Wholesale_dataset.R

acer

2023-02-18

```
#The analysis is performed on the "Wholesale customers" dataset from the UCI Machine Learning Repository[http://archive.ics.uci.edu/ml].
#The data set refers to clients of a wholesale distributor.
#It includes the annual spending in monetary units (m.u.) on diverse product categories
#The target feature are:
#•FRESH: annual spending (m.u.) on fresh products (Continuous);
#•MILK: annual spending (m.u.) on milk products (Continuous);
#•GROCERY: annual spending (m.u.)on grocery products (Continuous);
#•FROZEN: annual spending (m.u.)on frozen products (Continuous);
#•DETERGENTS PAPER: annual spending (m.u.) on detergents and paper products (Continuous);
#•DELICATESSEN: annual spending (m.u.)on and delicatessen products (Continuous);
#•CHANNEL: customers Channel - Horeca (Hotel/Restaurant/Cafè) or Retail channel (Nominal);
#•REGION: customers Region are splitted in: Lisbon, Oporto or Other Region (Nominal)


library("readr")
library(moments)
library("gamlss")
```

```
## Caricamento del pacchetto richiesto: splines
```

```
## Caricamento del pacchetto richiesto: gamlss.data
```

```
##
## Caricamento pacchetto: 'gamlss.data'
```

```
## Il seguente oggetto è mascherato da 'package:datasets':
##
##     sleep
```

```
## Caricamento del pacchetto richiesto: gamlss.dist
```

```
## Caricamento del pacchetto richiesto: MASS
```

```
## Caricamento del pacchetto richiesto: nlme
```

```
## Caricamento del pacchetto richiesto: parallel
```

```
##  **********   GAMLSS Version 5.4-12  **********
```

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

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

```
library(MASS)
library("fitdistrplus")
```

```
## Caricamento del pacchetto richiesto: survival
```

```
library("vcd")
```

```
## Caricamento del pacchetto richiesto: grid
```

```
library("corrplot")
```

```
## corrplot 0.92 loaded
```

```
library(gridExtra)
library(lmtest)
```

```
## Caricamento del pacchetto richiesto: zoo
```

```
##
## Caricamento pacchetto: 'zoo'
```

```
## I seguenti oggetti sono mascherati da 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library("pca3d")
library("ggplot2")
library("factoextra")
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(cluster)
library("hopkins")
library("NbClust")
library("fpc")
library("mclust")
```

```
##                    __           __
##    ____ ___  _____/ /_  _______/ /_
##   / __ `__ \/ ___/ / / / / ___/ __/
##  / / / / / / /__/ / /_/ (__  ) /_
## /_/ /_/ /_/\___/_/\__,_/____/\__/   version 6.0.0
## Type 'citation("mclust")' for citing this R package in publications.
```

```
##
## Caricamento pacchetto: 'mclust'
```

```
## Il seguente oggetto è mascherato da 'package:gamlss.data':
##
##     acidity
```

```
library("clValid")
library("gamlss.mx")
```

```
## Caricamento del pacchetto richiesto: nnet
```

```
##Feature Analysis
data <- read_csv("C:/Users/acer/Downloads/Wholesale customers data.csv")
```

```
## Registered S3 method overwritten by 'bit':
##   method   from
##   print.ri gamlss
```

```
## Rows: 440 Columns: 8
## ── Column specification ────────────────────────────────────────
## Delimiter: ","
## dbl (8): Channel, Region, Fresh, Milk, Gr...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
str(data)
```

```
## spc_tbl_ [440 × 8] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Channel         : num [1:440] 2 2 2 1 2 2 2 2 1 2 ...
##  $ Region          : num [1:440] 3 3 3 3 3 3 3 3 3 3 ...
##  $ Fresh           : num [1:440] 12669 7057 6353 13265 22615 ...
##  $ Milk            : num [1:440] 9656 9810 8808 1196 5410 ...
##  $ Grocery         : num [1:440] 7561 9568 7684 4221 7198 ...
##  $ Frozen          : num [1:440] 214 1762 2405 6404 3915 ...
##  $ Detergents_Paper: num [1:440] 2674 3293 3516 507 1777 ...
##  $ Delicassen      : num [1:440] 1338 1776 7844 1788 5185 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Channel = col_double(),
##   ..   Region = col_double(),
##   ..   Fresh = col_double(),
##   ..   Milk = col_double(),
##   ..   Grocery = col_double(),
##   ..   Frozen = col_double(),
##   ..   Detergents_Paper = col_double(),
##   ..   Delicassen = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
head(data)
```

```
## # A tibble: 6 × 8
##   Channel Region Fresh  Milk Grocery Frozen
##     <dbl>  <dbl> <dbl> <dbl>   <dbl>  <dbl>
## 1       2      3 12669  9656    7561    214
## 2       2      3  7057  9810    9568   1762
## 3       2      3  6353  8808    7684   2405
## 4       1      3 13265  1196    4221   6404
## 5       2      3 22615  5410    7198   3915
## 6       2      3  9413  8259    5126    666
## # … with 2 more variables:
## #   Detergents_Paper <dbl>, Delicassen <dbl>
```

```
dim(data)
```

```
## [1] 440   8
```

```
data$Channel[data$Channel==1]<-"Ho.re.ca"
data$Channel[data$Channel==2]<-"Retail"
data$Region[data$Region==1]<-"Lisbon"
data$Region[data$Region==2]<-"Oporto"
data$Region[data$Region==3]<-"Other"
head(data)
```

```
## # A tibble: 6 × 8
##   Channel  Region Fresh  Milk Grocery Frozen
##   <chr>    <chr>  <dbl> <dbl>   <dbl>  <dbl>
## 1 Retail   Other  12669  9656    7561    214
## 2 Retail   Other   7057  9810    9568   1762
## 3 Retail   Other   6353  8808    7684   2405
## 4 Ho.re.ca Other  13265  1196    4221   6404
## 5 Retail   Other  22615  5410    7198   3915
## 6 Retail   Other   9413  8259    5126    666
## # … with 2 more variables:
## #   Detergents_Paper <dbl>, Delicassen <dbl>
```

