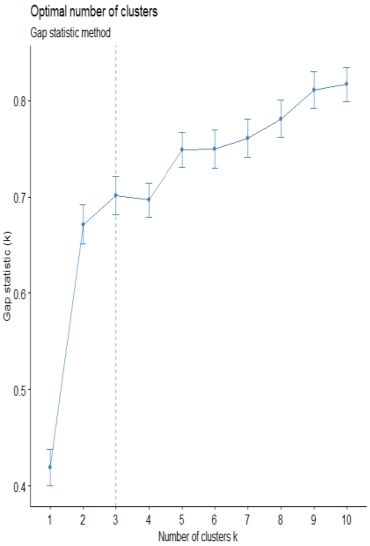
After that, we must find the optimal number of clusters because we have to cut the dendrogram:



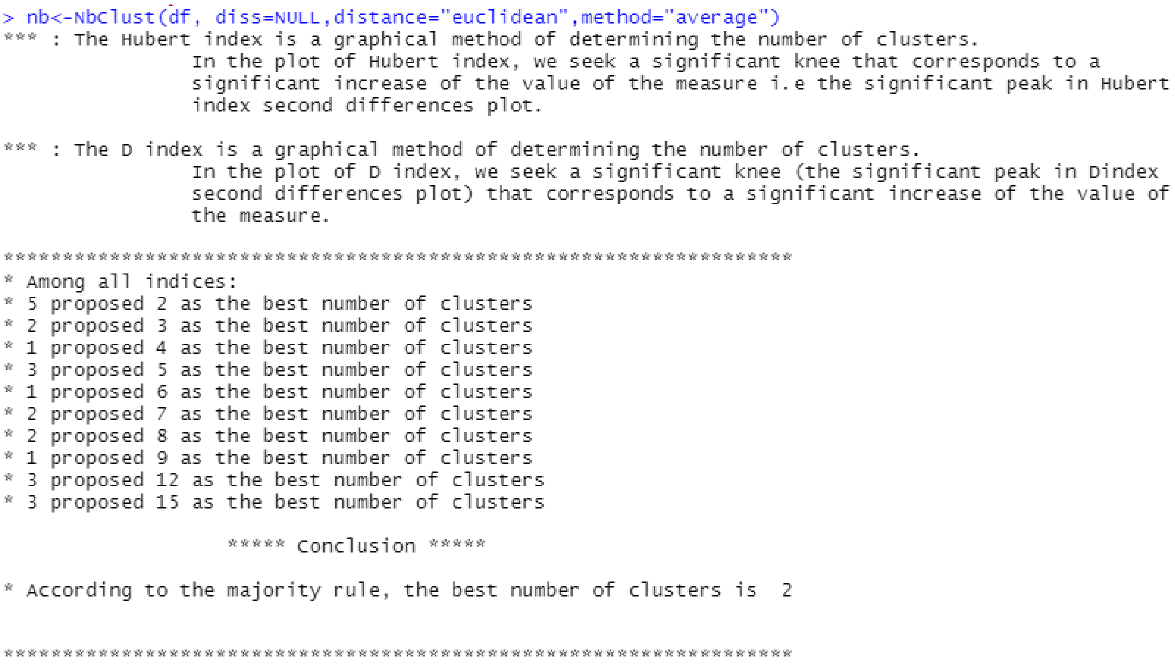
```
> fviz_nbclust(df, hcut, method="wss",distance="euclidean")+
+   labs(subtitle="Elbow method")+
+   geom_vline(xintercept=2,linetype=2)
> fviz_nbclust(df,hcut,method="silhouette",distance="euclidean")+
+   labs(subtitle="Silhouette method")
> fviz_nbclust(df,hcut,method="gap_stat",distance="euclidean",nboot
=50)+
+   labs(subtitle="Gap statistic method")
Clustering k = 1,2,..., K.max (= 10): .. done
Bootstrapping, b = 1,2,..., B (= 50)  [one "." per sample]:
.................................................. 50
```



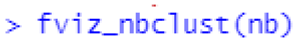




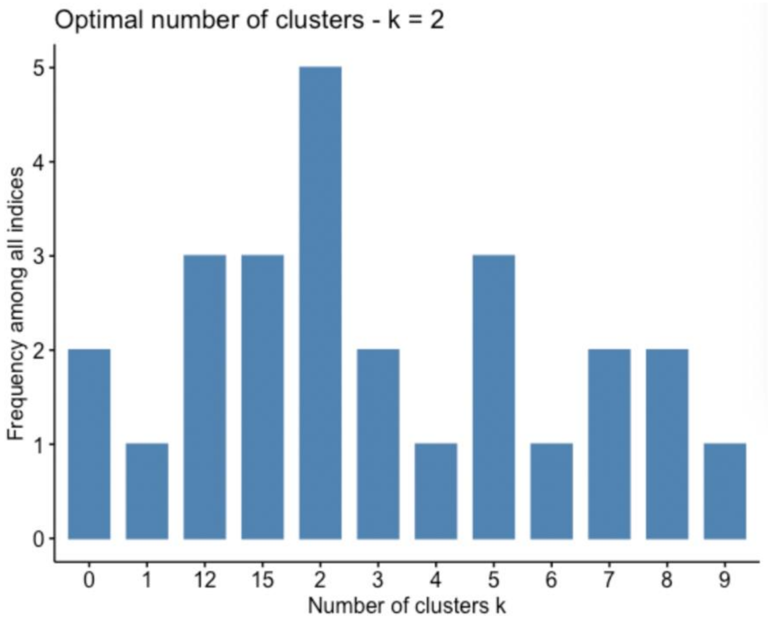
The possible solutions are, for the elbow method 2 clusters, for the silhouette method 2 clusters and for the gap statistic method 3 clusters.
So, according with the previous result of elbow and silhouette, we can define K=2; this result is confirmed by the output of the function NbCust().



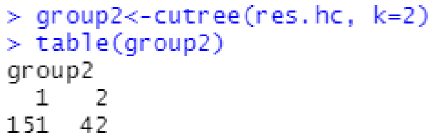
```
> nb<-NbClust(df, diss=NULL,distance="euclidean",method="average")
*** : The Hubert index is a graphical method of determining the number of clusters.
                In the plot of Hubert index, we seek a significant knee that corresponds to a
                significant increase of the value of 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
* 2 proposed 3 as the best number of clusters
* 1 proposed 4 as the best number of clusters
* 3 proposed 5 as the best number of clusters
* 1 proposed 6 as the best number of clusters
* 2 proposed 7 as the best number of clusters
* 2 proposed 8 as the best number of clusters
* 1 proposed 9 as the best number of clusters
* 3 proposed 12 as the best number of clusters
* 3 proposed 15 as the best number of clusters

                   ***** Conclusion *****

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


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



```
> fviz_nbclust(nb)
```



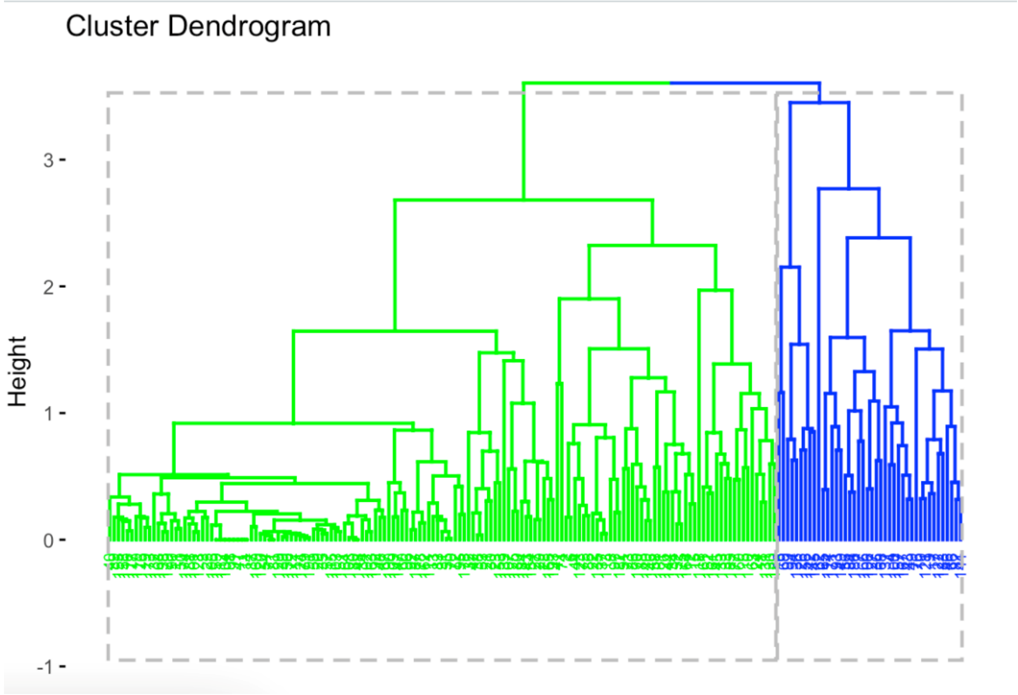
We can also compute:


```
> group2<-cutree(res.hc, k=2)
> table(group2)
group2
  1   2 
151  42 
```

As we can see 151 units belong to the cluster 1, and 42 units belong to cluster 2.

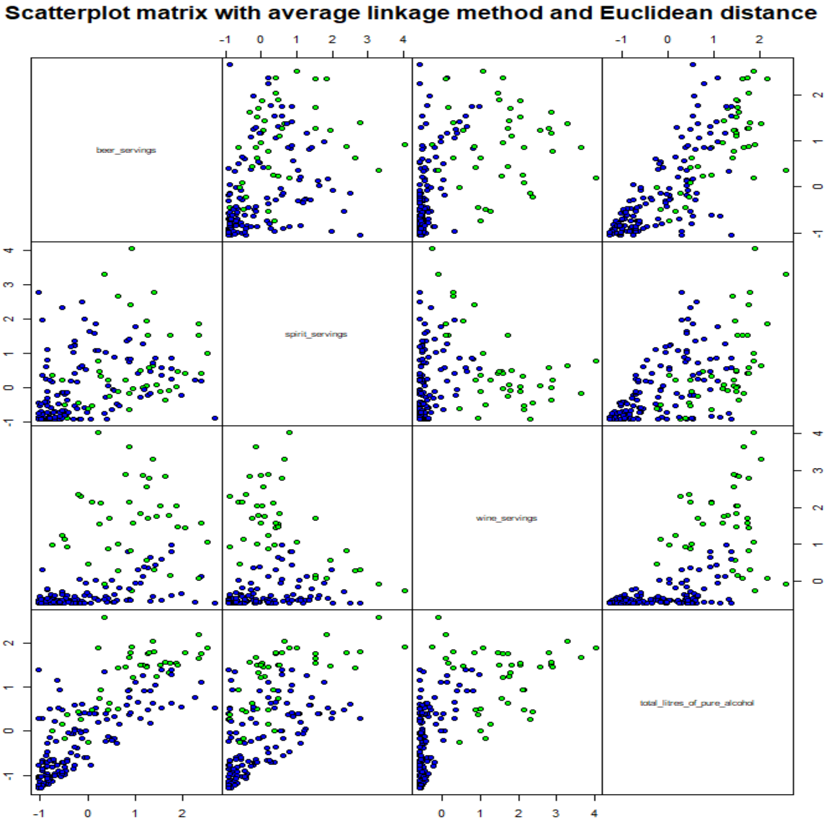
More specific, the 78% of the observations are in the first cluster, and the 22% are on the second cluster.

Then we can organize the dendrogram by colours.



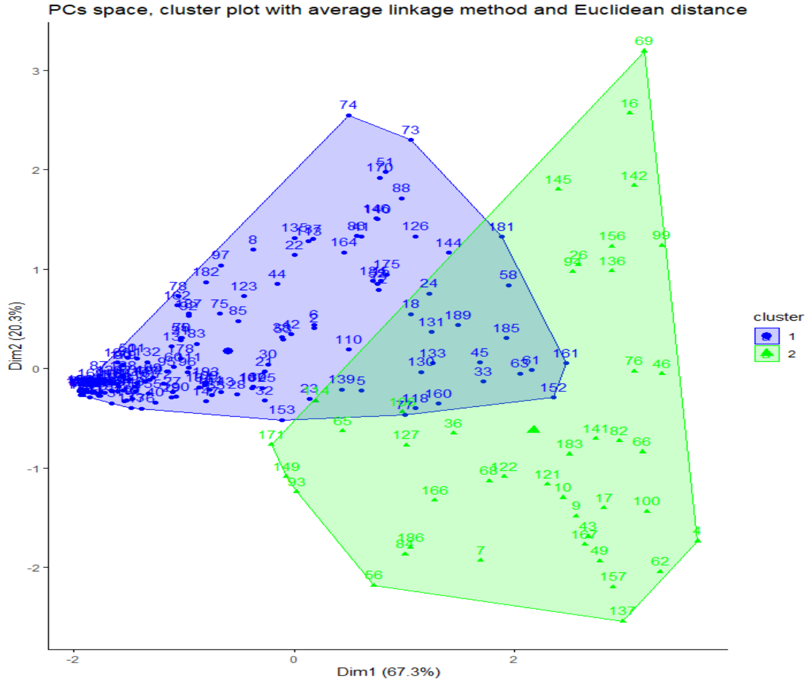
Based on the percentage, the image above gives us information about the distribution of the units in the dendrogram. After that, we are going to compute the scatterplot trough the average linkage method and Euclidean distance.

```
> pairs(df,gap=0,main="Scatterplot matrix with average lin
kage method and Euclidean distance K=2", pch=21, bg=c("blu
e","green")[group2])
```


Scatterplot matrix with average linkage method and Euclidean distance

In the following code, there is the result of the first two PCs.

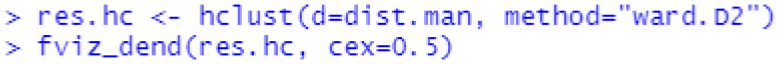
```
> fviz_cluster(list(data=df, cluster=group2), palette=c("b
lue","green"),ellipse.type="convex",main="PCs space, clust
er plot with average linkage method and Euclidean distanc
e",repel=FALSE,ggtheme=theme_classic())
```


PCs space, cluster plot with average linkage method and Euclidean distance

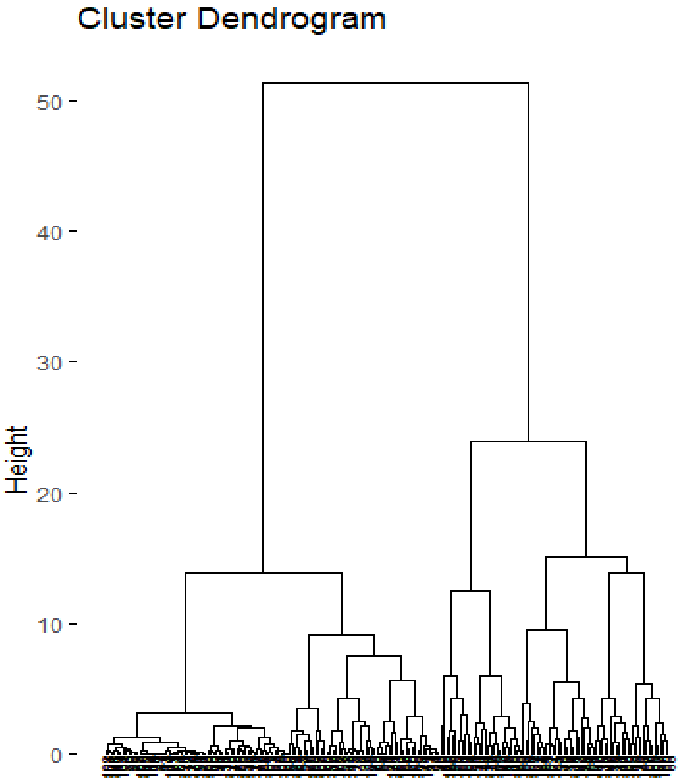
Take a look at the image above, we can affirm that the two clusters are not well separated.

1.5 AHC based on Ward's linkage method and Manhattan distance

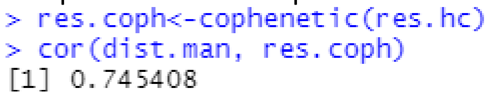
With the following code, there will be the computation of dendrogram, this time with the Ward's linkage method including the Manhattan distance.



```
> res.hc <- hclust(d=dist.man, method="ward.D2")
> fviz_dend(res.hc, cex=0.5)
```



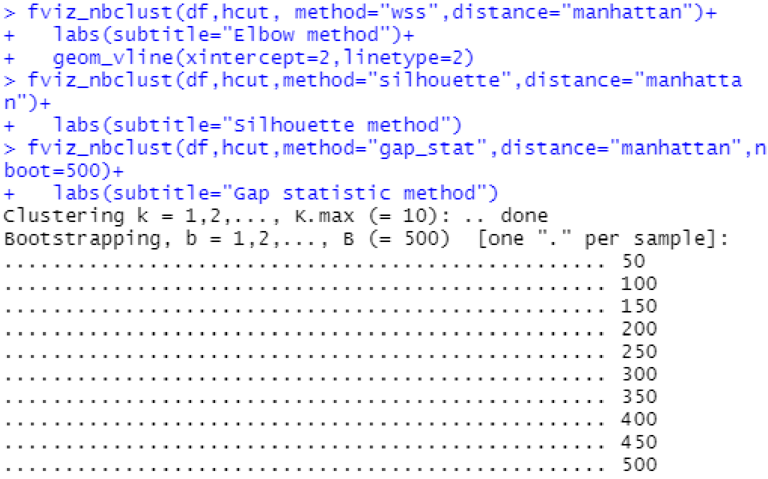
Computation of the cophenetic distance:



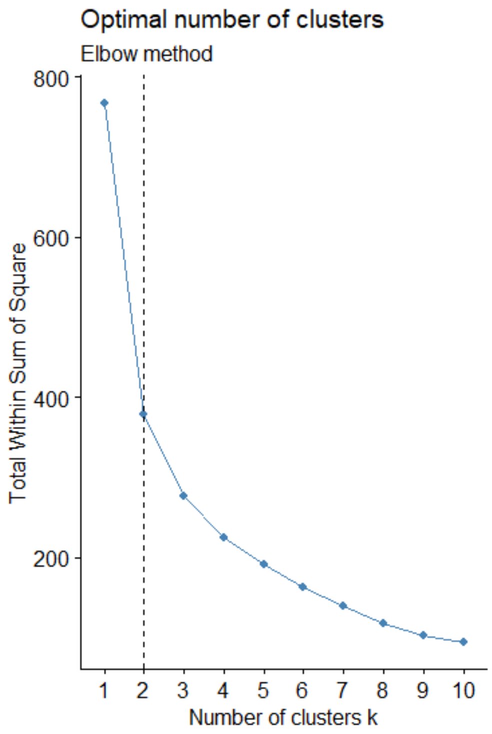
```
> res.coph<-cophenetic(res.hc)
> cor(dist.man, res.coph)
[1] 0.745408
```

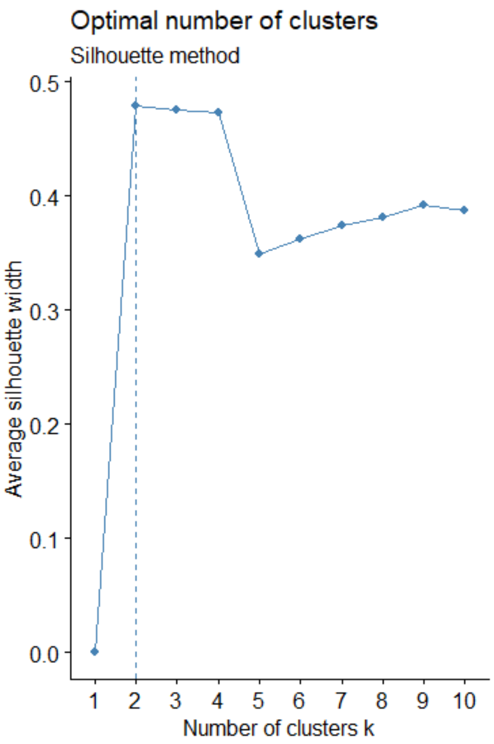
The value (0.745408) is close to 0.75, so we can affirm that Ward's method preserves the original distance quite well.

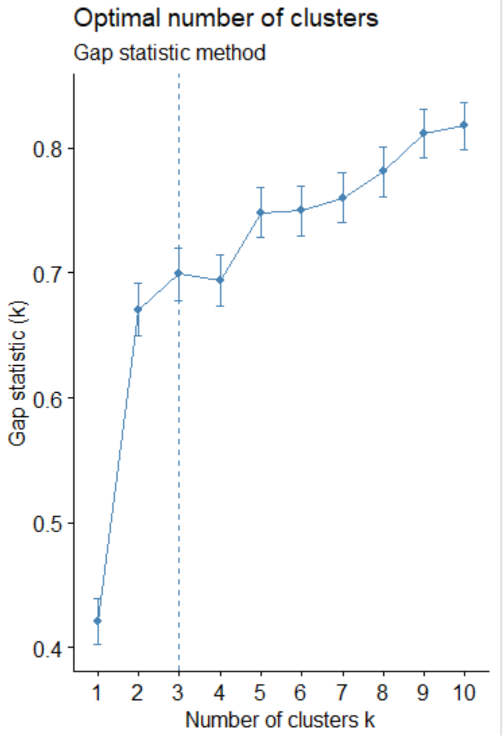
Optimal number of clusters K:



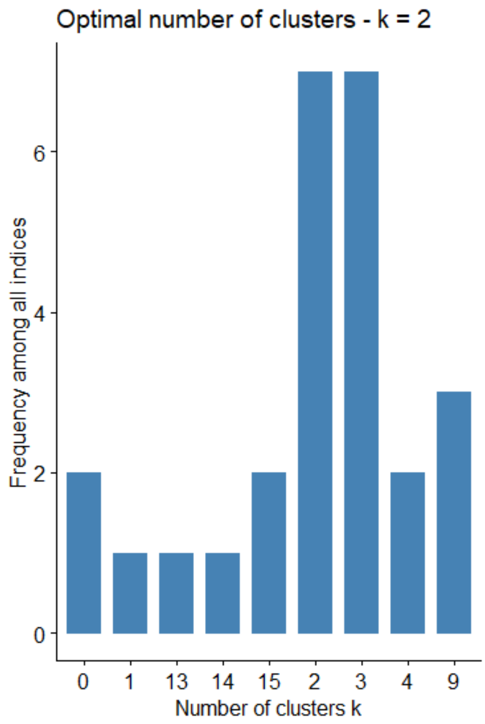
```
> fviz_nbclust(df,hcut, method="wss",distance="manhattan")+
+   labs(subtitle="Elbow method")+
+   geom_vline(xintercept=2,linetype=2)
> fviz_nbclust(df,hcut,method="silhouette",distance="manhatta
n")+
+   labs(subtitle="Silhouette method")
> fviz_nbclust(df,hcut,method="gap_stat",distance="manhattan",n
boot=500)+
+   labs(subtitle="Gap statistic method")
Clustering k = 1,2,..., K.max (= 10): .. done
Bootstrapping, b = 1,2,..., B (= 500)  [one "." per sample]:
.................................................. 50
.................................................. 100
.................................................. 150
.................................................. 200
.................................................. 250
.................................................. 300
.................................................. 350
.................................................. 400
.................................................. 450
.................................................. 500
```


Optimal number of clusters
Elbow method
Total Within Sum of Square
800
600
400
200
1 2 3 4 5 6 7 8 9 10
Number of clusters k


Optimal number of clusters
Silhouette method
Average silhouette width
0.5
0.4
0.3
0.2
0.1
0.0
1 2 3 4 5 6 7 8 9 10
Number of clusters k

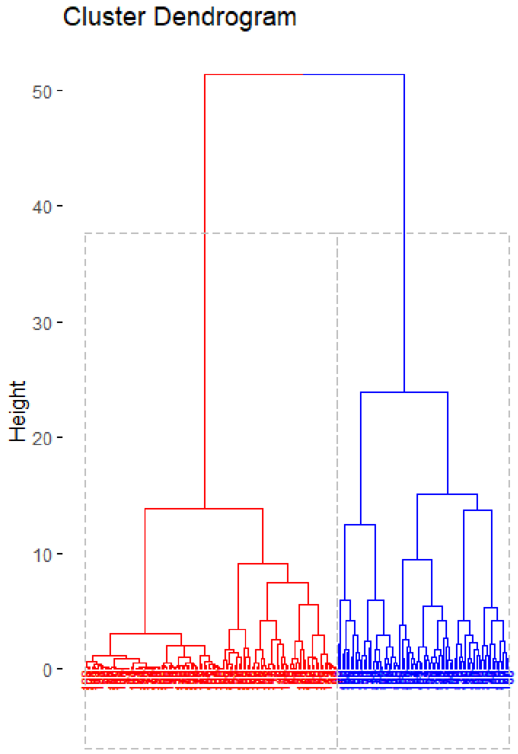

Optimal number of clusters
Gap statistic method
Gap statistic (k)
0.8
0.7
0.6
0.5
0.4
1 2 3 4 5 6 7 8 9 10
Number of clusters k