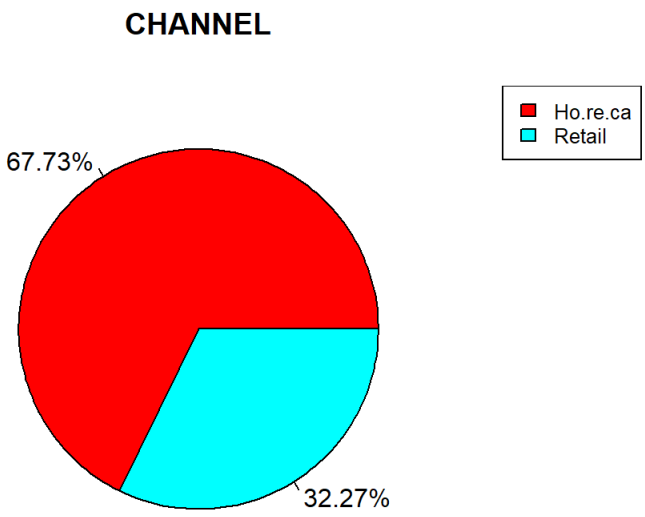
```
data$Channel<-as.factor(data$Channel)
data$Region<-as.factor(data$Region)
summary(data)
```

```
##      Channel       Region
##  Ho.re.ca:298   Lisbon: 77
##  Retail  :142   Oporto: 47
##                 Other :316
##
##
##
##      Fresh            Milk
##  Min.   :    3   Min.   :   55
##  1st Qu.:  3128   1st Qu.: 1533
##  Median :  8504   Median : 3627
##  Mean   : 12000   Mean   : 5796
##  3rd Qu.: 16934   3rd Qu.: 7190
##  Max.   :112151   Max.   :73498
##     Grocery          Frozen
##  Min.   :    3   Min.   :  25.0
##  1st Qu.: 2153   1st Qu.:  742.2
##  Median : 4756   Median : 1526.0
##  Mean   : 7951   Mean   : 3071.9
##  3rd Qu.:10656   3rd Qu.: 3554.2
##  Max.   :92780   Max.   :60869.0
##  Detergents_Paper    Delicassen
##  Min.   :    3.0   Min.   :    3.0
##  1st Qu.:  256.8   1st Qu.:  408.2
##  Median :  816.5   Median :  965.5
##  Mean   : 2881.5   Mean   : 1524.9
##  3rd Qu.: 3922.0   3rd Qu.: 1820.2
##  Max.   :40827.0   Max.   :47943.0
```

```
##About Channel
#is a nominal categorical variable that represents two different Channel of distribution
fac_channel<-as.factor(data$Channel)
summary(fac_channel)
```

```
## Ho.re.ca   Retail
##      298      142
```

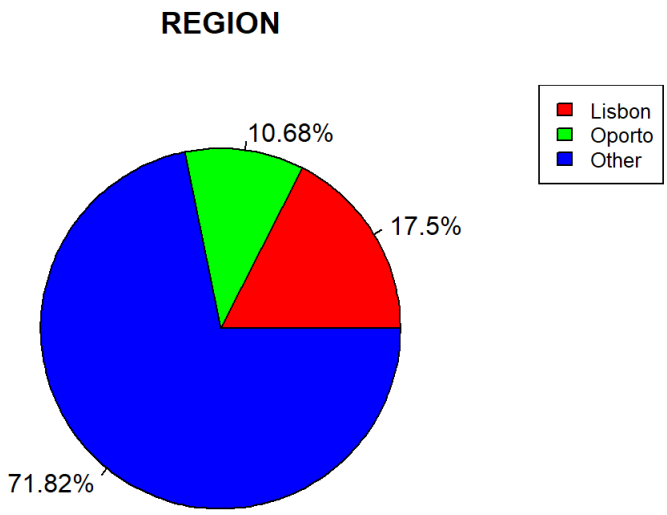
```
pie(table(data$Channel), labels =paste0( round((table(data$Channel)/length(data$Channel))*100,2) , "%") ,main = "CHANNEL",col = rainbow(length(table(
data$Channel))))
legend("topright", names(table(data$Channel)), cex = 0.8,
       fill = rainbow(length(table(data$Channel))))
```




```
##About Region
#is a nominal categorical variable that represents 3 most important Region for this study
fac_region<-as.factor(data$Region)
summary(fac_region)
```

```
## Lisbon Oporto  Other
##     77     47    316
```

```
pie(table(data$Region), labels =paste0( round((table(data$Region)/length(data$Region))*100,2) , "%") ,main = "REGION",col = rainbow(length(table(data$Region))))
legend("topright", names(table(data$Region)), cex = 0.8,
       fill = rainbow(length(table(data$Region))))
```




```
##About Annual Spending on Fresh Product
#is a numerical continuous variable, defined on the interval [0,+∞)
summary(data$Fresh)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.
##       3    3128    8504   12000   16934
##    Max.
##  112151
```

```
kurtosis(data$Fresh)
```

```
## [1] 14.39212
```