```
> nb<-NbClust(df,diss=NULL,method="ward.D2",distance="manhatta
n")
*** : The Hubert index is a graphical method of determining the
 number of clusters.
                In the plot of Hubert index, we seek a signific
ant knee that corresponds to a
                significant increase of the value of the measur
e i.e the significant peak in Hubert
                index second differences plot.

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

*******************************************************************
****
* Among all indices:

* 7 proposed 2 as the best number of clusters
* 7 proposed 3 as the best number of clusters
* 2 proposed 4 as the best number of clusters
* 3 proposed 9 as the best number of clusters
* 1 proposed 13 as the best number of clusters
* 1 proposed 14 as the best number of clusters
* 2 proposed 15 as the best number of clusters

                   ***** Conclusion *****


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


*******************************************************************
****
> fviz_nbclust(nb)
```



According to the results, the optimal number of clusters is K=2.

```
> group<-cutree(res.hc, k=2)
> fviz_dend(res.hc, k=2, cex=0.5, k_colors = c("red",
+         "blue"), color_labels_by_k = TRUE,
+         rect=TRUE)
```



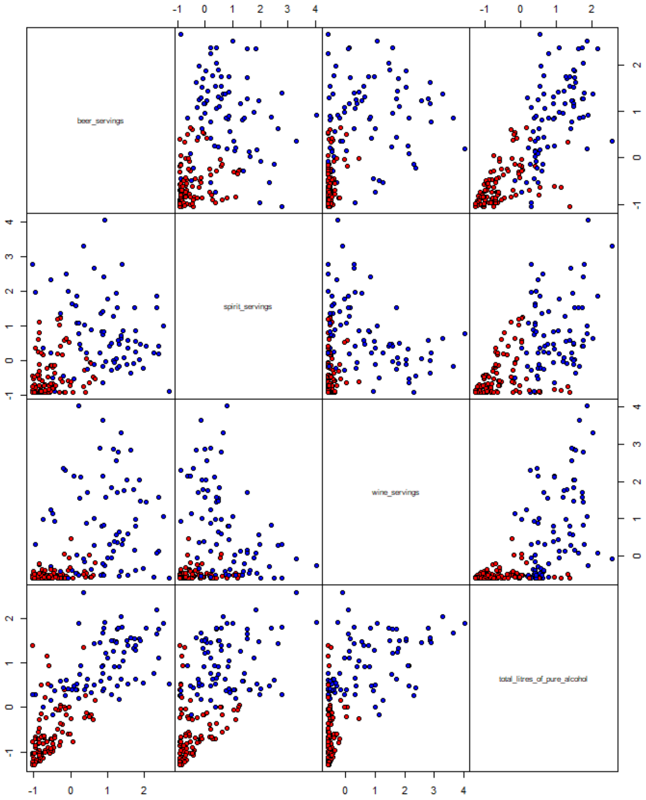
```
> table(group)
group
  1   2 
115  78 
```

By the output, we can attest that the cluster in red is the biggest, with 115 units; the second cluster in blue has 78 units.

Now, we can compute the cluster into the original space and then in PCs space.

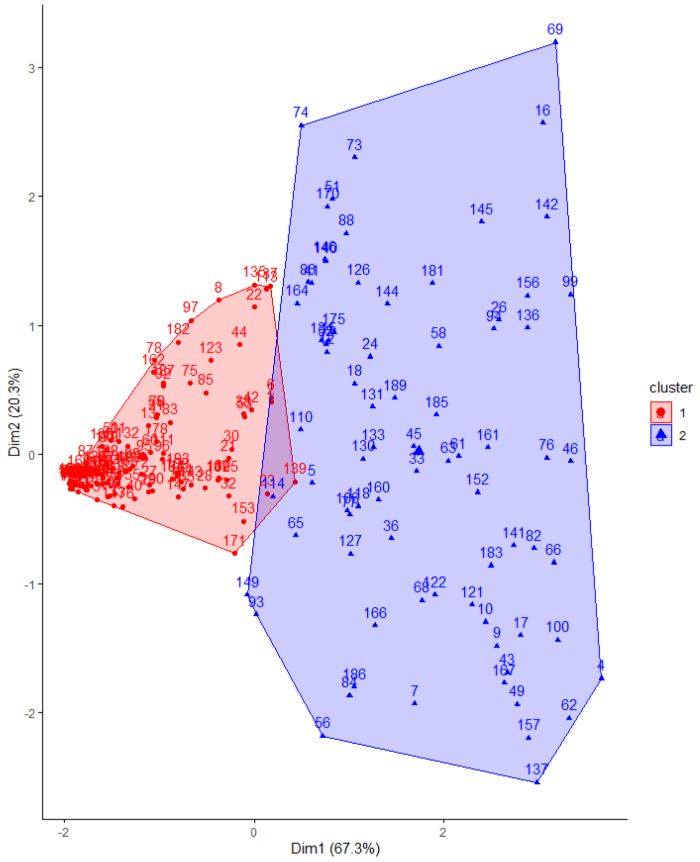
```
> pairs(df,gap=0,main="Scatterplot matrix with ward's linkage m
ethod and manhattan distance K=2", pch=21, bg=c("red","blue")[g
roup])
```

Scatterplot matrix with ward's linkage method and manhattan distance K=2



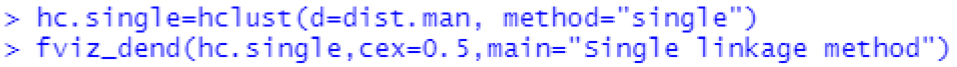
```
> fviz_cluster(list(data=df, cluster=group), palette=c("red","b
lue"),ellipse.type="convex",main="PCs space, cluster plot with
 ward's linkage method and manhattan distance",repel=FALSE,ggth
eme=theme_classic())
```

PCs space, cluster plot with ward's linkage method and manhattan distance

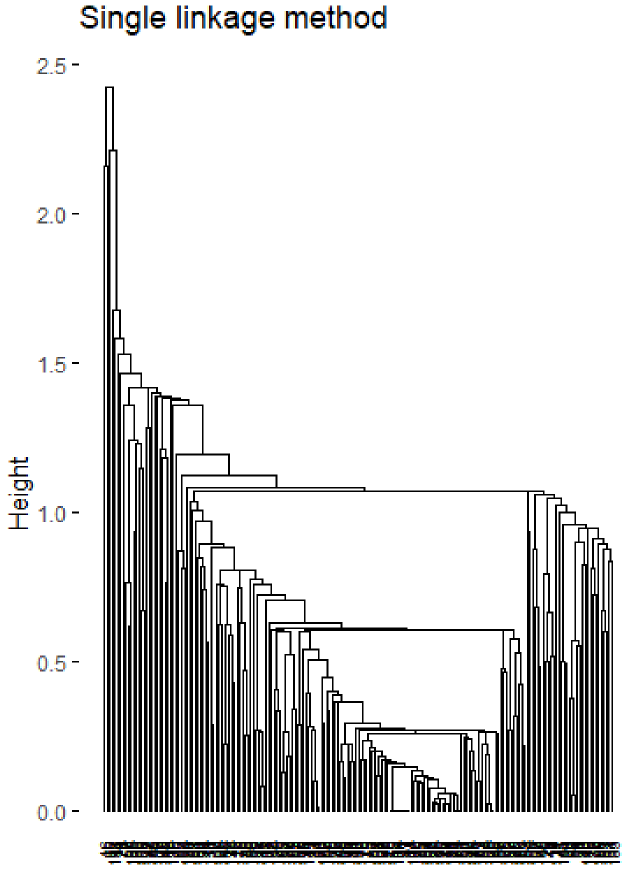


1.6 AHC based on single linkage method and Manhattan distance

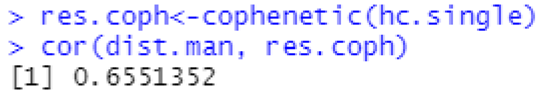
With the following code, there will be the computation of dendrogram, this time with the single linkage method including the Manhattan distance.



```
> hc.single=hclust(d=dist.man, method="single")
> fviz_dend(hc.single,cex=0.5,main="Single linkage method")
```



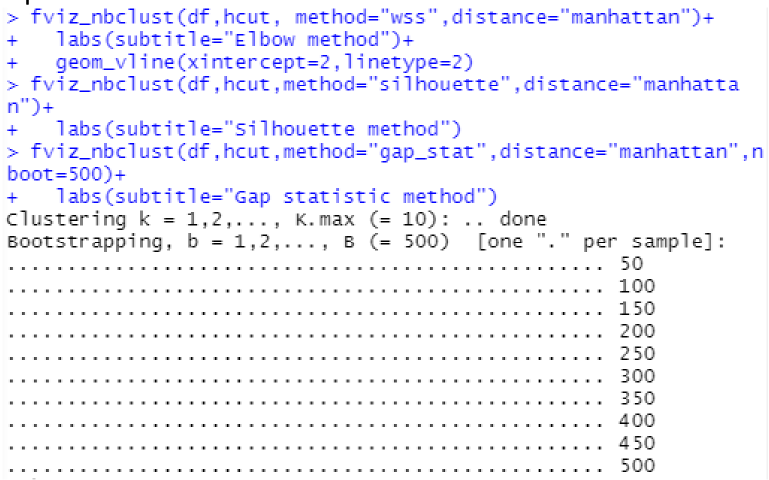
Next the computation of the cophenetic distance:



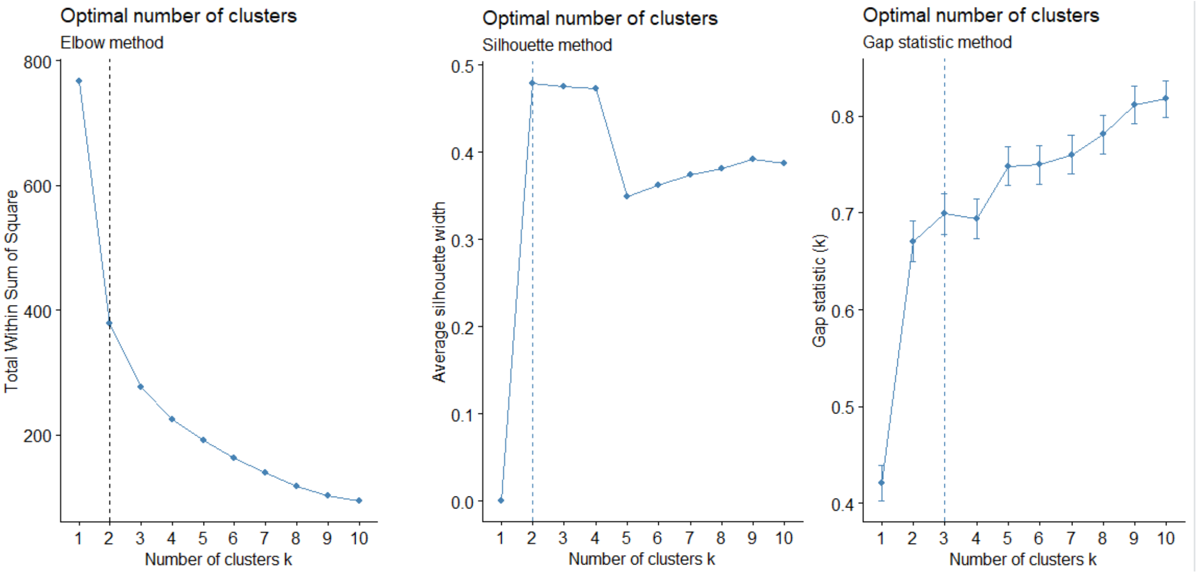
```
> res.coph<-cophenetic(hc.single)
> cor(dist.man, res.coph)
 [1] 0.6551352
```

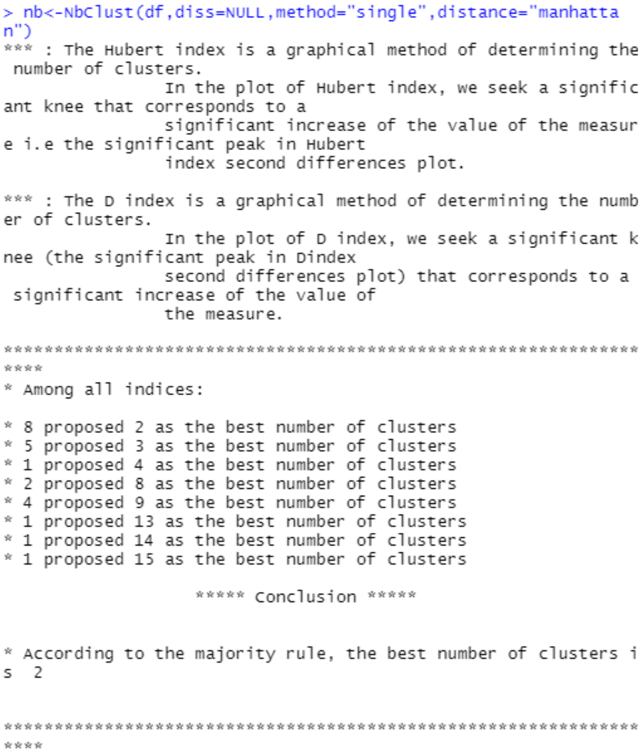
The value (0.6551352) is not good, because is far from the threshold value 0.75. So, we can affirm that the single linkage method with Manhattan distance doesn't preserves the original distance.

Optimal number of clusters K:



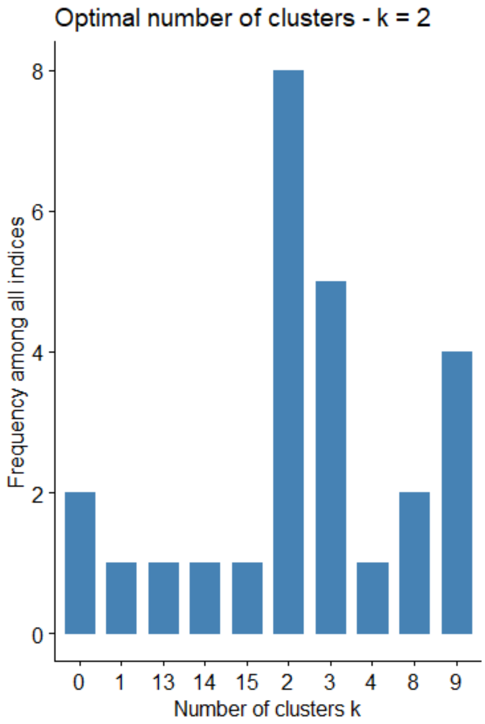
```
> fviz_nbclust(df,hcut, method="wss",distance="manhattan")+
+   labs(subtitle="Elbow method")+
+   geom_vline(xintercept=2,linetype=2)
> fviz_nbclust(df,hcut,method="silhouette",distance="manhatta
n")+
+   labs(subtitle="Silhouette method")
> fviz_nbclust(df,hcut,method="gap_stat",distance="manhattan",n
boot=500)+
+   labs(subtitle="Gap statistic method")
Clustering k = 1,2,..., K.max (= 10): .. done
Bootstrapping, b = 1,2,..., B (= 500)  [one "." per sample]:
.................................................. 50
.................................................. 100
.................................................. 150
.................................................. 200
.................................................. 250
.................................................. 300
.................................................. 350
.................................................. 400
.................................................. 450
.................................................. 500
```



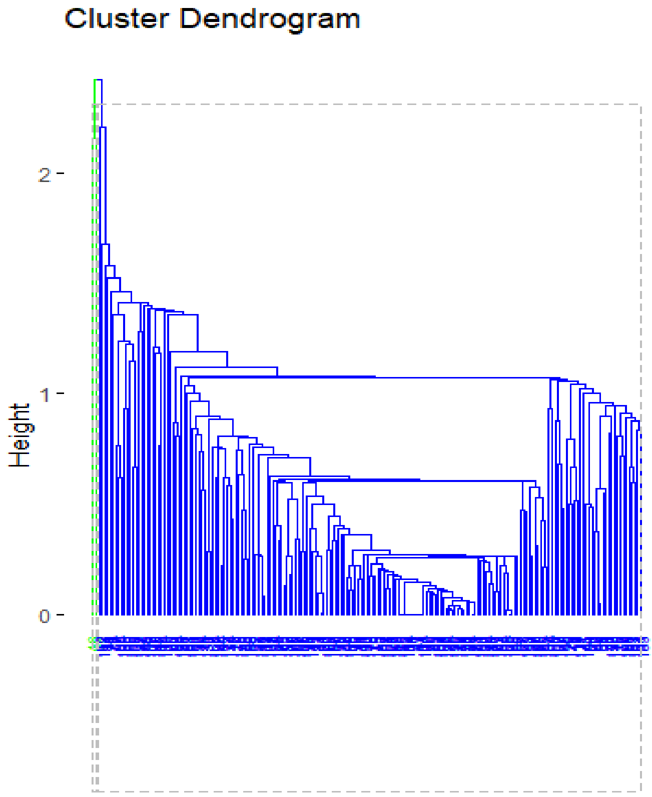


```
> nb<-NbClust(df,diss=NULL,method="single",distance="manhatta
n")
*** : The Hubert index is a graphical method of determining the
 number of clusters.
                In the plot of Hubert index, we seek a signific
ant knee that corresponds to a
                significant increase of the value of the measur
e i.e the significant peak in Hubert
                index second differences plot.

*** : The D index is a graphical method of determining the numb
er of clusters.
                In the plot of D index, we seek a significant k
nee (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
* 1 proposed 4 as the best number of clusters
* 2 proposed 8 as the best number of clusters
* 4 proposed 9 as the best number of clusters
* 1 proposed 13 as the best number of clusters
* 1 proposed 14 as the best number of clusters
* 1 proposed 15 as the best number of clusters

                   ***** Conclusion *****


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

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

We can establish, as above, an optimal number of clusters K=2.

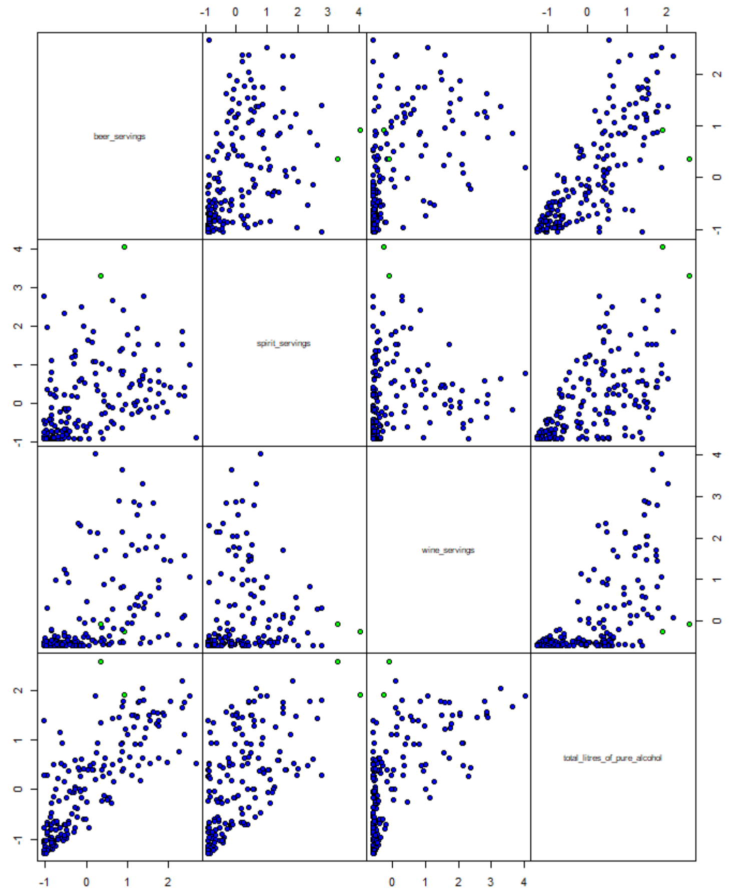


```
> group<-cutree(hc.single, k=2)
> fviz_dend(hc.single, k=2, cex=0.5, k_colors = c("green",
+         "blue"), color_labels_by_k = TRUE,
+         rect=TRUE)
```



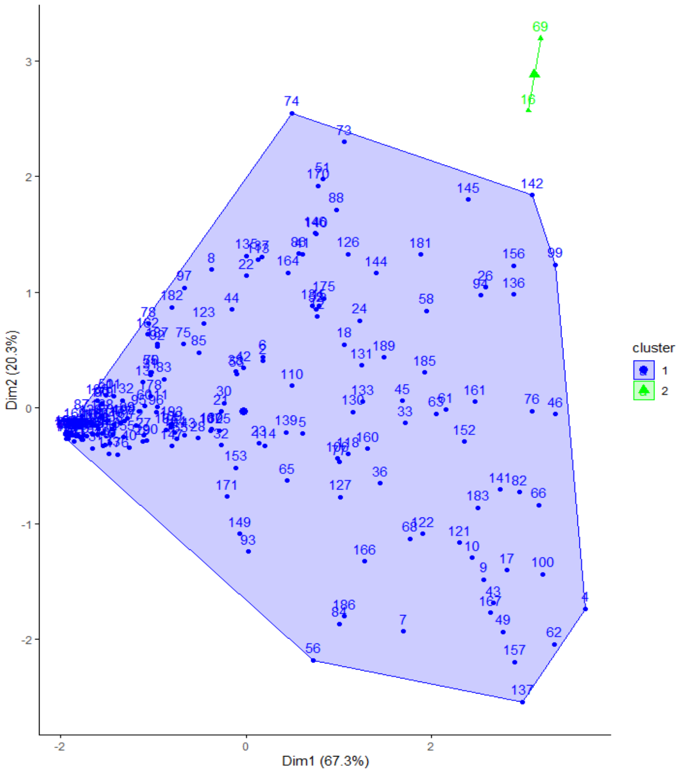
```
> table(group)
group
  1   2 
191   2 
> pairs(df,gap=0,main="Scatterplot matrix with single linkage m
ethod and manhattan distance K=2", pch=21, bg=c("blue","green")
[group])
```

Scatterplot matrix with single linkage method and manhattan distance K=2



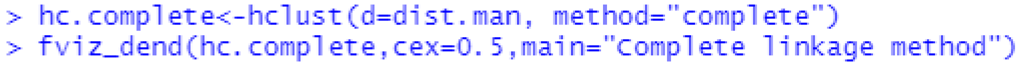
```
> fviz_cluster(list(data=df, cluster=group), palette=c("blu
e","green"),ellipse.type="convex",main="PCs space, cluster plot
 with single linkage method and manhattan distance",repel=FALS
E,ggtheme=theme_classic())
```

PCs space, cluster plot with single linkage method and manhattan distance

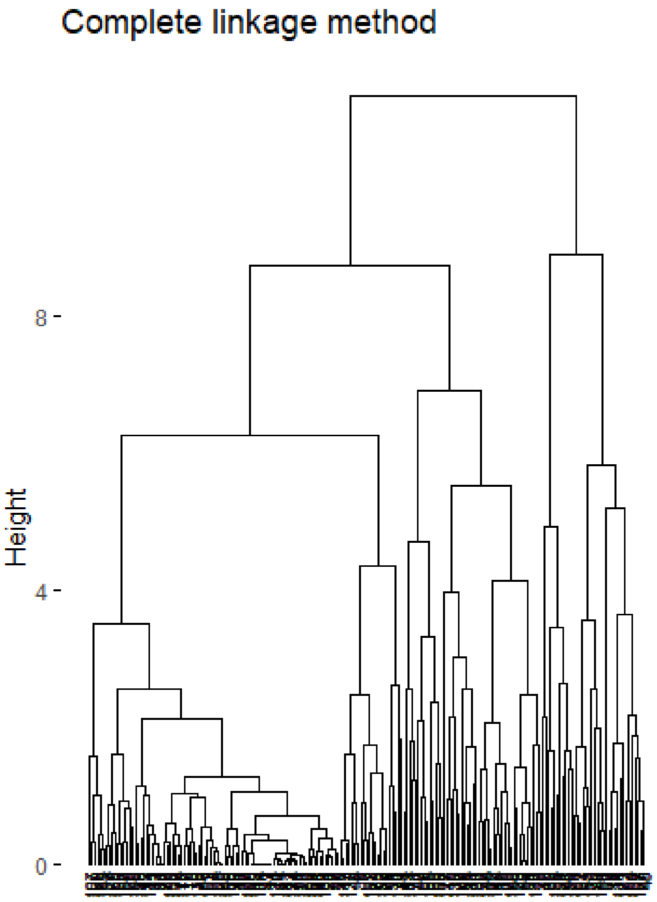


1.7 AHC based on complete linkage method and Manhattan distance

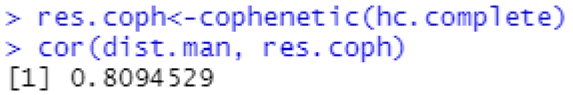
With the following code, there will be the computation of dendrogram, this time with the complete linkage method including the Manhattan distance.



```
> hc.complete<-hclust(d=dist.man, method="complete")
> fviz_dend(hc.complete,cex=0.5,main="Complete linkage method")
```



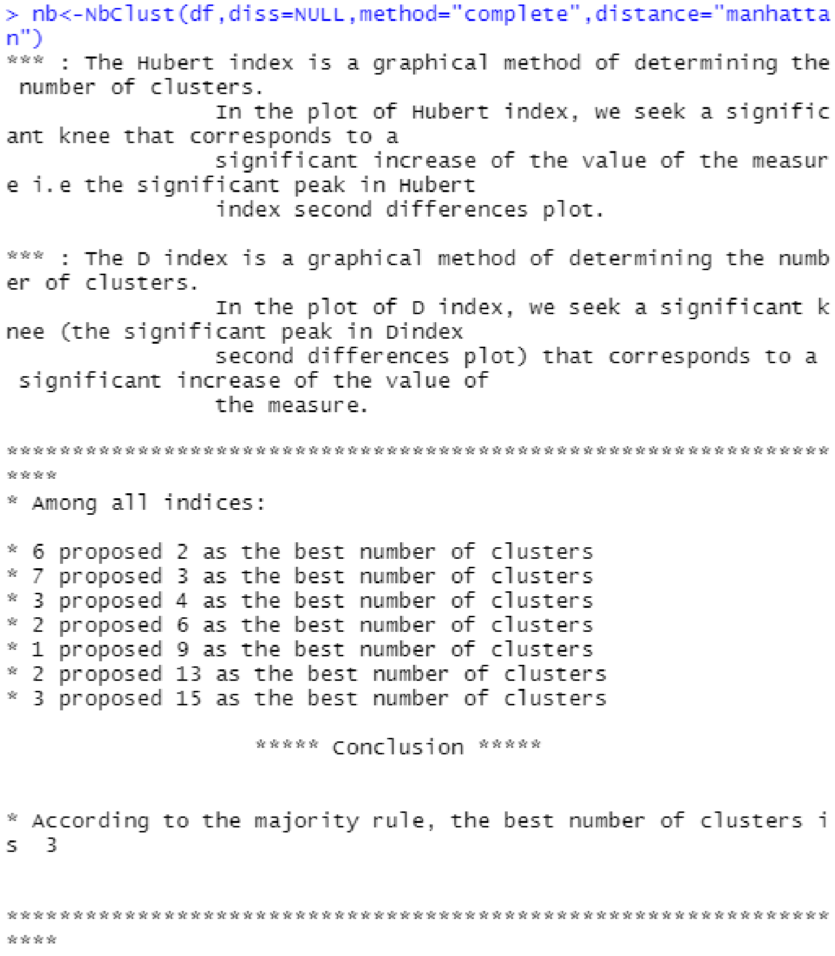
Cophenetic distance below:



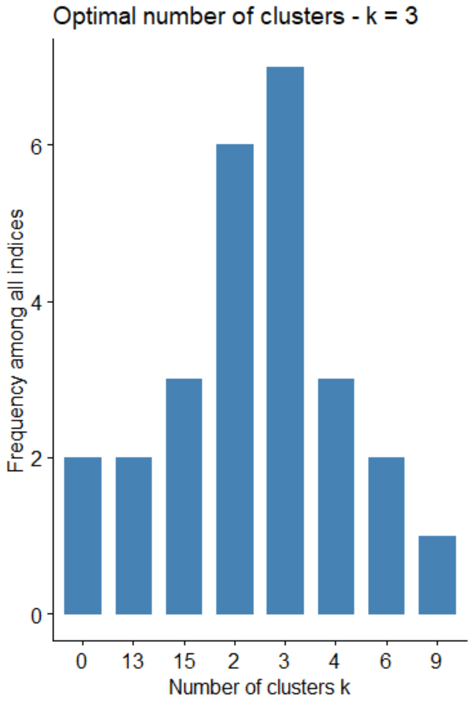
```
> res.coph<-cophenetic(hc.complete)
> cor(dist.man, res.coph)
[1] 0.8094529
```

The value (0.8094529) comes out from the computation is a good one because is greater than 0.75, so we can attest that the complete linkage method with the Manhattan distance preserves the original distance between units.

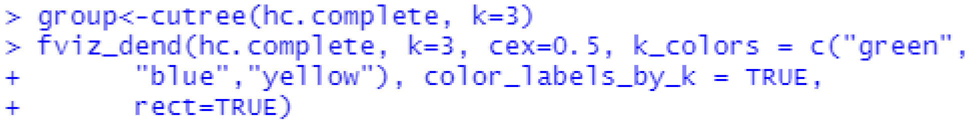
Optimal number of clusters K:



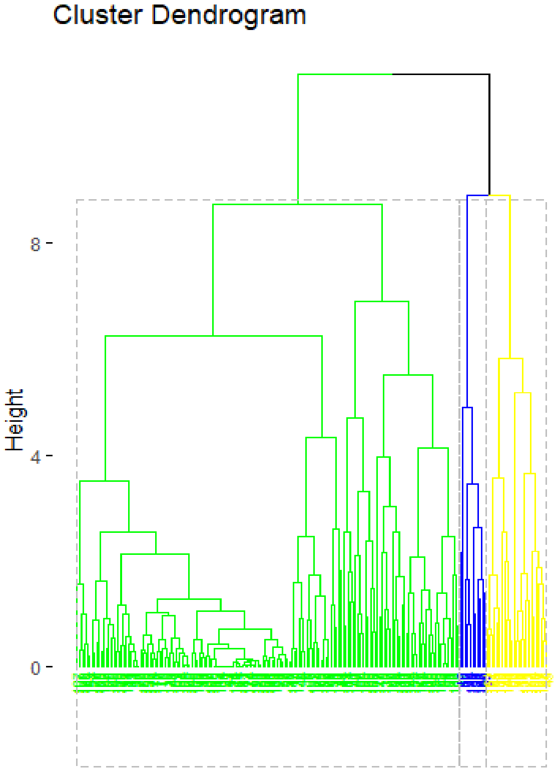
```
> nb<-NbClust(df,diss=NULL,method="complete",distance="manhatta
n")
*** : The Hubert index is a graphical method of determining the
 number of clusters.
                In the plot of Hubert index, we seek a signific
ant knee that corresponds to a
                significant increase of the value of the measur
e i.e the significant peak in Hubert
                index second differences plot.

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

***************************************************************
****
* Among all indices:

* 6 proposed 2 as the best number of clusters
* 7 proposed 3 as the best number of clusters
* 3 proposed 4 as the best number of clusters
* 2 proposed 6 as the best number of clusters
* 1 proposed 9 as the best number of clusters
* 2 proposed 13 as the best number of clusters
* 3 proposed 15 as the best number of clusters

                  ***** Conclusion *****


* According to the majority rule, the best number of clusters i
s  3


***************************************************************
****
> fviz_nbclust(nb)
```

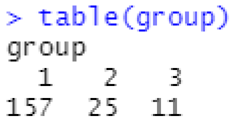


According to the results, we can establish an optimal number of clusters K=3. Next, we'll cut the dendrogram.



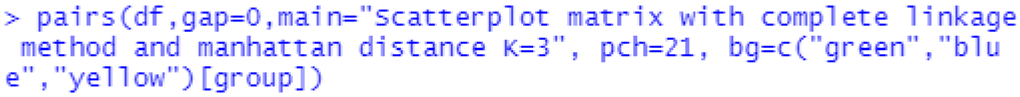
```
> group<-cutree(hc.complete, k=3)
> fviz_dend(hc.complete, k=3, cex=0.5, k_colors = c("green",
+        "blue","yellow"), color_labels_by_k = TRUE,
+        rect=TRUE)
```





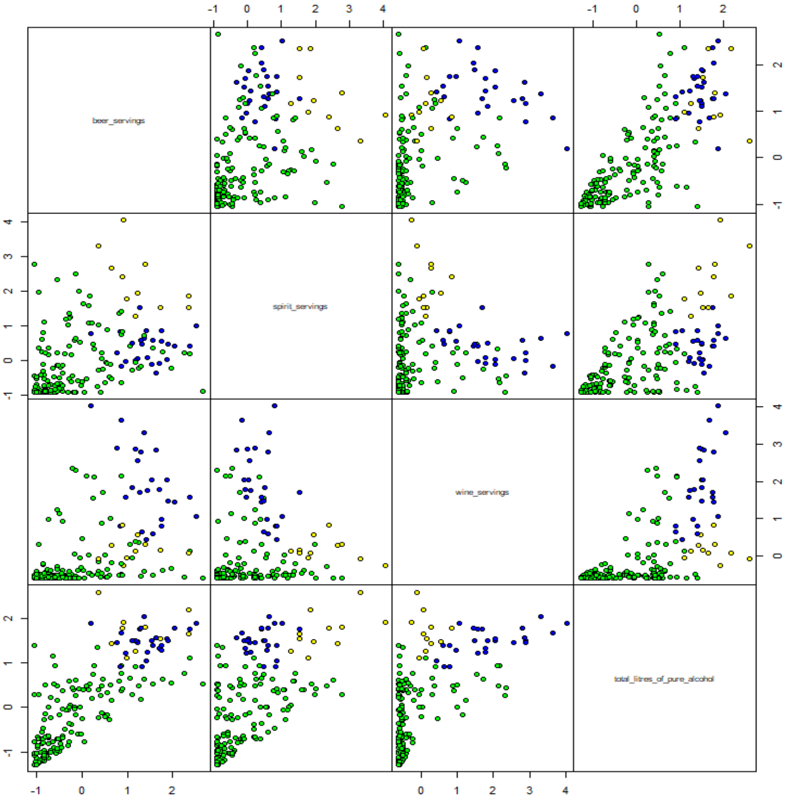
```
> table(group)
group
  1   2   3 
157  25  11 
```

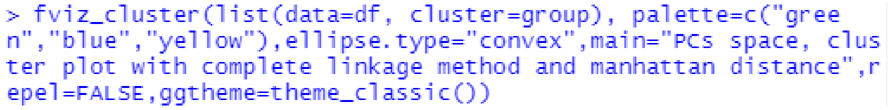
By looking at the image, we can see that the cluster in green is the biggest, with 157 observations. Now we'll plot the clusters in the original space and then in PCs space.



```
> pairs(df,gap=0,main="Scatterplot matrix with complete linkage
 method and manhattan distance K=3", pch=21, bg=c("green","blu
e","yellow")[group])
```

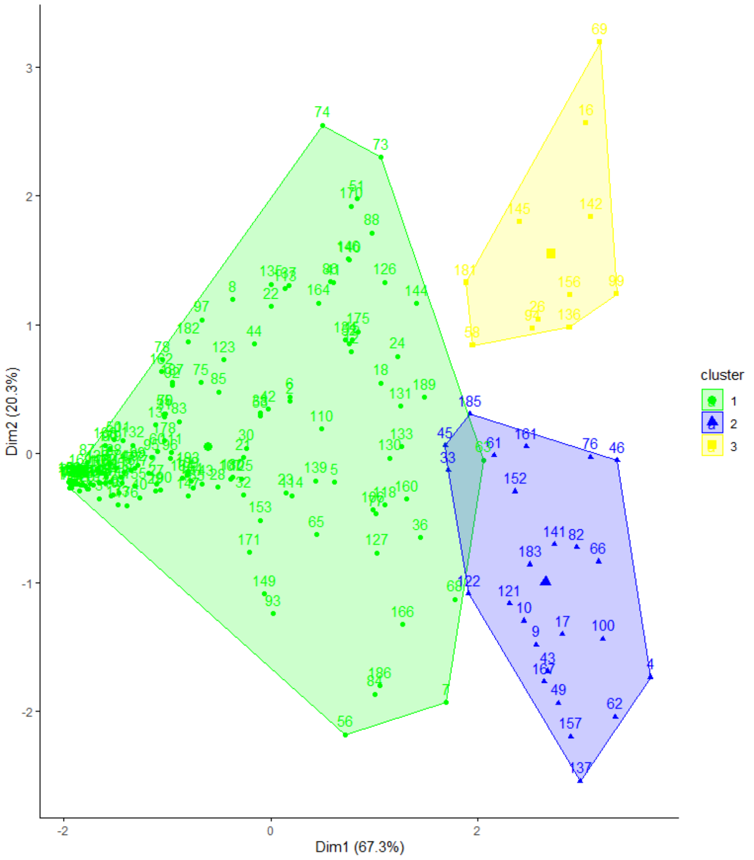
Scatterplot matrix with complete linkage method and manhattan distance K=3





```
> fviz_cluster(list(data=df, cluster=group), palette=c("gree
n","blue","yellow"),ellipse.type="convex",main="PCs space, clus
ter plot with complete linkage method and manhattan distance",r
epel=FALSE,ggtheme=theme_classic())
```

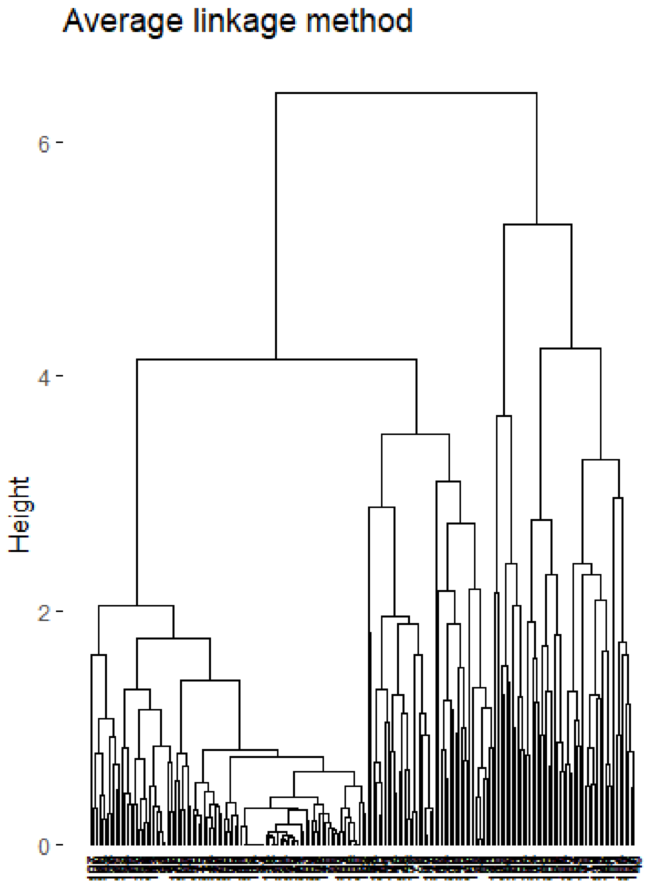
PCs space, cluster plot with complete linkage method and manhattan distance



1.8 AHC based on average linkage method and Manhattan distance

With the following code, there will be the computation of dendrogram, this time with the average linkage method including the Manhattan distance.

```
> res.hc <- hclust(d=dist.man, method="average")
> fviz_dend(res.hc, cex=0.5, main="Average linkage method")
```



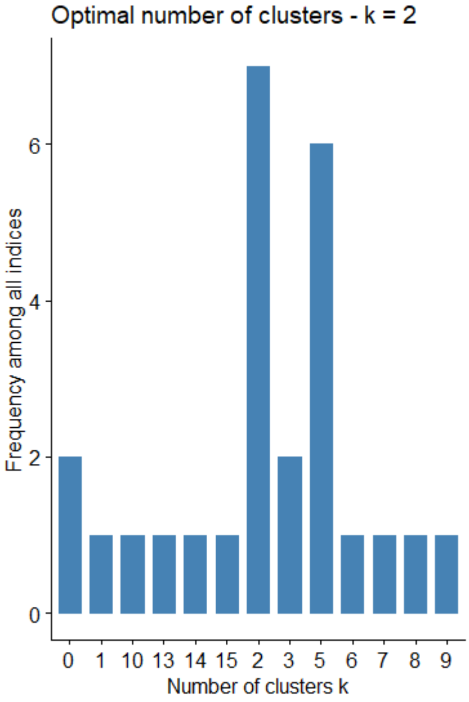
Cophenetic distance:

```
> res.coph<-cophenetic(res.hc)
> cor(dist.man, res.coph)
[1] 0.8093436
```

As the results (0.8093436) say, the value is good because is up to 0.75, so we can say that the average linkage method with Manhattan distance preserves the original distance.