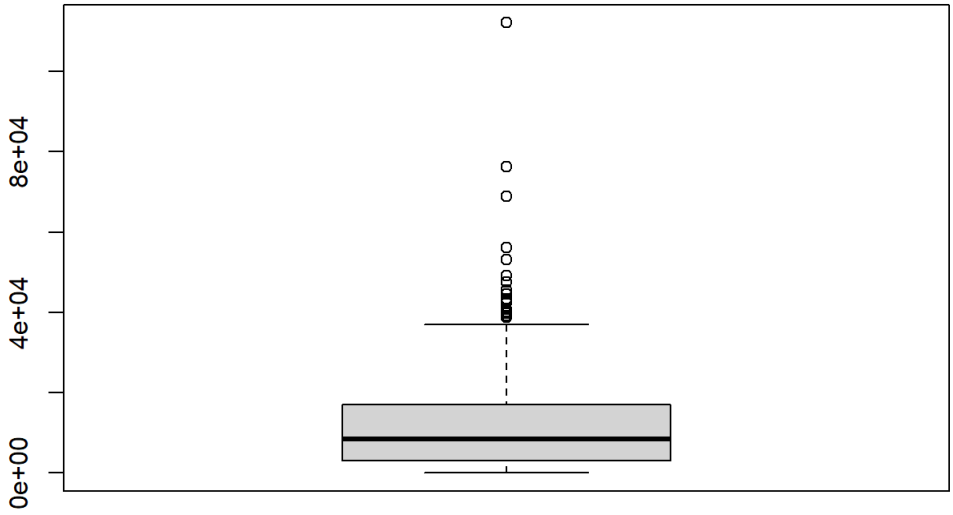
```
#A positive kurtosis value indicates we are dealing with a fat tailed distribution, where extreme outcomes are more common than would be predicted by a s
tandard normal distribution.
#if the kurtosis = 3.0 is a mesokurtic distribution.
#This distribution has a kurtosis similar to that of the normal distribution, meaning the extreme value characteristic of the distribution is similar to that of a no
rmal distribution.
#In this case, k>3 so we have a leptokurtic distribution. Any distribution that is leptokurtic displays greater kurtosis than a mesokurtic distribution.
#This distribution appears as a curve one with long tails (outliers)
#The final type of distribution is platykurtic distribution. These types of distributions have short tails (fewer outliers).
#Platykurtic distributions have demonstrated more stability than other curves

skewness(data$Fresh)
```

```
## [1] 2.552583
```

```
#A positive skewness would indicate a distribution would be biased towards higher values, such that the mean of the distribution will exceed the median of
the distribution.
#Right Skewed distributions greater than 0.8 often indicate the presence of a handful of exceptionally high outliers.

jarque.test(data$Fresh)
```

```
##
##  Jarque-Bera Normality Test
##
## data:  data$Fresh
## JB = 2857.1, p-value < 2.2e-16
## alternative hypothesis: greater
```

```
#For this test the Null Hypothesis is: the dataset has a skewness and kurtosis that matches a normal distribution.
#While Alternative Hypothesis is: the dataset has a skewness and kurtosis that does not match a normal distribution.

box<-boxplot(data$Fresh)
```




```
#The boxplot shows us the presence of many outliers
box$out
```

```
##  [1]  43088  56159  44466  40721  43265
##  [6]  56082  76237  42312  45640 112151
## [11]  47493  56083  53205  49063  68951
## [16]  40254  42786  39679  38793  39228
```

```
data[which(data$Fresh%in% box$out),]
```

```
## # A tibble: 20 × 8
##    Channel Region  Fresh  Milk Grocery Frozen
##    <fct>   <fct>   <dbl> <dbl>   <dbl>  <dbl>
##  1 Ho.re.… Other   43088  2100    2609   1200
##  2 Ho.re.… Other   56159   555     902  10002
##  3 Retail  Other   44466 54259   55571   7782
##  4 Retail  Other   40721  3916    5876    532
##  5 Ho.re.… Other   43265  5025    8117   6312
##  6 Ho.re.… Other   56082  3504    8906  18028
##  7 Ho.re.… Other   76237  3473    7102  16538
##  8 Ho.re.… Other   42312   926    1510   1718
##  9 Ho.re.… Other   45640  6958    6536   7368
## 10 Ho.re.… Other  112151 29627   18148  16745
## 11 Ho.re.… Lisbon  47493  2567    3779   5243
## 12 Ho.re.… Lisbon  56083  4563    2124   6422
## 13 Ho.re.… Lisbon  53205  4959    7336   3012
## 14 Ho.re.… Other   49063  3965    4252   5970
## 15 Ho.re.… Other   68951  4411   12609   8692
## 16 Ho.re.… Other   40254   640    3600   1042
## 17 Ho.re.… Other   42786   286     471   1388
## 18 Retail  Other   39679  3944    4955   1364
## 19 Ho.re.… Other   38793  3154    2648   1034
## 20 Ho.re.… Other   39228  1431     764   4510
## # … with 2 more variables:
## #   Detergents_Paper <dbl>, Delicassen <dbl>
```

```
#Most outliers come from Channel Ho.re.ca, this might be related to the fact that
#Ho.re.ca is characterized by high values of annual spending on Fresh Product than other channel.

table(data$Fresh)
```

```
##
##    3    9   18   23   37   85
##    2    1    1    1    1    1
##   97  117  140  161  180  190
##    1    1    1    1    1    1
##  200  219  243  255  260  286
##    1    1    1    1    1    1
##  327  355  381  403  444  491
##    1    1    1    1    1    1
##  503  514  518  542  572  583
##    1    2    1    1    1    1
##  608  622  630  660  680  688
##    1    1    1    1    1    1
##  694  717  759  796  834  894
##    1    1    1    1    1    1
##  918  955  964 1020 1073 1107
##    1    1    1    1    1    1
## 1182 1198 1206 1210 1289 1406
##    1    1    1    1    1    1
## 1420 1454 1479 1502 1531 1537
##    1    1    1    1    1    1
## 1640 1689 1725 1774 1821 1838
##    1    1    1    1    1    1
## 1869 1956 1982 1989 2083 2101
##    1    1    1    1    1    1
## 2121 2126 2137 2153 2320 2343
##    1    1    1    1    1    1
## 2344 2362 2427 2438 2446 2532
##    1    1    1    1    1    1
## 2541 2599 2612 2615 2617 2647
##    1    1    1    1    1    1
## 2708 2771 2787 2790 2806 2838
##    1    1    1    1    1    1
## 2861 2886 2926 2932 3009 3043
##    1    1    1    1    1    1
## 3047 3062 3067 3087 3097 3103
##    1    1    1    1    1    1
## 3136 3157 3191 3225 3317 3347
##    1    1    1    1    1    1
## 3352 3366 3428 3463 3477 3521
##    1    2    1    1    1    1
## 3830 3884 3996 4020 4042 4048
##    1    1    1    1    1    1
## 4098 4113 4155 4389 4414 4420
##    1    1    1    1    1    1
```