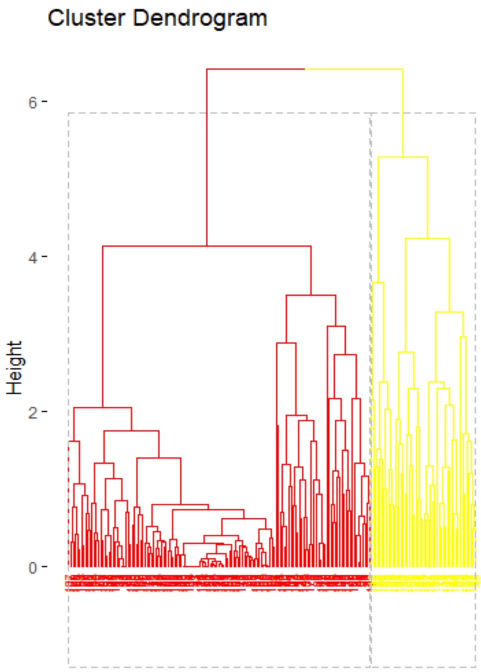
Optimal number of clusters K:

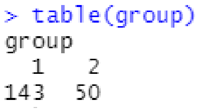
```
> nb<-NbClust(df,diss=NULL,method="average",distance="manhatta
n")
*** : The Hubert index is a graphical method of determining the
 number of clusters.
                In the plot of Hubert index, we seek a signific
ant knee that corresponds to a 
                significant increase of the value of the measur
e i.e the significant peak in Hubert
                index second differences plot. 

*** : The D index is a graphical method of determining the numb
er of clusters. 
                In the plot of D index, we seek a significant k
nee (the significant peak in Dindex
                second differences plot) that corresponds to a
 significant increase of the value of
                the measure. 
 
*******************************************************************
****
* Among all indices:

* 7 proposed 2 as the best number of clusters 
* 2 proposed 3 as the best number of clusters 
* 6 proposed 5 as the best number of clusters 
* 1 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 9 as the best number of clusters 
* 1 proposed 10 as the best number of clusters 
* 1 proposed 13 as the best number of clusters 
* 1 proposed 14 as the best number of clusters 
* 1 proposed 15 as the best number of clusters 

                   ***** Conclusion *****                            
 

* According to the majority rule, the best number of clusters i
s  2 
 
 
*******************************************************************
****
> fviz_nbclust(nb)
```



According with the suggested result, the optimal number of clusters is K=2.

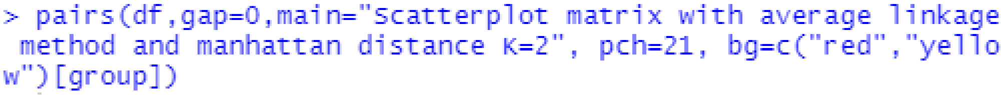
```
> group<-cutree(res.hc, k=2)
> fviz_dend(res.hc, k=2, cex=0.5, k_colors = c("red","yellow"),
 color_labels_by_k = TRUE,
+         rect=TRUE)
```



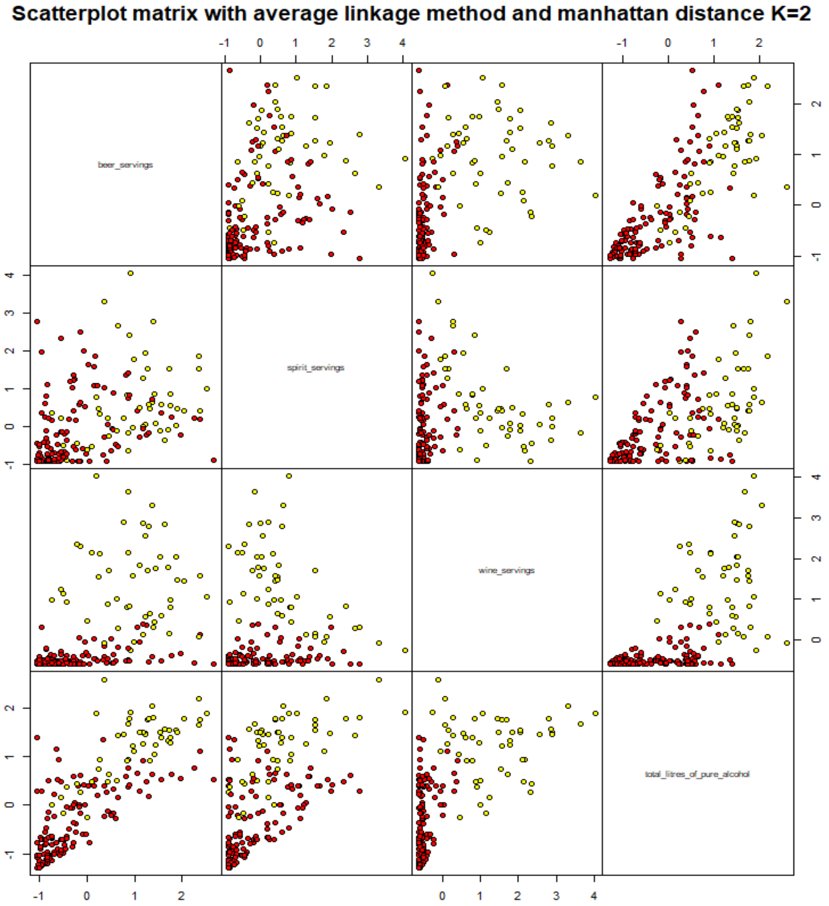


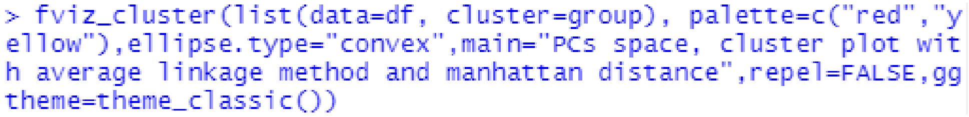
```
> table(group)
group
  1   2 
143  50 
```

We can visualize the clusters in the original space and then in the PCs space.

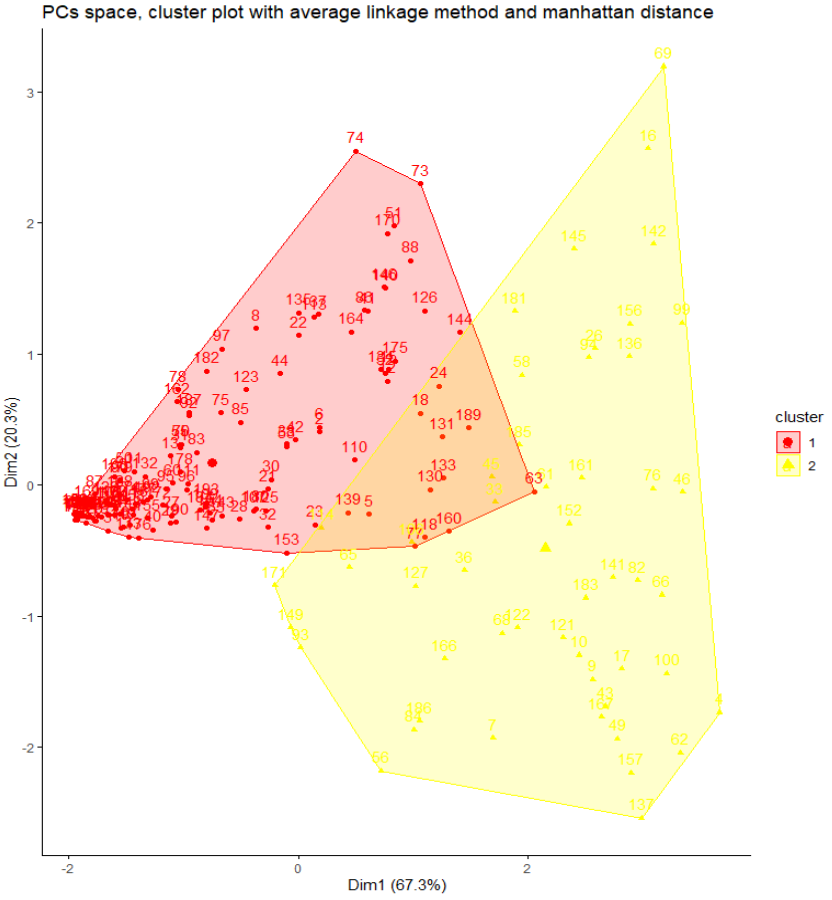


```
> pairs(df,gap=0,main="Scatterplot matrix with average linkage
 method and manhattan distance K=2", pch=21, bg=c("red","yello
w")[group])
```





```
> fviz_cluster(list(data=df, cluster=group), palette=c("red","y
ellow"),ellipse.type="convex",main="PCs space, cluster plot wit
h average linkage method and manhattan distance",repel=FALSE,gg
theme=theme_classic())
```



CLUSTER VALIDATION STATISTICS

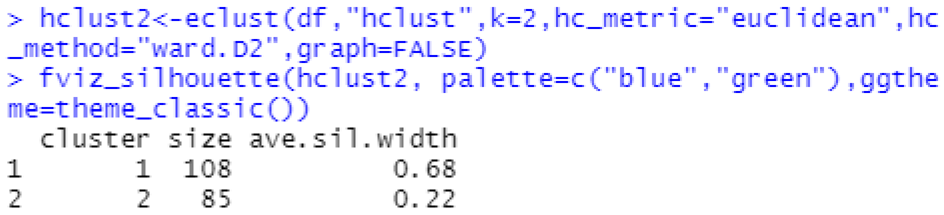
The cluster validation statistics is a procedure that aims to evaluate and measure the goodness of clustering results. There are two main types of clustering validation:

- Internal cluster validation that uses internal information of the clustering process to evaluate the goodness of a clustering structure.
- External cluster validation that uses externally known result in order to compare the result of a cluster analysis.

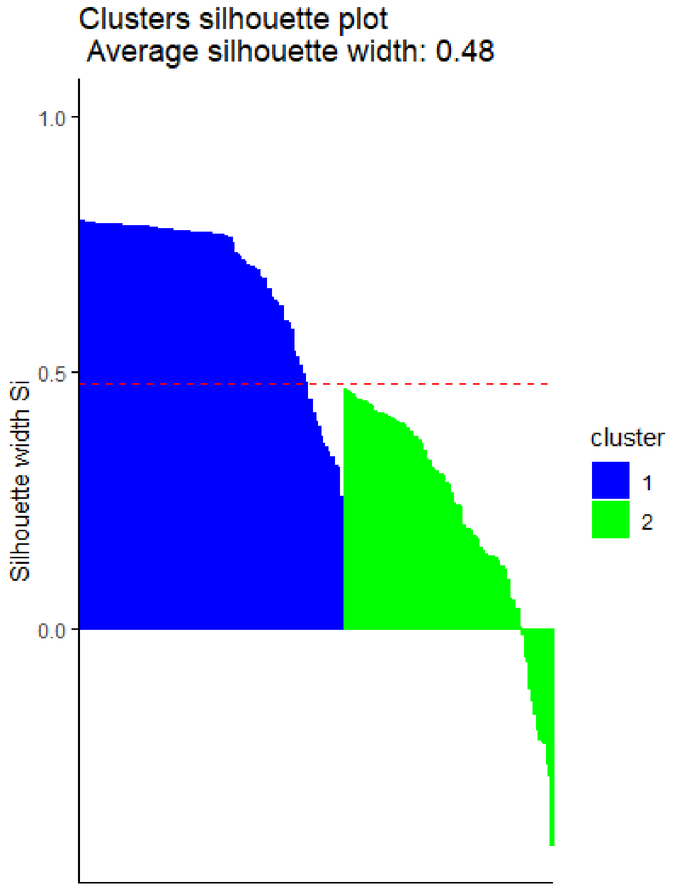
1.1. INTERNAL CLUSTER VALIDATION

The following step consist of evaluate the clustering result, starting from the internal cluster validation of the ***AHC with Ward's linkage method and Euclidean distance***.

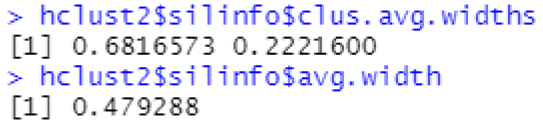
The ***silhouette width*** is an approach that aims to measure the separation and the cohesion of the clusters to be analysed. His values are included in the interval [-1;1]; we can affirm that values close to 1 indicates that units are well clustered, instead if the values are closer to -1 or 0, we can say that the clustering results are not good.



```
> hclust2<-eclust(df,"hclust",k=2,hc_metric="euclidean",hc
_method="ward.D2",graph=FALSE)
> fviz_silhouette(hclust2, palette=c("blue","green"),ggthe
me=theme_classic())
  cluster size ave.sil.width
1       1  108          0.68
2       2   85          0.22
```

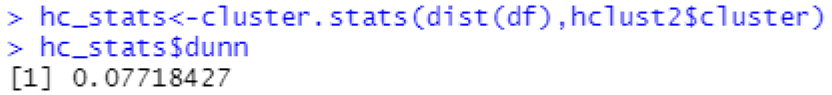


From the silhouette we can extract the Average silhouette widths of both clusters (close to 1 is well clustered):



```
> hclust2$silinfo$clus.avg.widths
[1] 0.6816573 0.2221600
> hclust2$silinfo$avg.width
[1] 0.479288
```

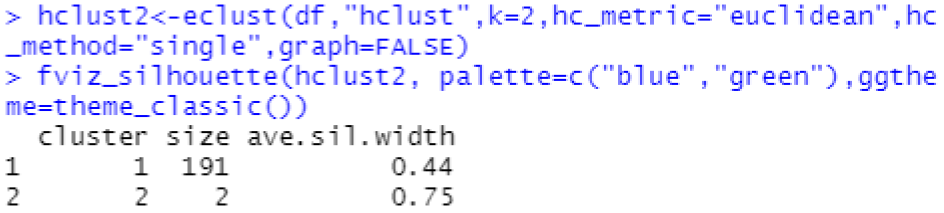
Next to be computed is the ***Dunn index***.
The Dunn index is an internal validation measure. A higher Dunn Index will indicate compact, well-separated clusters, while a lower index will indicate less compact or less well-separated clusters.



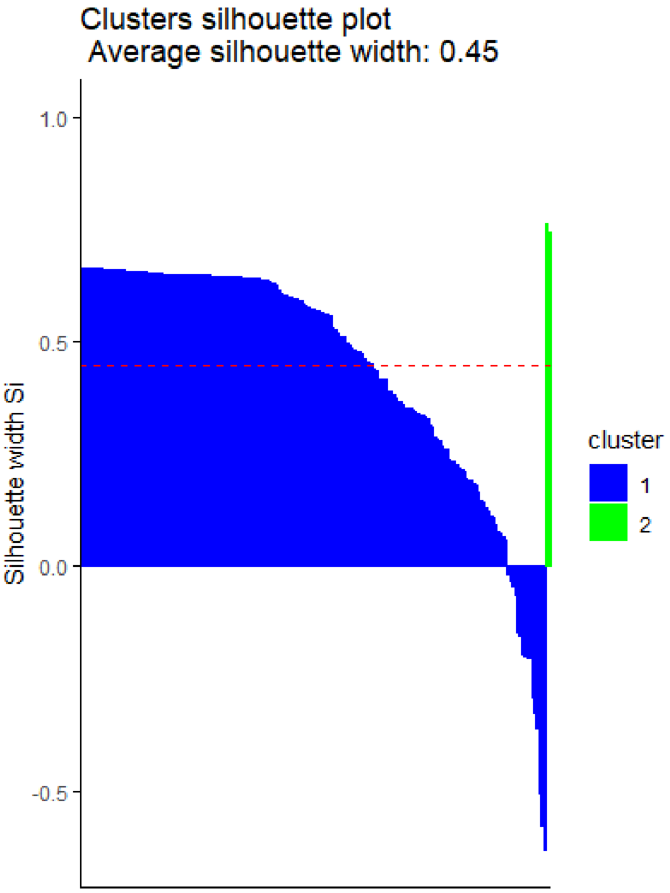
```
> hc_stats<-cluster.stats(dist(df),hclust2$cluster)
> hc_stats$dunn
[1] 0.07718427
```

1.2. INTERNAL CLUSTER VALIDATION

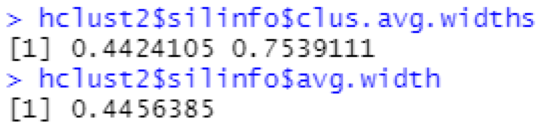
The following step consist of evaluate the clustering result, starting from the internal cluster validation of the ***AHC with single linkage method and Euclidean distance***.
Silhouette width with the following code:



```
> hclust2<-eclust(df,"hclust",k=2,hc_metric="euclidean",hc
_method="single",graph=FALSE)
> fviz_silhouette(hclust2, palette=c("blue","green"),ggthe
me=theme_classic())
  cluster size ave.sil.width
1       1  191          0.44
2       2    2          0.75
```

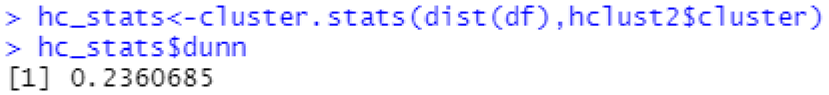


From the silhouette we can extract the Average silhouette widths of both clusters:



```
> hclust2$silinfo$clus.avg.widths
[1] 0.4424105 0.7539111
> hclust2$silinfo$avg.width
[1] 0.4456385
```

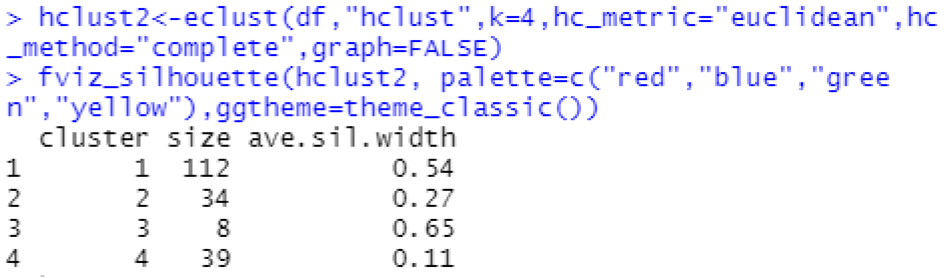
Then the ***Dunn index***:



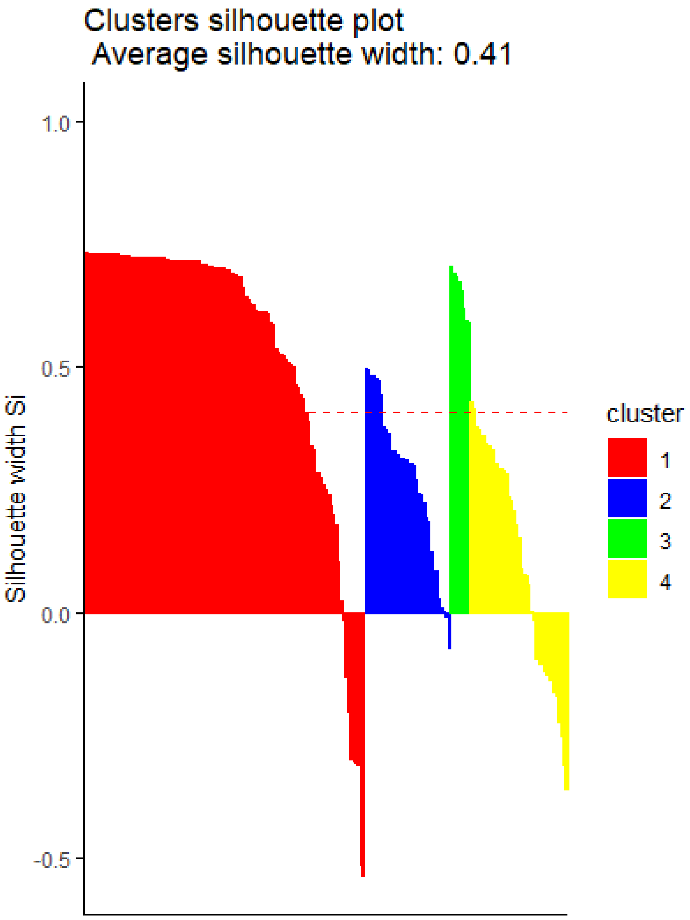
```
> hc_stats<-cluster.stats(dist(df),hclust2$cluster)
> hc_stats$dunn
[1] 0.2360685
```

1.3. INTERNAL CLUSTER VALIDATION

The following step consist of evaluate the clustering result, starting from the internal cluster validation of the ***AHC with complete linkage method and Euclidean distance***.
Going on, the computation of the ***silhouette width***:



```
> hclust2<-eclust(df,"hclust",k=4,hc_metric="euclidean",hc
_method="complete",graph=FALSE)
> fviz_silhouette(hclust2, palette=c("red","blue","gree
n","yellow"),ggtheme=theme_classic())
  cluster size ave.sil.width
1       1  112          0.54
2       2   34          0.27
3       3    8          0.65
4       4   39          0.11
```



From the silhouette we can extract the Average silhouette widths of both clusters:

```
> hclust2$silinfo$clus.avg.widths
[1] 0.5384968 0.2714281 0.6509804 0.1057274
> hclust2$silinfo$avg.width
[1] 0.4086601
```

Next to be computed, the ***Dunn index***:

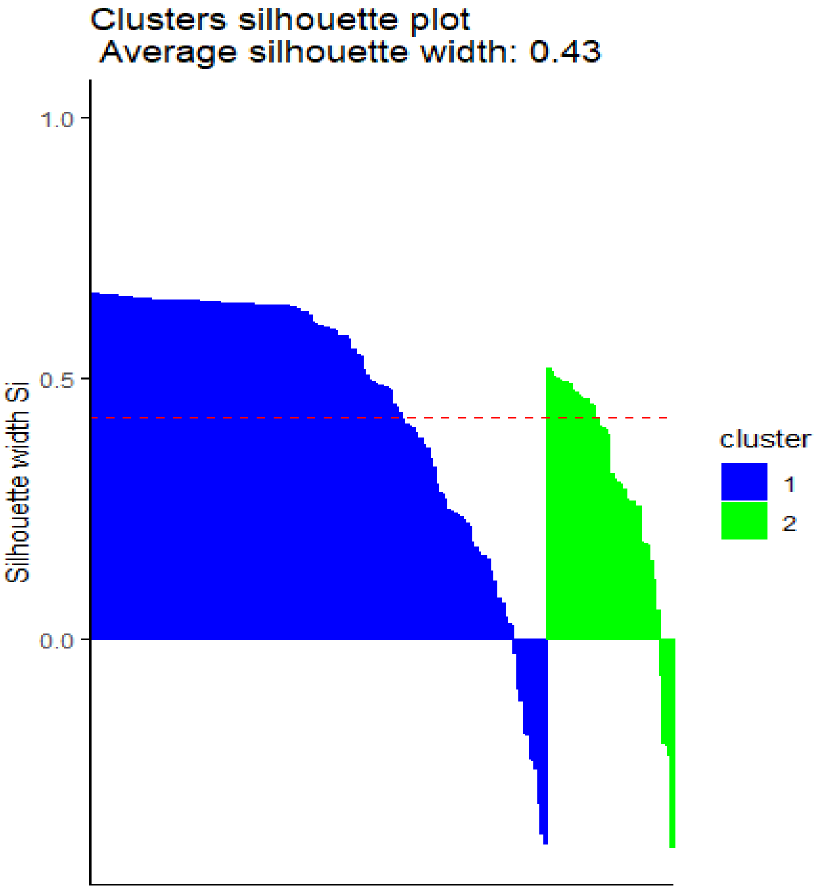
```
> hc_stats<-cluster.stats(dist(df),hclust2$cluster)
> hc_stats$dunn
[1] 0.1109053
```

1.4. INTERNAL CLUSTER VALIDATION

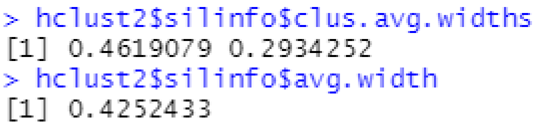
The following step consist of evaluate the clustering result, starting from the internal cluster validation of the ***AHC with average linkage method and Euclidean distance***.

Compute starting from the ***silhouette width***:

```
> hclust2<-eclust(df,"hclust",k=2,hc_metric="euclidean",hc
_method="average",graph=FALSE)
> fviz_silhouette(hclust2, palette=c("blue","green"),ggthe
me=theme_classic())
  cluster size ave.sil.width
1       1  151          0.46
2       2   42          0.29
```

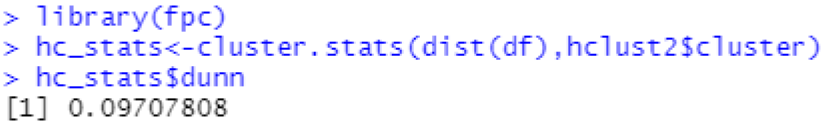


From the silhouette we can extract the Average silhouette widths of both clusters:



```
> hclust2$silinfo$clus.avg.widths
[1] 0.4619079 0.2934252
> hclust2$silinfo$avg.width
[1] 0.4252433
```

Next to be computed is the ***Dunn index***.

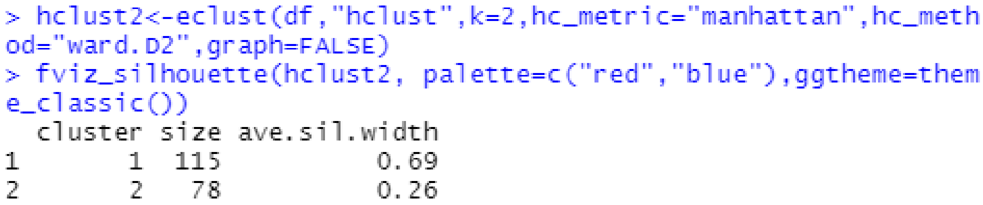


```
> library(fpc)
> hc_stats<-cluster.stats(dist(df),hclust2$cluster)
> hc_stats$dunn
[1] 0.09707808
```

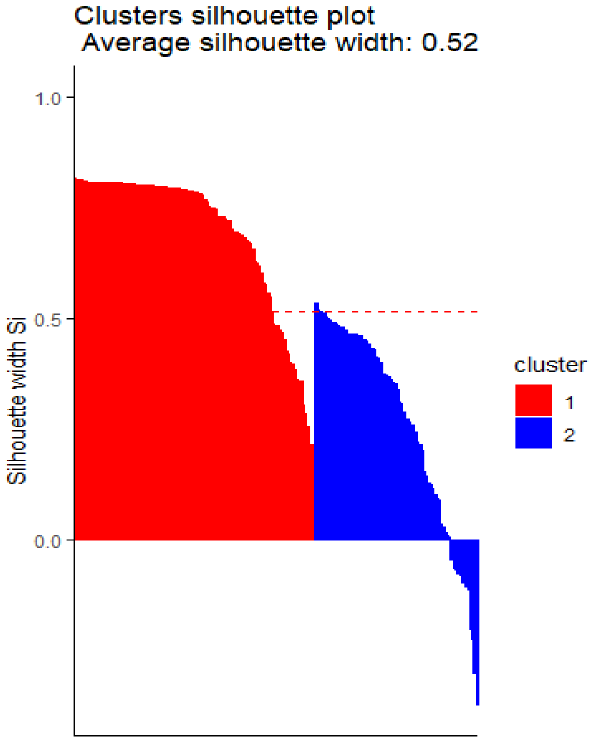
1.5. INTERNAL CLUSTER VALIDATION

The following step consist of evaluate the clustering result, starting from the internal cluster validation of the ***AHC with Ward's linkage method and Manhattan distance***.

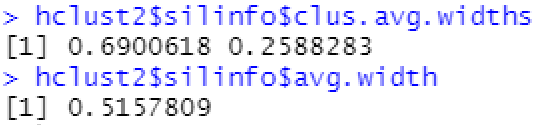
Let's compute the ***silhouette width***:



```
> hclust2<-eclust(df,"hclust",k=2,hc_metric="manhattan",hc_meth
od="ward.D2",graph=FALSE)
> fviz_silhouette(hclust2, palette=c("red","blue"),ggtheme=them
e_classic())
  cluster size ave.sil.width
1       1  115          0.69
2       2   78          0.26
```



From the silhouette we can extract the Average silhouette widths of both clusters:



```
> hclust2$silinfo$clus.avg.widths
[1] 0.6900618 0.2588283
> hclust2$silinfo$avg.width
[1] 0.5157809
```

Then we'll compute the ***Dunn index***.

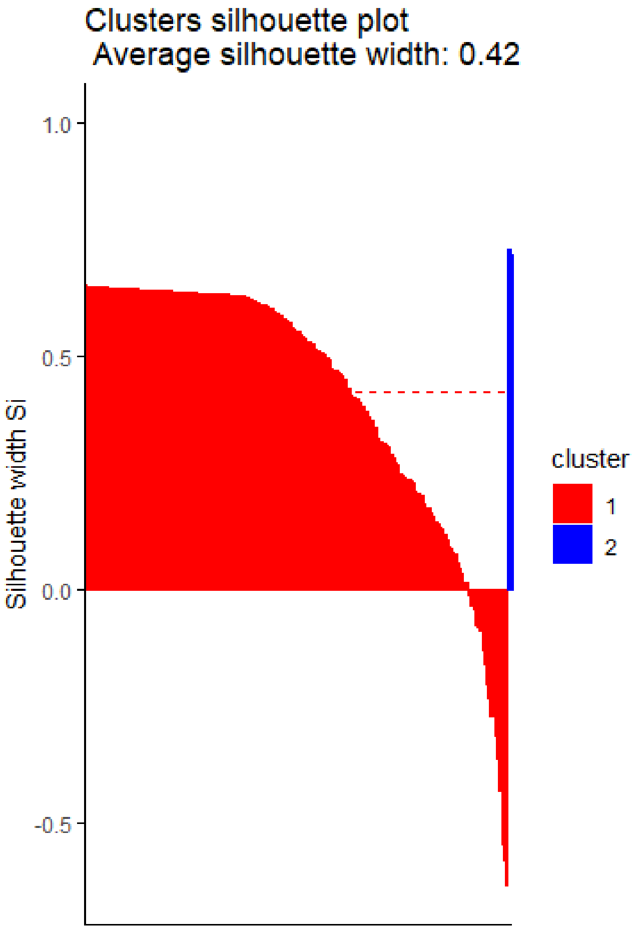
```
> hc_stats<-cluster.stats(dist(df),hclust2$cluster)
> hc_stats$dunn
[1] 0.0904593
```

1.6. INTERNAL CLUSTER VALIDATION

The following step consist of evaluate the clustering result, starting from the internal cluster validation of the ***AHC with single linkage method and Manhattan distance***.

Let's compute the ***silhouette width***:

```
> hclust2<-eclust(df,"hclust",k=2,hc_metric="manhattan",hc_meth
od="single",graph=FALSE)
> fviz_silhouette(hclust2, palette=c("red","blue"),ggtheme=them
e_classic())
  cluster size ave.sil.width
1       1  191          0.42
2       2    2          0.72
```



From the silhouette we can extract the Average silhouette widths of both clusters:

```
> hclust2$silinfo$clus.avg.widths
[1] 0.4178587 0.7222115
> hclust2$silinfo$avg.width
[1] 0.4210126
```

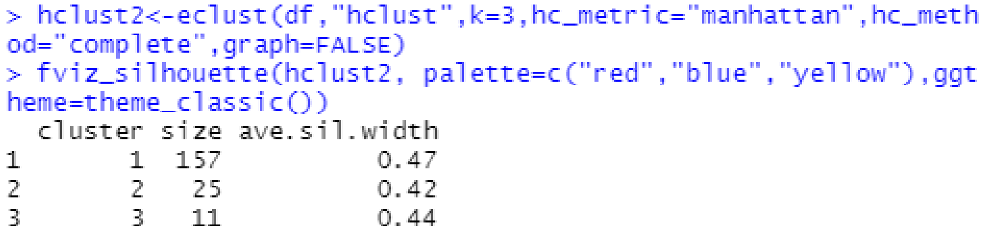
Next to be computed is the ***Dunn index***.

```
> hc_stats<-cluster.stats(dist(df),hclust2$cluster)
> hc_stats$dunn
[1] 0.2360685
```

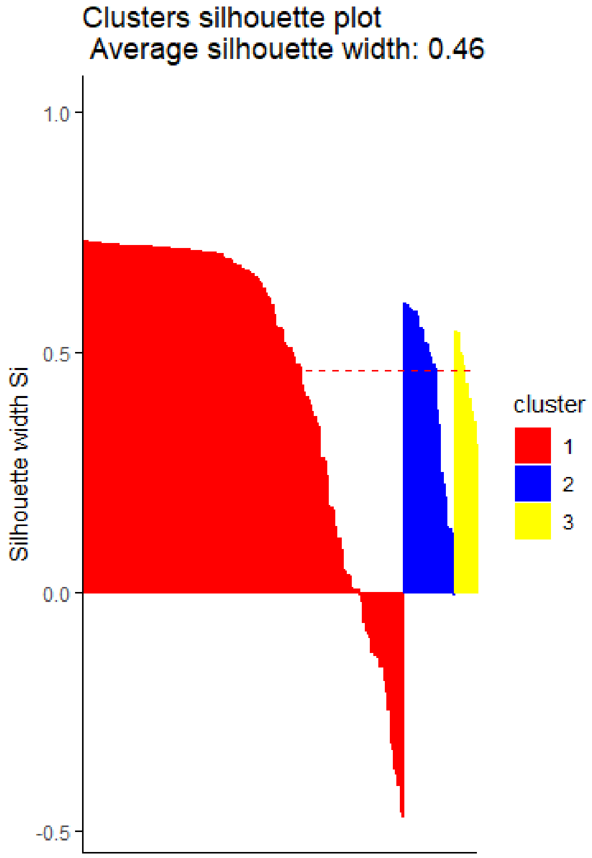
1.7. INTERNAL CLUSTER VALIDATION

The following step consist of evaluate the clustering result, starting from the internal cluster validation of the ***AHC with complete linkage method and Manhattan distance***.

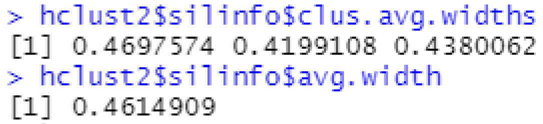
Let’s compute the ***silhouette width***:



```
> hclust2<-eclust(df,"hclust",k=3,hc_metric="manhattan",hc_meth
od="complete",graph=FALSE)
> fviz_silhouette(hclust2, palette=c("red","blue","yellow"),ggt
heme=theme_classic())
  cluster size ave.sil.width
1       1  157          0.47
2       2   25          0.42
3       3   11          0.44
```

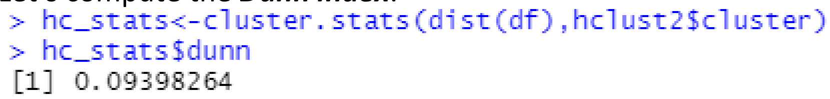


From the silhouette we can extract the Average silhouette widths of both clusters:



```
> hclust2$silinfo$clus.avg.widths
[1] 0.4697574 0.4199108 0.4380062
> hclust2$silinfo$avg.width
[1] 0.4614909
```

Let’s compute the ***Dunn index***.

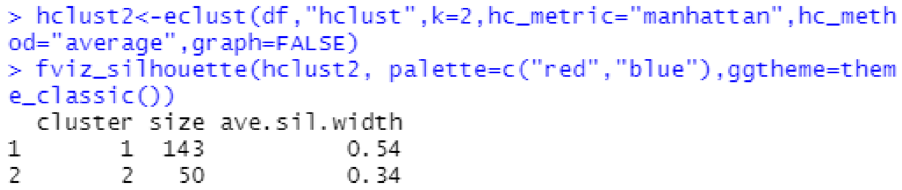


```
> hc_stats<-cluster.stats(dist(df),hclust2$cluster)
> hc_stats$dunn
[1] 0.09398264
```

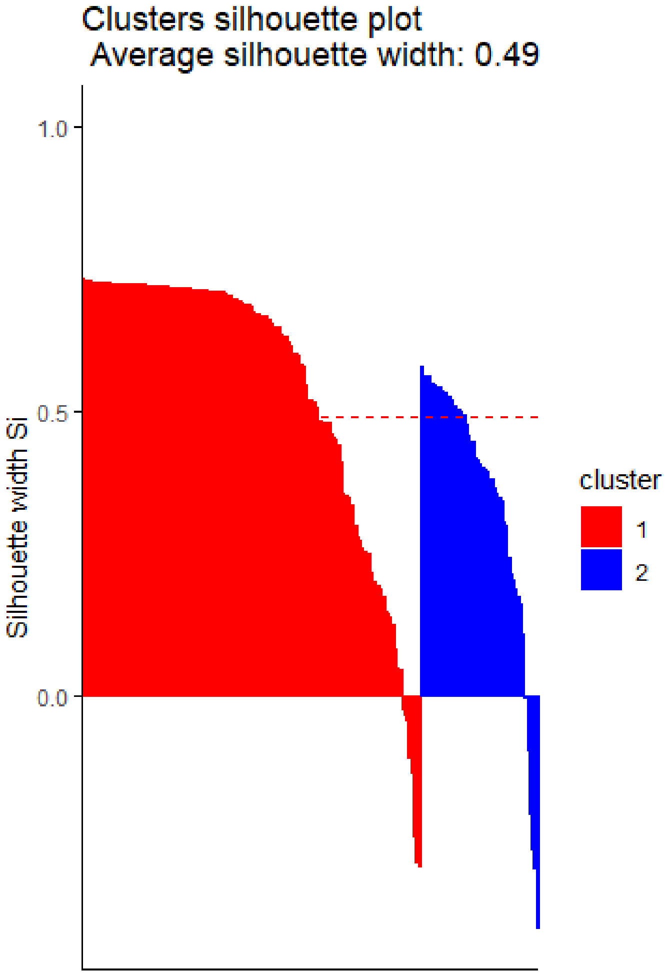
1.8. INTERNAL CLUSTER VALIDATION

The following step consist of evaluate the clustering result, starting from the internal cluster validation of the ***AHC with average linkage method and Manhattan distance***.