```
##  4446  4456  4484  4515  4591  4625
##     1     1     1     1     1     1
##  4627  4692  4720  4734  4760  4822
##     1     1     1     1     1     1
##  4967  4983  5041  5065  5113  5181
##     1     1     1     1     1     1
##  5224  5264  5283  5387  5396  5414
##     1     1     1     1     1     1
##  5417  5509  5531  5550  5567  5626
##     1     1     1     1     1     1
##  5809  5841  5876  5909  5924  5963
##     1     1     1     1     1     1
##  5969  5981  6006  6022  6134  6137
##     1     1     1     1     1     1
##  6202  6211  6269  6300  6338  6353
##     1     1     1     1     1     1
##  6373  6468  6623  6633  6758  6884
##     1     1     1     1     1     1
##  6987  6990  7005  7034  7057  7107
##     1     1     1     1     1     1
##  7127  7149  7274  7291  7362  7363
##     1     2     1     1     1     1
##  7388  7579  7588  7769  7780  7823
##     1     1     1     1     1     1
##  7842  7858  7864  8040  8090  8170
##     1     1     1     2     1     1
##  8190  8257  8258  8352  8475  8533
##     1     1     1     1     1     1
##  8565  8590  8607  8635  8656  8708
##     1     1     1     1     1     1
##  8797  8861  8885  9061  9155  9193
##     1     1     1     1     1     1
##  9198  9203  9351  9385  9396  9413
##     1     1     1     1     1     1
##  9434  9561  9602  9612  9658  9670
##     1     1     1     1     1     2
##  9759  9784  9785  9790  9898 10253
##     1     1     1     1     1     1
## 10290 10362 10379 10405 10617 10683
##     1     1     1     1     1     1
## 10766 10850 11002 11072 11092 11134
##     1     1     1     1     1     1
## 11170 11173 11210 11223 11243 11314
##     1     1     1     1     1     1
## 11405 11442 11519 11535 11594 11635
##     1     1     1     1     1     1
## 11686 11693 11800 11818 11867 11908
##     1     1     1     1     1     1
## 12119 12126 12205 12212 12238 12356
##     1     1     1     1     1     1
## 12377 12434 12579 12669 12680 12754
##     1     1     1     1     1     1
## 12759 13134 13146 13265 13360 13537
##     1     1     1     1     1     1
## 13569 13624 13779 13970 14039 14100
##     1     1     1     1     1     1
## 14276 14438 14531 14755 14903 15076
##     1     1     1     1     1     1
## 15168 15177 15218 15354 15587 15603
##     1     1     1     1     1     1
## 15615 15671 15881 16117 16165 16225
##     1     1     1     1     1     1
## 16260 16448 16705 16731 16823 16933
##     1     1     1     1     1     1
## 16936 16980 17023 17063 17160 17327
##     1     1     1     1     1     1
## 17360 17546 17565 17623 17770 17773
##     1     1     1     1     1     1
## 18044 18073 18226 18291 18567 18601
##     2     1     1     1     1     1
## 18692 18815 18827 18840 19046 19087
##     1     1     1     1     1     1
## 19176 19219 19746 19899 19913 20049
##     1     1     1     1     1     1
## 20105 20398 20782 20874 20893 20918
##     1     1     1     1     1     1
## 21117 21217 21273 21465 21632 22039
##     1     1     1     1     1     1
```

```
## 22096 22321 22335 22615 22647 22686
##     1     1     1     1     1     1
## 22925 23257 23632 24025 24653 24904
##     1     1     1     1     1     1
## 24929 25066 25203 25606 25767 25962
##     1     1     1     1     1     1
## 25977 26373 26400 26539 27082 27167
##     1     1     1     1     1     1
## 27329 27380 27901 28257 29526 29635
##     1     1     1     1     1     1
## 29703 29729 29955 30379 30624 31012
##     1     1     1     1     1     1
## 31276 31614 31714 31812 32717 34454
##     1     1     1     1     1     1
## 35942 36050 36817 36847 37036 38793
##     1     1     1     1     1     1
## 39228 39679 40254 40721 42312 42786
##     1     1     1     1     1     1
## 43088 43265 44466 45640 47493 49063
##     1     1     1     1     1     1
## 53205 56082 56083 56159 68951 76237
##     1     1     1     1     1     1
## 112151
##      1
```

```
length(unique(data$Fresh))
```

```
## [1] 433
```

```
quantile(data$Fresh, 0.25)
```

```
##     25%
## 3127.75
```

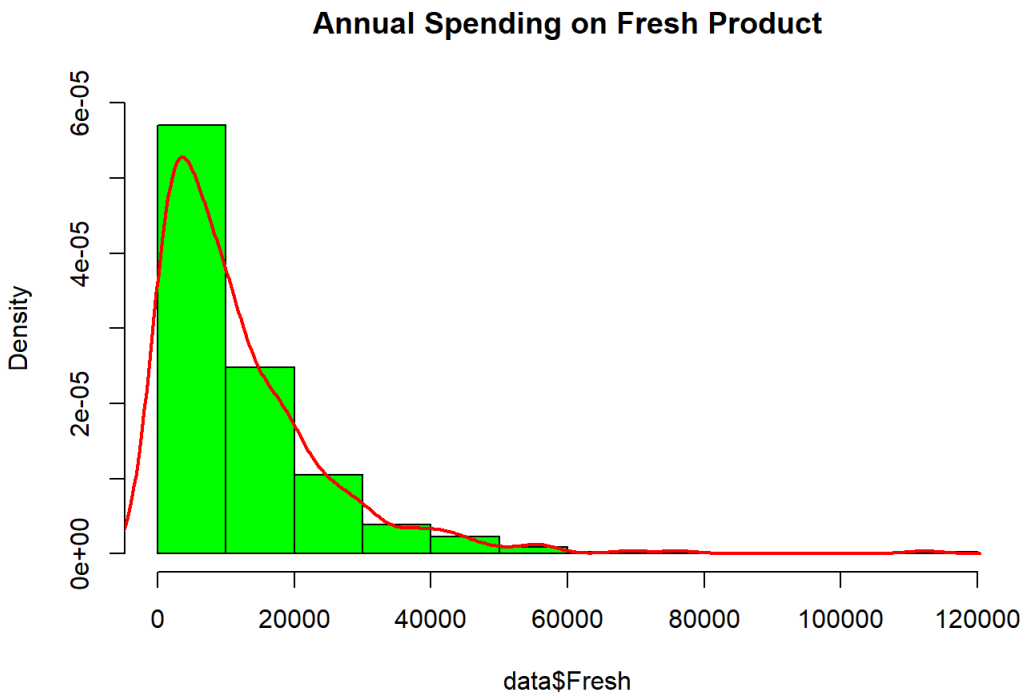
```
quantile(data$Fresh, 0.50)
```

```
##  50%
## 8504
```