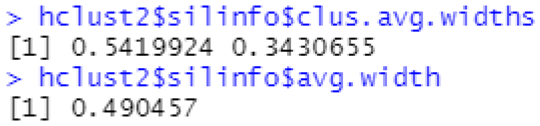
Let’s compute the ***silhouette width***:



```
> hclust2<-eclust(df,"hclust",k=2,hc_metric="manhattan",hc_meth
od="average",graph=FALSE)
> fviz_silhouette(hclust2, palette=c("red","blue"),ggtheme=them
e_classic())
  cluster size ave.sil.width
1       1  143          0.54
2       2   50          0.34
```

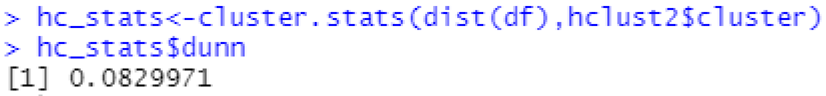


From the silhouette we can extract the Average silhouette widths of both clusters:



```
> hclust2$silinfo$clus.avg.widths
[1] 0.5419924 0.3430655
> hclust2$silinfo$avg.width
[1] 0.490457
```

Let’s compute the ***Dunn index***.



```
> hc_stats<-cluster.stats(dist(df),hclust2$cluster)
> hc_stats$dunn
[1] 0.0829971
```

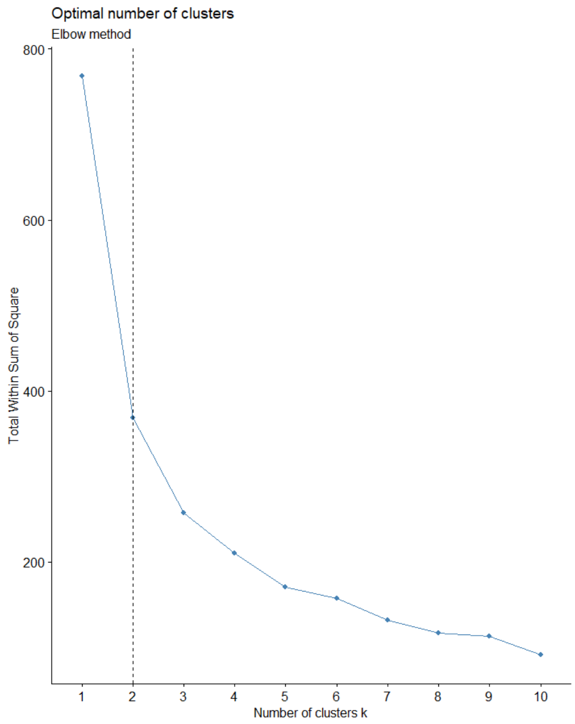
PARTIONING CLUSTERING

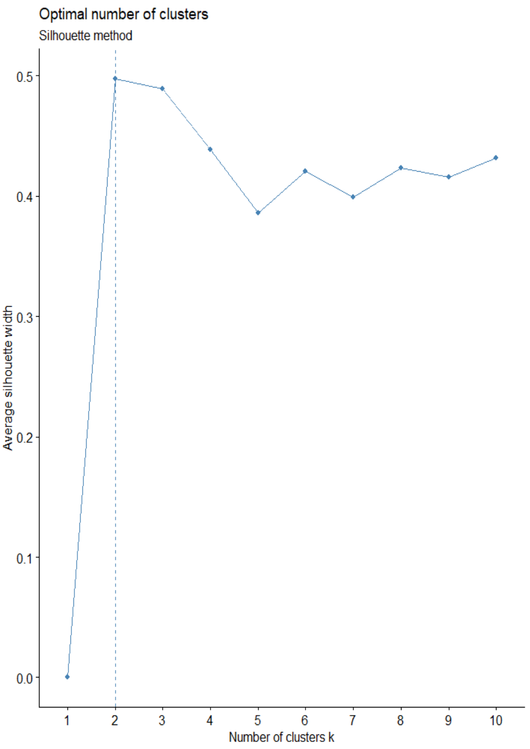
The partitioning clustering approach is another method used to perform the cluster analysis. There are two different types of partitioning clustering:

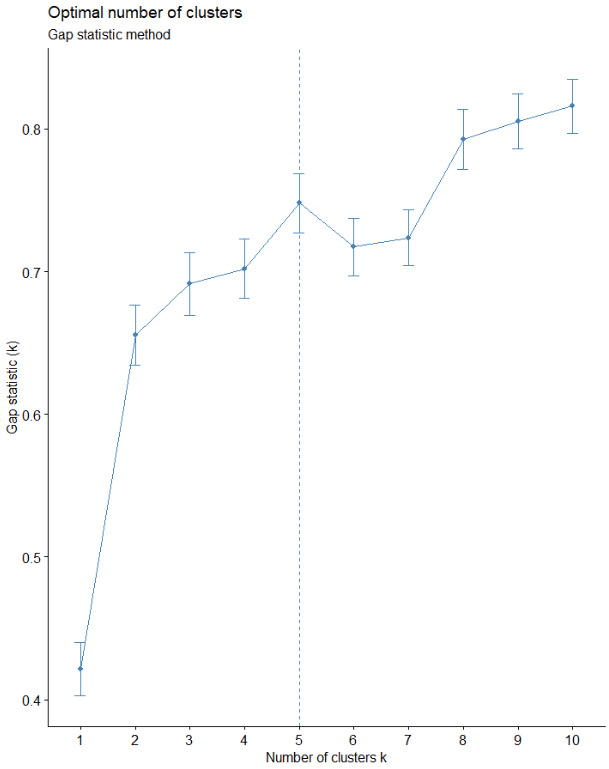
- ***K-means***, that classifies the units to the clusters K, in order to provide a high cluster cohesion and high cluster separation.
- ***K-medoids***, where each cluster is represented by one of the data points (also known as cluster medoids) in the cluster.

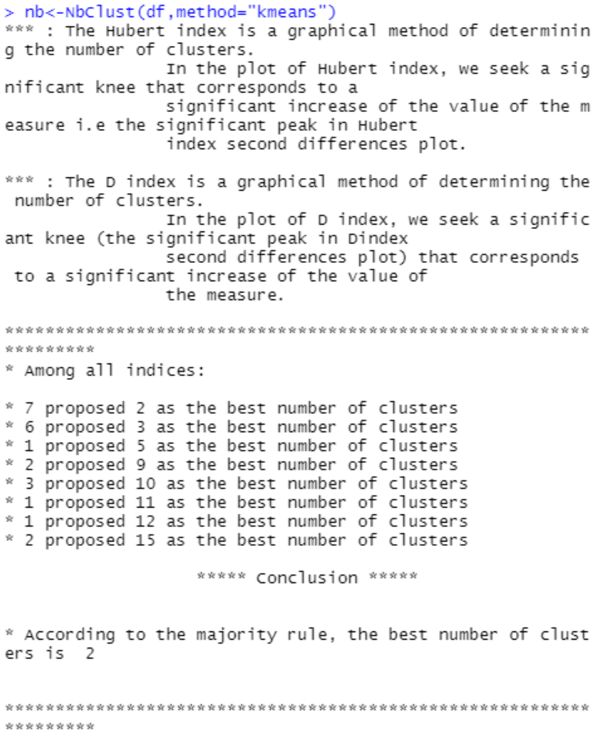
We'll start our analysis with the ***K-means***, and we are going to determine the optimal number of clusters.

```
> fviz_nbclust(df, kmeans, method="wss")+
+   labs(subtitle="Elbow method")+
+   geom_vline(xintercept=2,linetype=2)
> fviz_nbclust(df, kmeans, method="wss")+
+   labs(subtitle="Elbow method")+
+   geom_vline(xintercept=2,linetype=2)
> fviz_nbclust(df,kmeans,method="silhouette")+
+   labs(subtitle="Silhouette method")
> fviz_nbclust(df,kmeans,method="gap_stat",nboot=500)+
+   labs(subtitle="Gap statistic method")
Clustering k = 1,2,..., K.max (= 10): .. done
Bootstrapping, b = 1,2,..., B (= 500)  [one "." per sampl
e]:
.................................................. 50
.................................................. 100
.................................................. 150
.................................................. 200
.................................................. 250
.................................................. 300
.................................................. 350
.................................................. 400
.................................................. 450
.................................................. 500
```

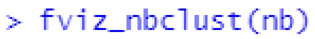




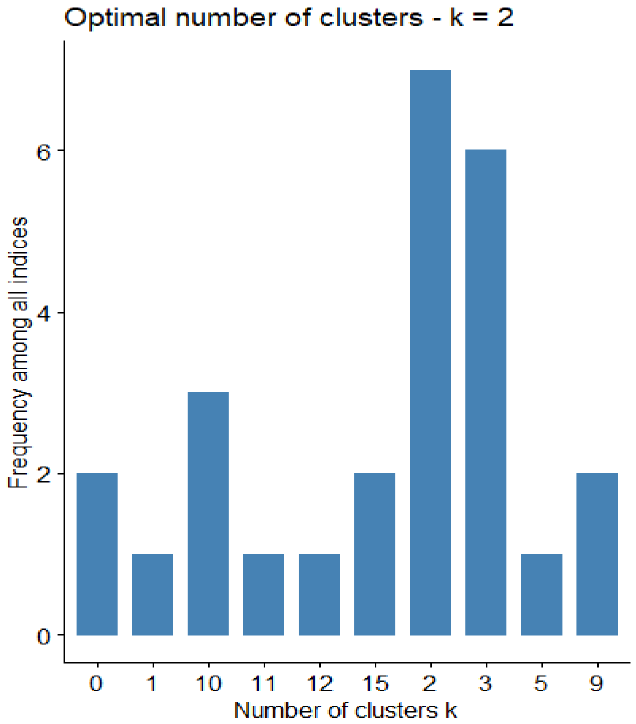




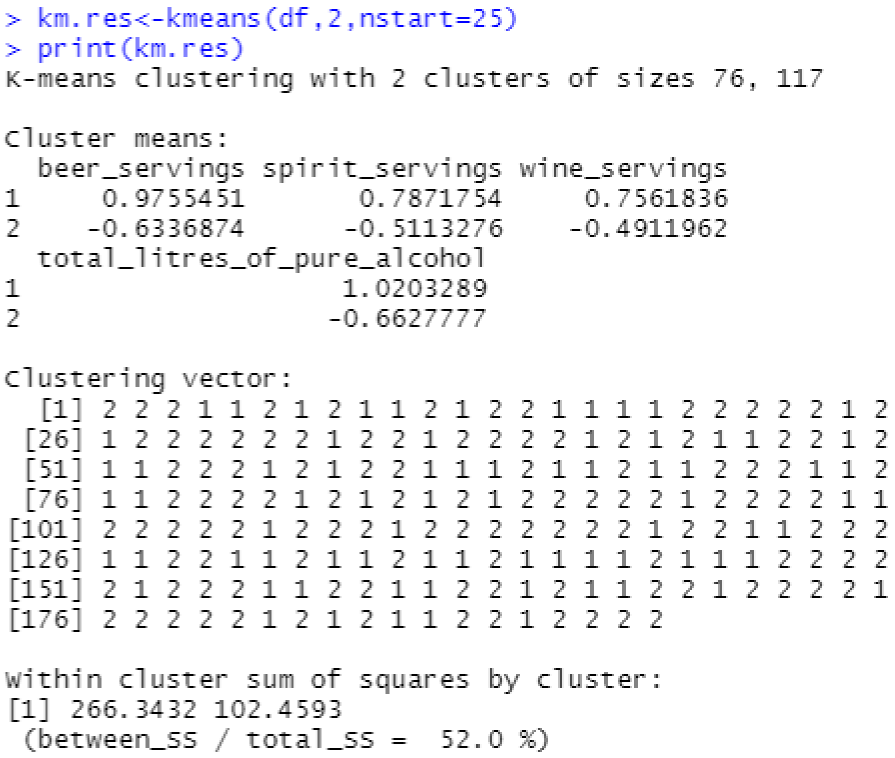
```
> nb<-NbClust(df,method="kmeans")
*** : The Hubert index is a graphical method of determinin
g the number of clusters.
                In the plot of Hubert index, we seek a sig
nificant knee that corresponds to a
                significant increase of the value of the m
easure i.e the significant 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 signific
ant knee (the significant peak in Dindex
                second differences plot) that corresponds
 to a significant increase of the value of
                the measure.

***********************************************************
*********
* Among all indices:

* 7 proposed 2 as the best number of clusters
* 6 proposed 3 as the best number of clusters
* 1 proposed 5 as the best number of clusters
* 2 proposed 9 as the best number of clusters
* 3 proposed 10 as the best number of clusters
* 1 proposed 11 as the best number of clusters
* 1 proposed 12 as the best number of clusters
* 2 proposed 15 as the best number of clusters

                   ***** Conclusion *****


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


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



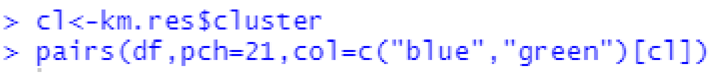
```
> fviz_nbclust(nb)
```

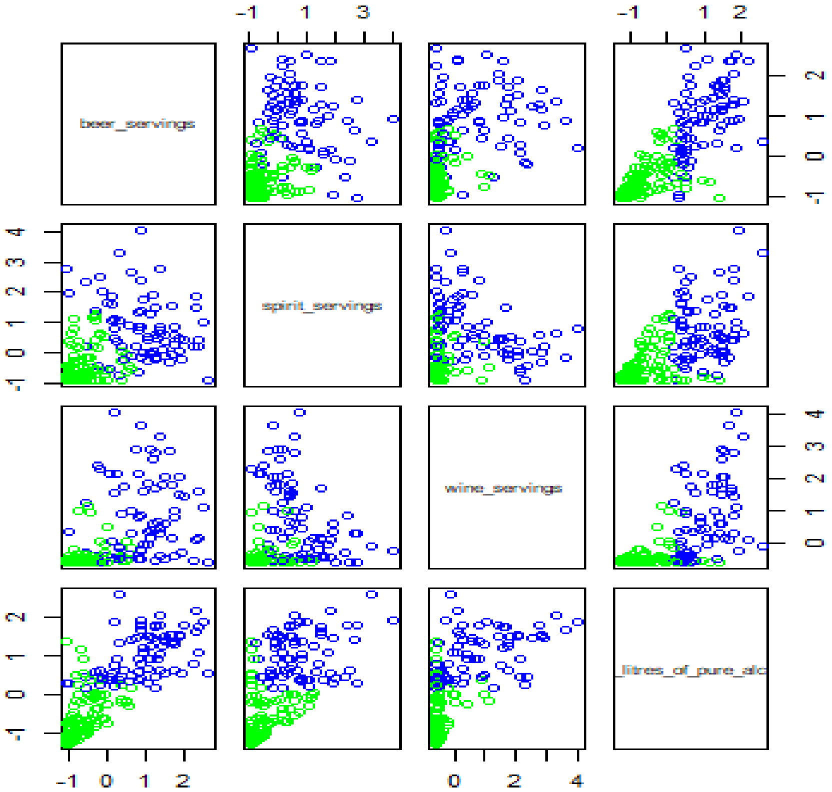


According to the computation, the optimal number of clusters for the K-means is K=2.

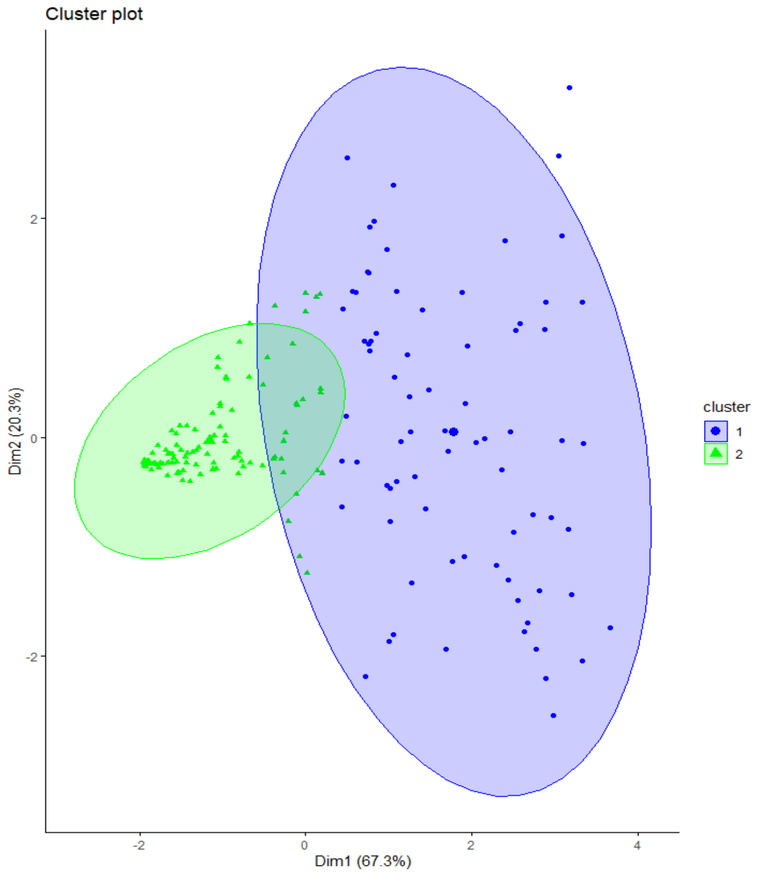

```
> km.res<-kmeans(df,2,nstart=25)
> print(km.res)
K-means clustering with 2 clusters of sizes 76, 117

Cluster means:
  beer_servings spirit_servings wine_servings
1     0.9755451       0.7871754     0.7561836
2    -0.6336874      -0.5113276    -0.4911962
  total_litres_of_pure_alcohol
1                    1.0203289
2                   -0.6627777

Clustering vector:
  [1] 2 2 2 1 1 2 1 2 1 1 2 1 2 2 1 1 1 1 2 2 2 2 2 1 2
 [26] 1 2 2 2 2 2 2 1 2 2 1 2 2 2 2 1 2 1 2 1 1 2 2 1 2
 [51] 1 1 2 2 2 1 2 1 2 2 1 1 1 2 1 1 2 1 1 2 2 2 1 1 2
 [76] 1 1 2 2 2 2 1 2 1 2 1 2 1 2 2 2 2 2 1 2 2 2 2 1 1
[101] 2 2 2 2 2 1 2 2 2 1 2 2 2 2 2 2 2 1 2 2 1 1 2 2 2
[126] 1 1 2 2 1 1 2 1 1 2 1 1 2 1 1 1 1 2 1 1 1 2 2 2 2
[151] 2 1 2 2 2 1 1 2 2 1 1 2 2 1 2 1 1 2 2 1 2 2 2 2 1
[176] 2 2 2 2 2 1 2 1 2 1 1 2 2 1 2 2 2 2

Within cluster sum of squares by cluster:
[1] 266.3432 102.4593
 (between_SS / total_SS =  52.0 %)
```

We can also represent the obtained clusters in the original space:


```
> cl<-km.res$cluster
> pairs(df,pch=21,col=c("blue","green")[cl])
```



```
> fviz_cluster(km.res,data=df, geom="point",ellipse.type
="norm",palette=c("blue","green"),ggtheme=theme_classic())
```

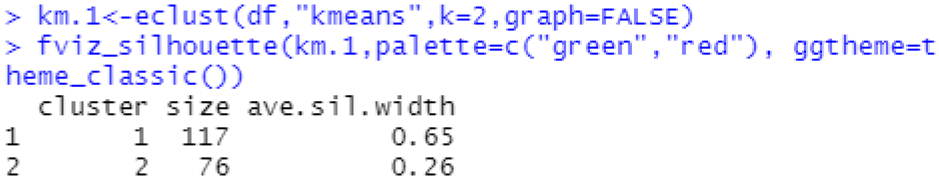


So, the results attest that the optimal number of clusters is K=2. We can also affirm that in the first cluster there are 76 units, and in the second cluster there are 117 units.

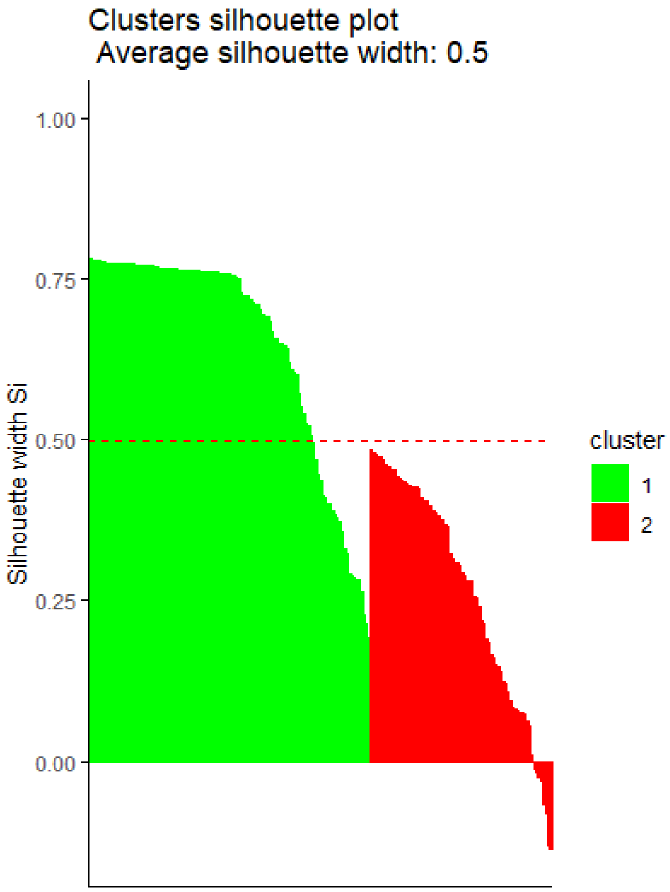
1.9. INTERNAL CLUSTER VALIDATION

The following step consist of evaluate the clustering result, starting from the internal cluster validation of the ***K-means method***.

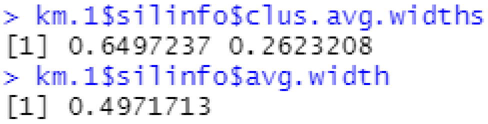
First to be computed is the ***silhouette width***.



```
> km.1<-eclust(df,"kmeans",k=2,graph=FALSE)
> fviz_silhouette(km.1,palette=c("green","red"), ggtheme=t
heme_classic())
  cluster size ave.sil.width
1       1  117          0.65
2       2   76          0.26
```



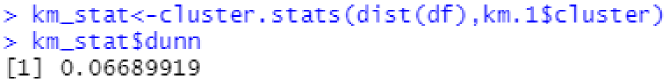
From the silhouette we can extract the Average silhouette widths of both clusters:



```
> km.1$silinfo$clus.avg.widths
[1] 0.6497237 0.2623208
> km.1$silinfo$avg.width
[1] 0.4971713
```

In this case, based on the value above (0.4971713), we can say that the units are quite well-clustered.

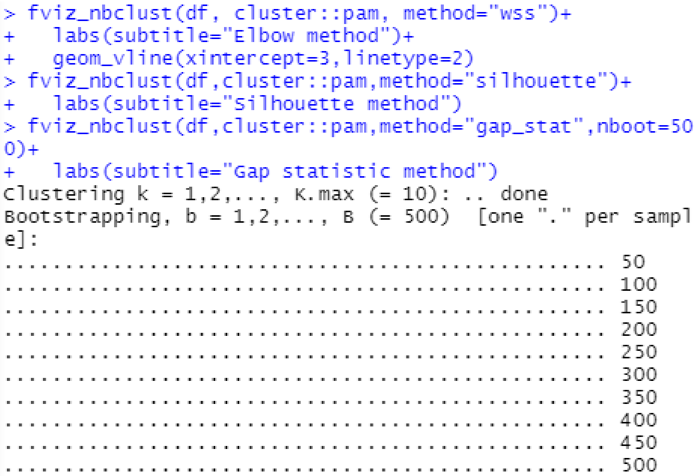
Then, the ***Dunn index***.



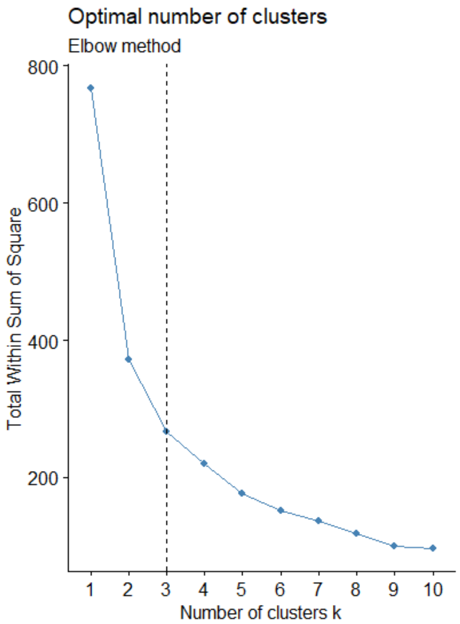
```
> km_stat<-cluster.stats(dist(df),km.1$cluster)
> km_stat$dunn
 [1] 0.06689919
```

After the internal cluster validation, we move on the ***K-medoids*** approach.

In the following code there is the computation of the optimal number of clusters.

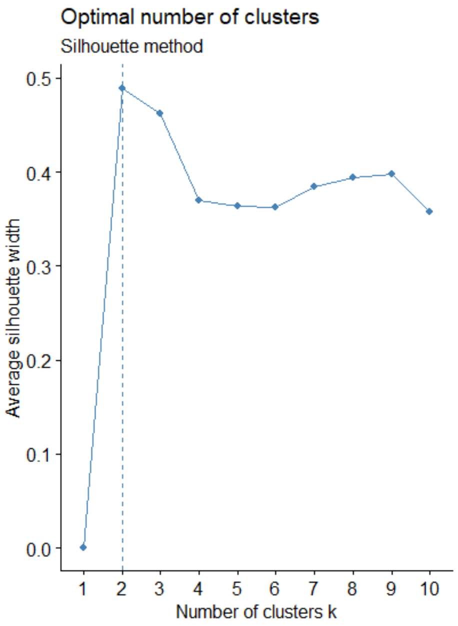


```
> fviz_nbclust(df, cluster::pam, method="wss")+
+   labs(subtitle="Elbow method")+
+   geom_vline(xintercept=3,linetype=2)
> fviz_nbclust(df,cluster::pam,method="silhouette")+
+   labs(subtitle="Silhouette method")
> fviz_nbclust(df,cluster::pam,method="gap_stat",nboot=50
0)+
+   labs(subtitle="Gap statistic method")
Clustering k = 1,2,..., K.max (= 10): .. done
Bootstrapping, b = 1,2,..., B (= 500)  [one "." per sampl
e]:
.................................................. 50
.................................................. 100
.................................................. 150
.................................................. 200
.................................................. 250
.................................................. 300
.................................................. 350
.................................................. 400
.................................................. 450
.................................................. 500
```



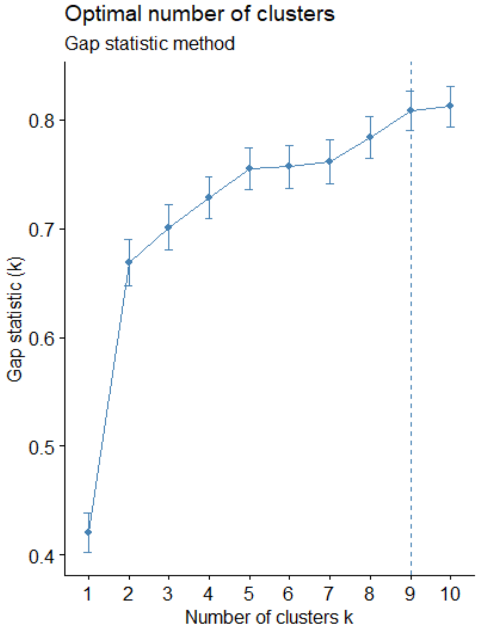
Optimal number of clusters

Elbow method



Optimal number of clusters

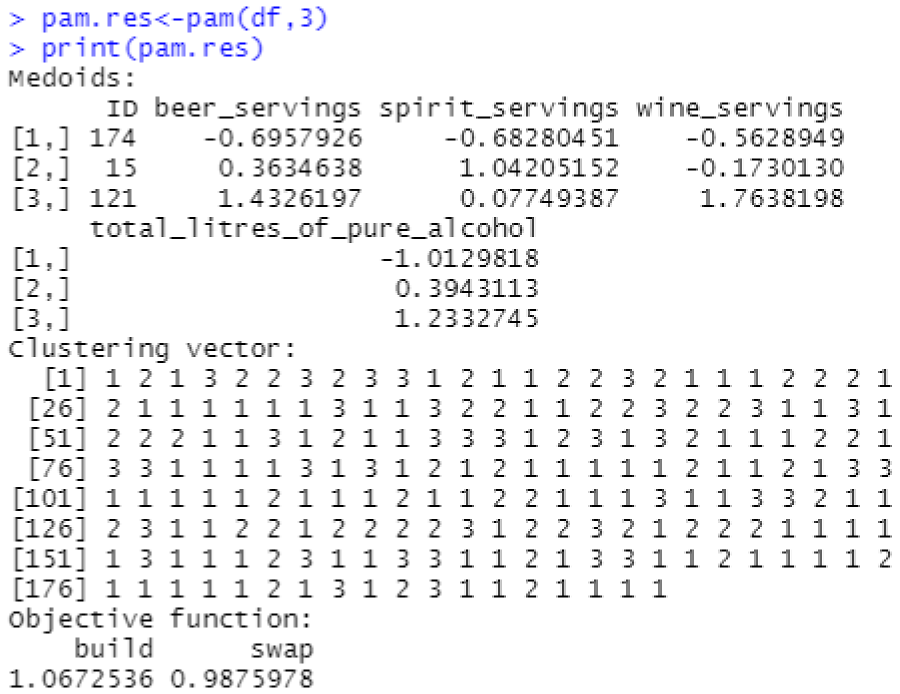
Silhouette method



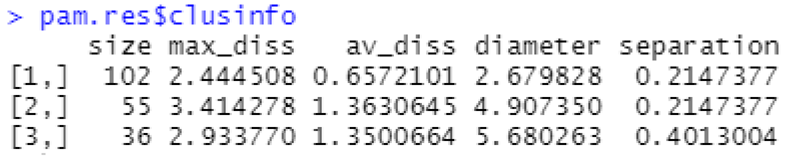
Optimal number of clusters

Gap statistic method

I decided to proceed with number of cluster K=3.

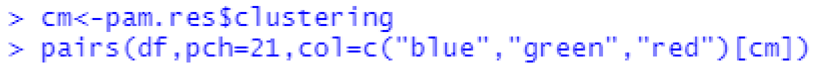


```
> pam.res<-pam(df,3)
> print(pam.res)
Medoids:
      ID beer_servings spirit_servings wine_servings
[1,] 174    -0.6957926     -0.68280451    -0.5628949
[2,]  15     0.3634638      1.04205152    -0.1730130
[3,] 121     1.4326197      0.07749387     1.7638198
     total_litres_of_pure_alcohol
[1,]                   -1.0129818
[2,]                    0.3943113
[3,]                    1.2332745
Clustering vector:
  [1] 1 2 1 3 2 2 3 2 3 3 1 2 1 1 2 2 3 2 1 1 1 2 2 2 1
 [26] 2 1 1 1 1 1 1 3 1 1 3 2 2 1 1 2 2 3 2 2 3 1 1 3 1
 [51] 2 2 2 1 1 3 1 2 1 1 3 3 3 1 2 3 1 3 2 1 1 1 2 2 1
 [76] 3 3 1 1 1 1 3 1 3 1 2 1 2 1 1 1 1 1 2 1 1 2 1 3 3
[101] 1 1 1 1 1 2 1 1 1 2 1 1 2 2 1 1 1 3 1 1 3 3 2 1 1
[126] 2 3 1 1 2 2 1 2 2 2 2 3 1 2 2 3 2 1 2 2 2 1 1 1 1
[151] 1 3 1 1 1 2 3 1 1 3 3 1 1 2 1 3 3 1 1 2 1 1 1 1 2
[176] 1 1 1 1 1 2 1 3 1 2 3 1 1 2 1 1 1 1
Objective function:
    build      swap 
1.0672536 0.9875978 
```

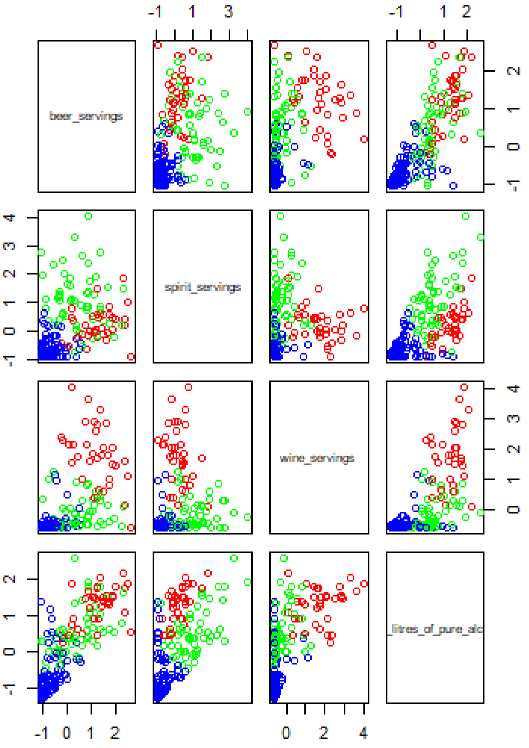


```
> pam.res$clusinfo
     size max_diss    av_diss diameter separation
[1,]  102 2.444508 0.6572101 2.679828  0.2147377
[2,]   55 3.414278 1.3630645 4.907350  0.2147377
[3,]   36 2.933770 1.3500664 5.680263  0.4013004
```

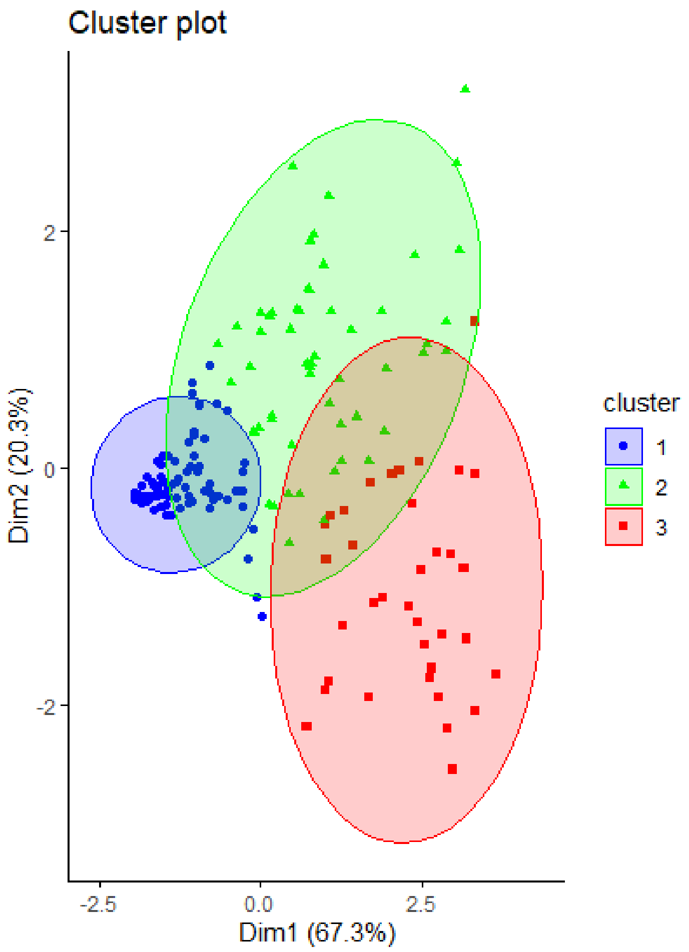
Going on, we can see the obtained clusters in the original space.



```
> cm<-pam.res$clustering
> pairs(df,pch=21,col=c("blue","green","red")[cm])
```



```
> fviz_cluster(pam.res,data=df, geom="point",ellipse.type
="norm",palette=c("blue","green","red"),ggtheme=theme_clas
sic())
```



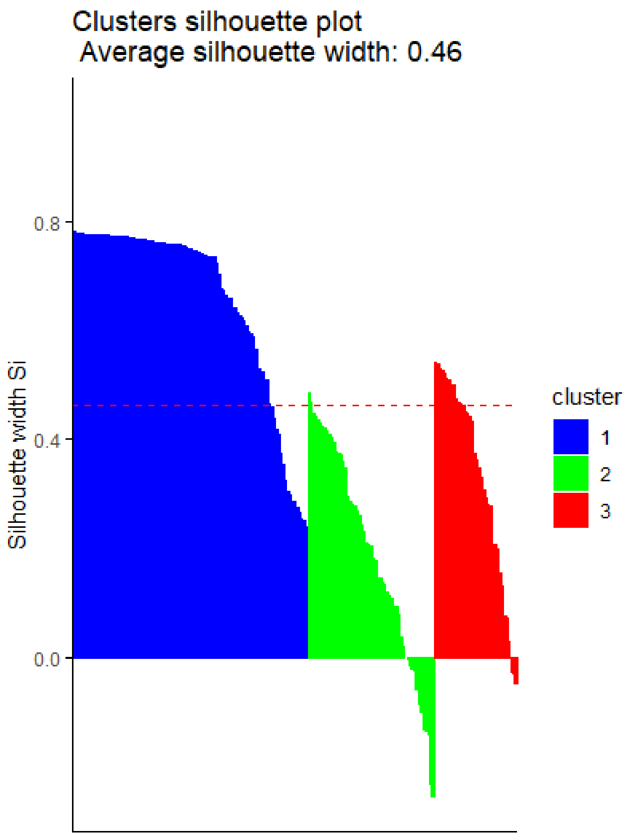
The K-medoids gives us 3 clusters: the first cluster, in blue, with 102 units, the second in green with 55 units, and the third one in red, with 36 units.

1.10. INTERNAL CLUSTER VALIDATION

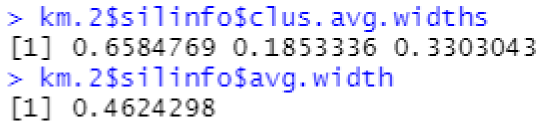
The following step consist of evaluate the clustering result, starting from the internal cluster validation of the ***K-medoids*** approach.

The ***silhouette width*** computation:

```
> km.2<-eclust(df,"pam",k=3, graph=FALSE)
> fviz_silhouette(km.2,palette=c("blue","green","red"), gg
theme=theme_classic())
  cluster size ave.sil.width
1       1  102          0.66
2       2   55          0.19
3       3   36          0.33
```

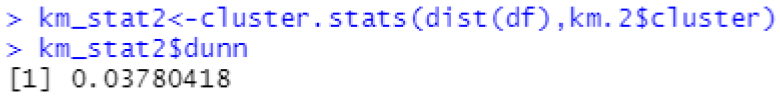


From the silhouette we can extract the Average silhouette widths of both clusters:



```
> km.2$silinfo$clus.avg.widths
[1] 0.6584769 0.1853336 0.3303043
> km.2$silinfo$avg.width
[1] 0.4624298
```

The value (0.4624298) is above 0, so the partitioning is quite good.
Next to be computed is the ***Dunn index.***

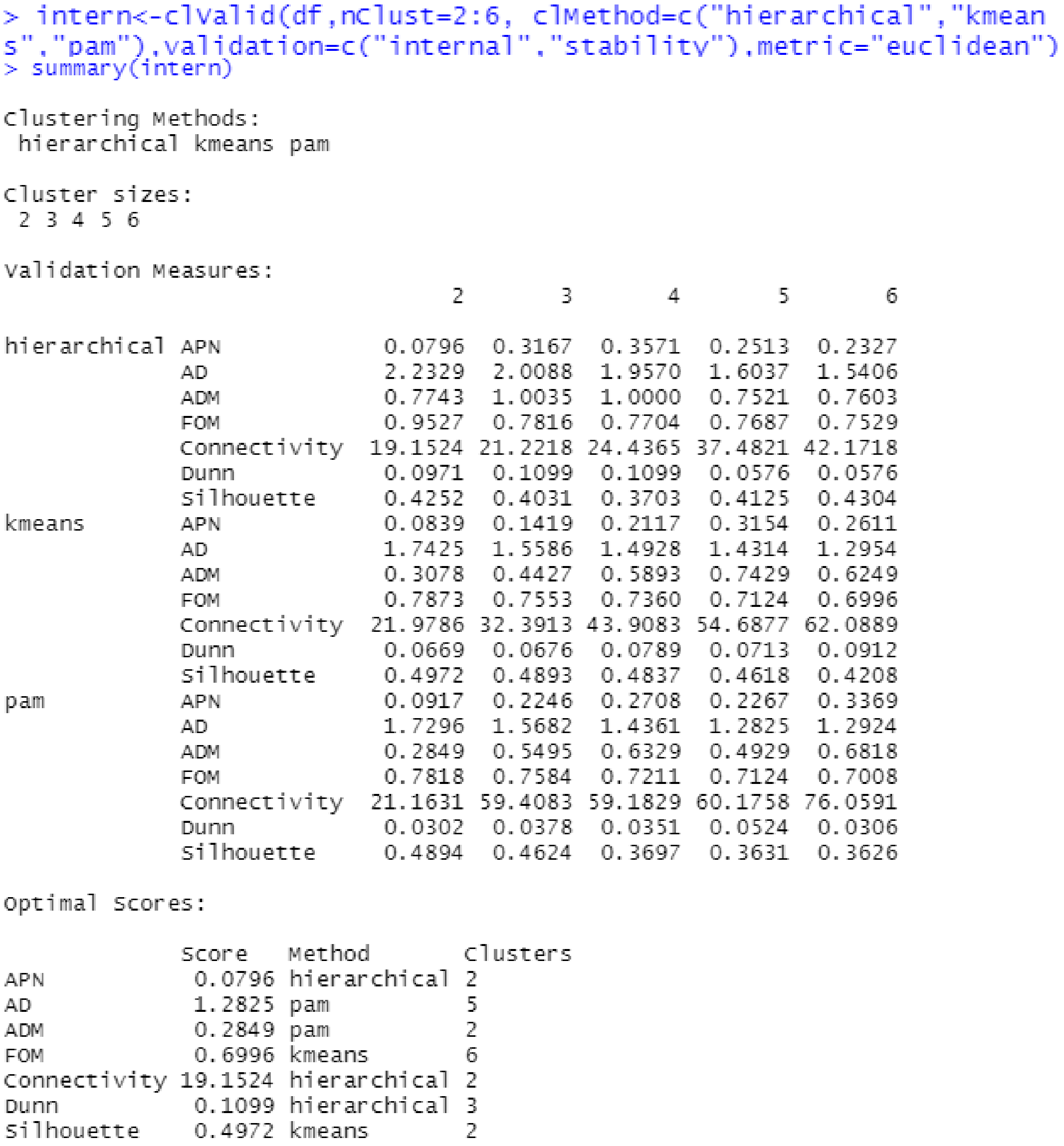


```
> km_stat2<-cluster.stats(dist(df),km.2$cluster)
> km_stat2$dunn
[1] 0.03780418
```

According to the value (0.03780418) that we obtained, the observations are not well-clustered.

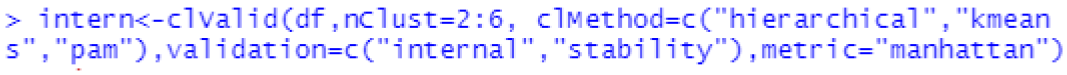
BEST CLUSTERING ALGORITHM

At this point, we have to choose the best clustering algorithm and the optimal number of clusters. To do that, we need to look and evaluate the internal and stability measures.

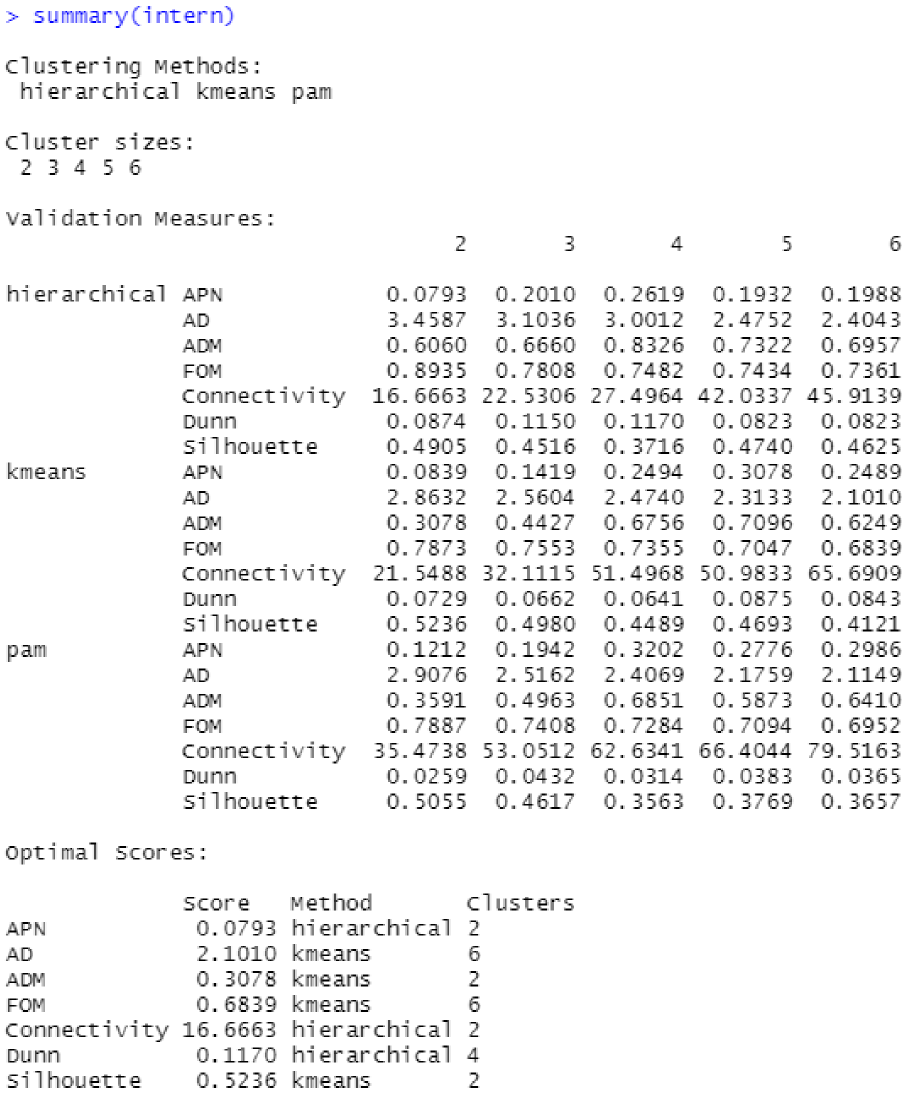


```
> intern<-clValid(df,nClust=2:6, clMethod=c("hierarchical","kmean
s","pam"),validation=c("internal","stability"),metric="euclidean")
> summary(intern)

Clustering Methods:
 hierarchical kmeans pam 

Cluster sizes:
 2 3 4 5 6 

Validation Measures:
                                 2       3       4       5       6
                                                                  
hierarchical APN            0.0796  0.3167  0.3571  0.2513  0.2327
             AD             2.2329  2.0088  1.9570  1.6037  1.5406
             ADM            0.7743  1.0035  1.0000  0.7521  0.7603
             FOM            0.9527  0.7816  0.7704  0.7687  0.7529
             Connectivity  19.1524 21.2218 24.4365 37.4821 42.1718
             Dunn           0.0971  0.1099  0.1099  0.0576  0.0576
             Silhouette     0.4252  0.4031  0.3703  0.4125  0.4304
kmeans       APN            0.0839  0.1419  0.2117  0.3154  0.2611
             AD             1.7425  1.5586  1.4928  1.4314  1.2954
             ADM            0.3078  0.4427  0.5893  0.7429  0.6249
             FOM            0.7873  0.7553  0.7360  0.7124  0.6996
             Connectivity  21.9786 32.3913 43.9083 54.6877 62.0889
             Dunn           0.0669  0.0676  0.0789  0.0713  0.0912
             Silhouette     0.4972  0.4893  0.4837  0.4618  0.4208
pam          APN            0.0917  0.2246  0.2708  0.2267  0.3369
             AD             1.7296  1.5682  1.4361  1.2825  1.2924
             ADM            0.2849  0.5495  0.6329  0.4929  0.6818
             FOM            0.7818  0.7584  0.7211  0.7124  0.7008
             Connectivity  21.1631 59.4083 59.1829 60.1758 76.0591
             Dunn           0.0302  0.0378  0.0351  0.0524  0.0306
             Silhouette     0.4894  0.4624  0.3697  0.3631  0.3626

Optimal Scores:

             Score   Method       Clusters
APN          0.0796  hierarchical 2       
AD           1.2825  pam          5       
ADM          0.2849  pam          2       
FOM          0.6996  kmeans       6       
Connectivity 19.1524 hierarchical 2       
Dunn         0.1099  hierarchical 3       
Silhouette   0.4972  kmeans       2       
```

Based on the results that we obtained with the Euclidean distance, as we can see 3 of the 7 indexes suggest that the best one is the hierarchical clustering approach, with an optimal number of clusters K=2.



```
> intern<-clValid(df,nClust=2:6, clMethod=c("hierarchical","kmean
s","pam"),validation=c("internal","stability"),metric="manhattan")
```

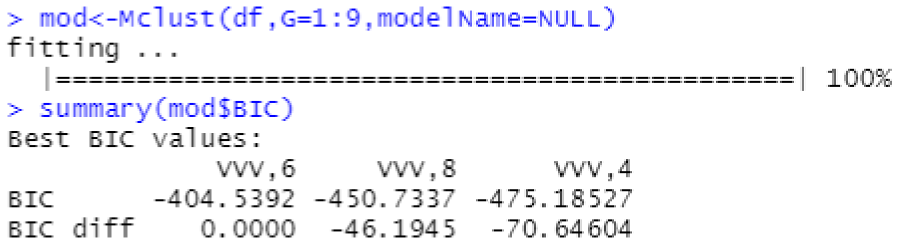


```
> summary(intern)

Clustering Methods:
 hierarchical kmeans pam

Cluster sizes:
 2 3 4 5 6

Validation Measures:
                                 2       3       4       5       6

hierarchical APN            0.0793  0.2010  0.2619  0.1932  0.1988
             AD             3.4587  3.1036  3.0012  2.4752  2.4043
             ADM            0.6060  0.6660  0.8326  0.7322  0.6957
             FOM            0.8935  0.7808  0.7482  0.7434  0.7361
             Connectivity  16.6663 22.5306 27.4964 42.0337 45.9139
             Dunn           0.0874  0.1150  0.1170  0.0823  0.0823
             Silhouette     0.4905  0.4516  0.3716  0.4740  0.4625
kmeans       APN            0.0839  0.1419  0.2494  0.3078  0.2489
             AD             2.8632  2.5604  2.4740  2.3133  2.1010
             ADM            0.3078  0.4427  0.6756  0.7096  0.6249
             FOM            0.7873  0.7553  0.7355  0.7047  0.6839
             Connectivity  21.5488 32.1115 51.4968 50.9833 65.6909
             Dunn           0.0729  0.0662  0.0641  0.0875  0.0843
             Silhouette     0.5236  0.4980  0.4489  0.4693  0.4121
pam          APN            0.1212  0.1942  0.3202  0.2776  0.2986
             AD             2.9076  2.5162  2.4069  2.1759  2.1149
             ADM            0.3591  0.4963  0.6851  0.5873  0.6410
             FOM            0.7887  0.7408  0.7284  0.7094  0.6952
             Connectivity  35.4738 53.0512 62.6341 66.4044 79.5163
             Dunn           0.0259  0.0432  0.0314  0.0383  0.0365
             Silhouette     0.5055  0.4617  0.3563  0.3769  0.3657

Optimal Scores:

             Score   Method       Clusters
APN           0.0793 hierarchical 2
AD            2.1010 kmeans       6
ADM           0.3078 kmeans       2
FOM           0.6839 kmeans       6
Connectivity 16.6663 hierarchical 2
Dunn          0.1170 hierarchical 4
Silhouette    0.5236 kmeans       2
```

Based on the results obtained with the Manhattan distance, the hierarchical clustering approach with K=2 gives the best score, but for other measure the k-means approach with K=2 has the best score.

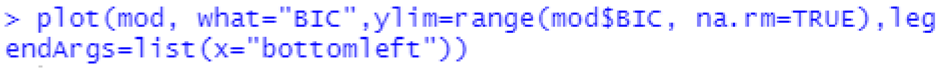
MODEL BASED CLUSTERING

Also known as a soft assignment, the model-based clustering considers that one observation can belong to all the different clusters obtained for our dataset but with a different probability. The model-based clustering is based on statistical models.
With the function Mclust() we will fit different parsimonious Gaussian mixtures on the standardized data. Default G is 1:9.

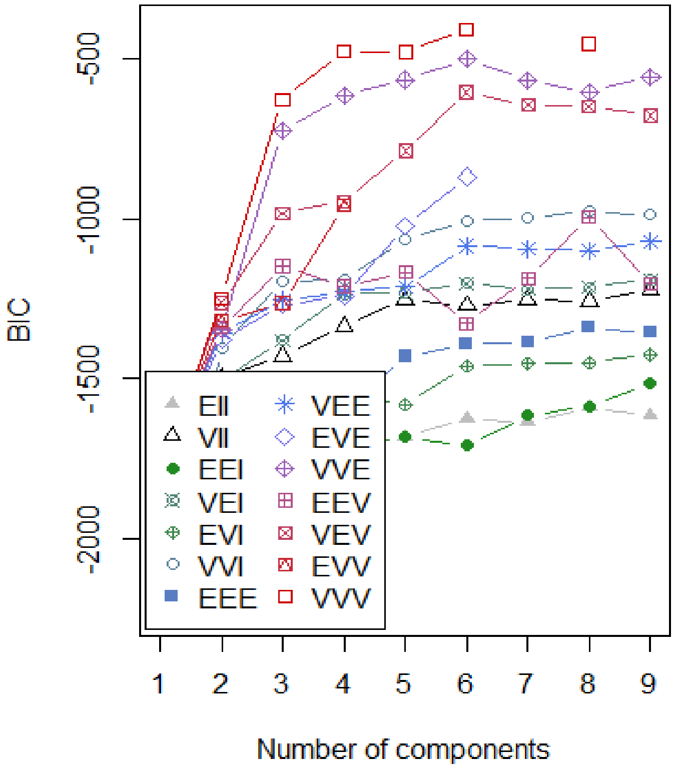


```
> mod<-Mclust(df,G=1:9,modelName=NULL)
fitting ...
  |==============================================| 100%
> summary(mod$BIC)
Best BIC values:
             VVV,6       VVV,8        VVV,4
BIC      -404.5392 -450.7337 -475.18527
BIC diff    0.0000  -46.1945  -70.64604
```

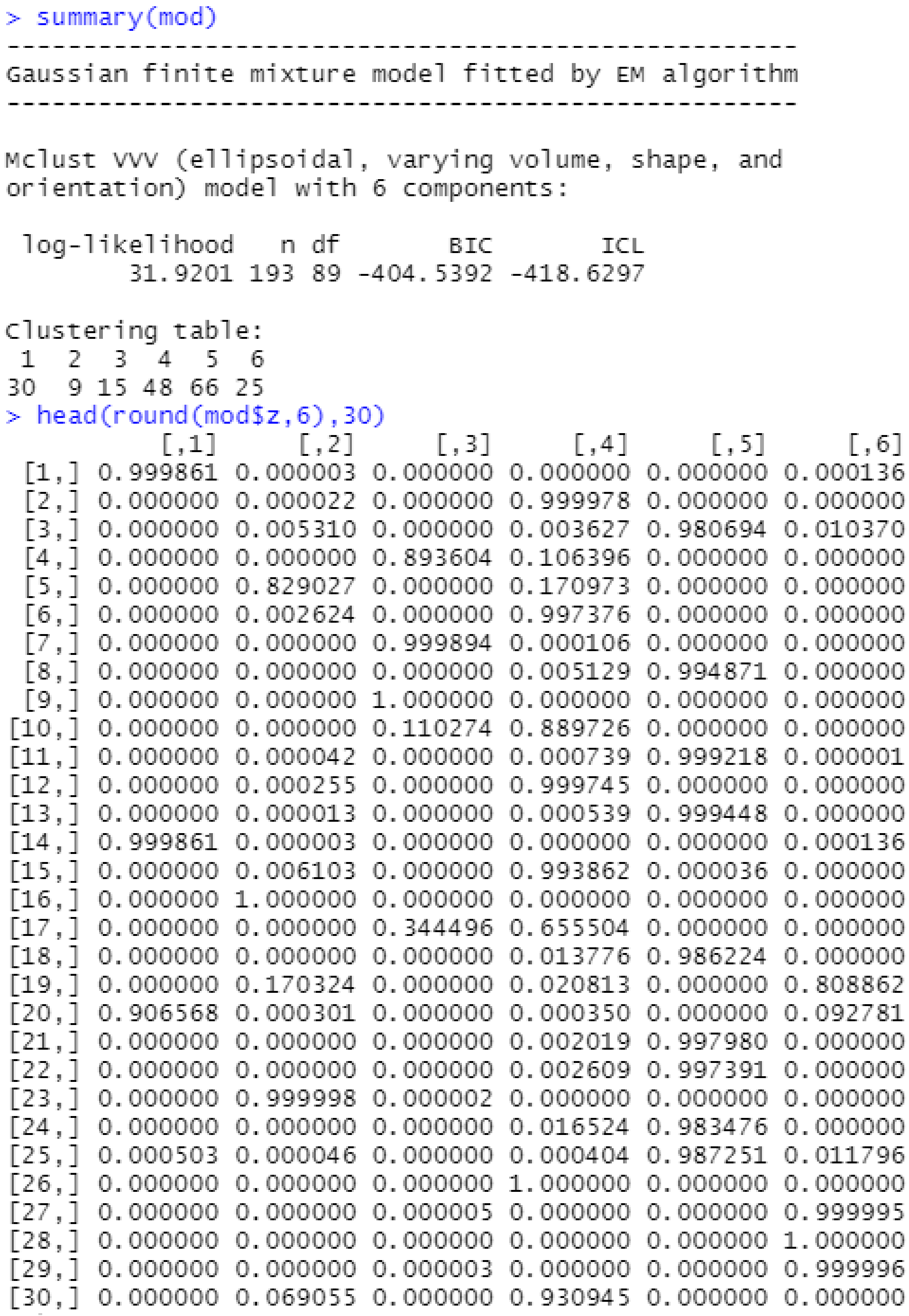
According with the output, we can see that the maximized value of BIC (Bayesian Information Criterion) is at the parsimonious model VVV with 6 clusters. The BIC diff. indicates the difference between the first and the second model.
Now we can plot the graphical counterpart of the previous code.



```
> plot(mod, what="BIC",ylim=range(mod$BIC, na.rm=TRUE),leg
endArgs=list(x="bottomleft"))
```

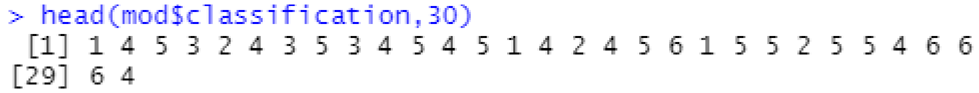


As we can see from the image above, we have different curve for each number of clusters and different symbols for all the parsimonious configurations. The highest point is for the VVV model with 6 clusters.



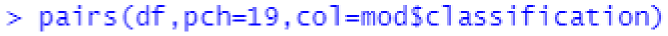
```
> summary(mod)
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust VVV (ellipsoidal, varying volume, shape, and
orientation) model with 6 components:

 log-likelihood   n df       BIC       ICL
        31.9201 193 89 -404.5392 -418.6297

Clustering table:
 1  2  3  4  5  6
30  9 15 48 66 25
> head(round(mod$z,6),30)
          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
 [1,] 0.999861 0.000003 0.000000 0.000000 0.000000 0.000136
 [2,] 0.000000 0.000022 0.000000 0.999978 0.000000 0.000000
 [3,] 0.000000 0.005310 0.000000 0.003627 0.980694 0.010370
 [4,] 0.000000 0.000000 0.893604 0.106396 0.000000 0.000000
 [5,] 0.000000 0.829027 0.000000 0.170973 0.000000 0.000000
 [6,] 0.000000 0.002624 0.000000 0.997376 0.000000 0.000000
 [7,] 0.000000 0.000000 0.999894 0.000106 0.000000 0.000000
 [8,] 0.000000 0.000000 0.000000 0.005129 0.994871 0.000000
 [9,] 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000
[10,] 0.000000 0.000000 0.110274 0.889726 0.000000 0.000000
[11,] 0.000000 0.000042 0.000000 0.000739 0.999218 0.000001
[12,] 0.000000 0.000255 0.000000 0.999745 0.000000 0.000000
[13,] 0.000000 0.000013 0.000000 0.000539 0.999448 0.000000
[14,] 0.999861 0.000003 0.000000 0.000000 0.000000 0.000136
[15,] 0.000000 0.006103 0.000000 0.993862 0.000036 0.000000
[16,] 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000
[17,] 0.000000 0.000000 0.344496 0.655504 0.000000 0.000000
[18,] 0.000000 0.000000 0.000000 0.013776 0.986224 0.000000
[19,] 0.000000 0.170324 0.000000 0.020813 0.000000 0.808862
[20,] 0.906568 0.000301 0.000000 0.000350 0.000000 0.092781
[21,] 0.000000 0.000000 0.000000 0.002019 0.997980 0.000000
[22,] 0.000000 0.000000 0.000000 0.002609 0.997391 0.000000
[23,] 0.000000 0.999998 0.000002 0.000000 0.000000 0.000000
[24,] 0.000000 0.000000 0.000000 0.016524 0.983476 0.000000
[25,] 0.000503 0.000046 0.000000 0.000404 0.987251 0.011796
[26,] 0.000000 0.000000 0.000000 1.000000 0.000000 0.000000
[27,] 0.000000 0.000000 0.000005 0.000000 0.000000 0.999995
[28,] 0.000000 0.000000 0.000000 0.000000 0.000000 1.000000
[29,] 0.000000 0.000000 0.000003 0.000000 0.000000 0.999996
[30,] 0.000000 0.069055 0.000000 0.930945 0.000000 0.000000
```

The image above indicates the probability to belong to a given cluster, better, this is the matrix of posterior probabilities.

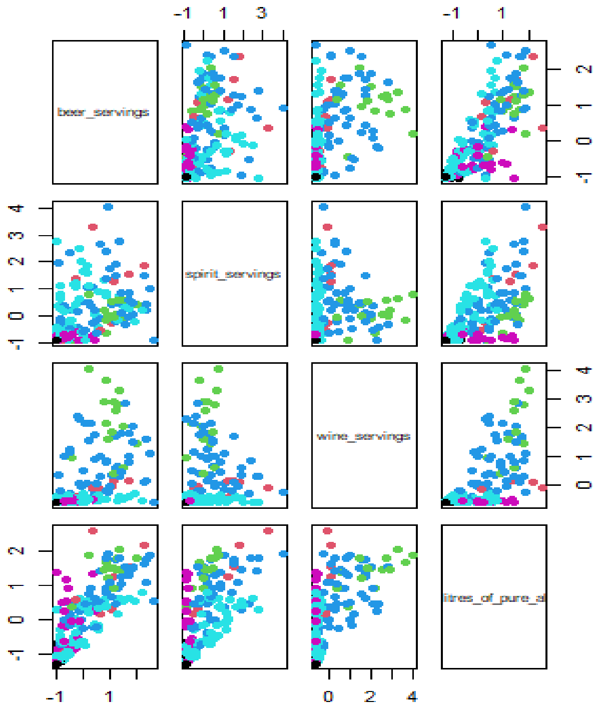


```
> head(mod$classification,30)
 [1] 1 4 5 3 2 4 3 5 3 4 5 4 5 1 4 2 4 5 6 1 5 5 2 5 5 4 6 6
[29] 6 4
```

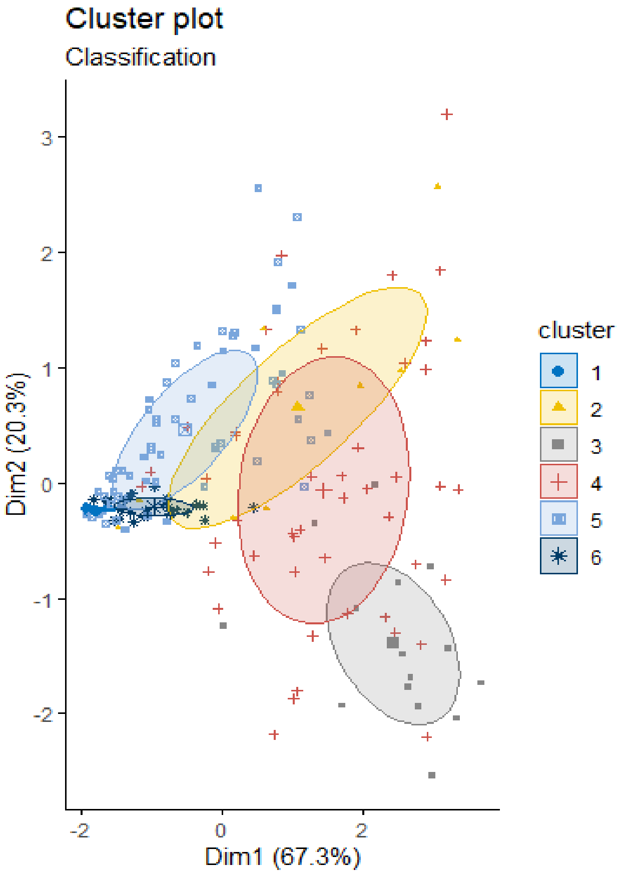
This above indicates the cluster assignment of each observation. After that we can visualize the clustering results it in the original space.



```
> pairs(df,pch=19,col=mod$classification)
```



```
> fviz_mclust(mod,"classification",geom="point",pointsize=1, pa
lette="jco")
```



As we can see, there are 6 clusters carried out by the VVV model in the PCs space.