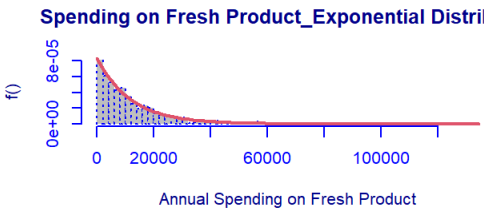
```
quantile(data$Fresh, 0.75)
```

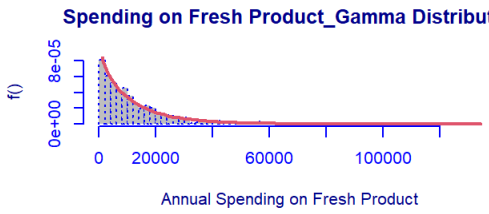
```
##     75%
## 16933.75
```

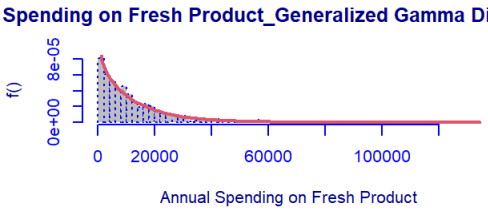
```
hist(data$Fresh, main="Annual Spending on Fresh Product", ylim=c(0,6e-05), col="green", freq = FALSE)
lines(density(data$Fresh), col="red", lwd=2)
```

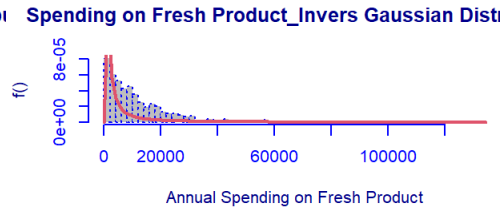


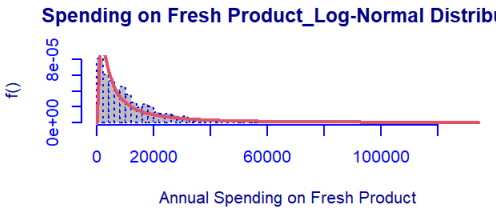

```
##Model Fitting of Fresh
#According to the domain of the variable, different distributions are fitted on the data.
#SBC, AIC and Loglikelihood value are used to evaluate the best fitted model.
par(mfrow=c(3,2))
Fresh.Exp<-histDist(data$Fresh, family=EXP, xlab="Annual Spending on Fresh Product", nbins=50, main="Spending on Fresh Product_Exponential Distributed")
Fresh.Ga<-histDist(data$Fresh, family=GA, xlab="Annual Spending on Fresh Product", nbins=50, main="Spending on Fresh Product_Gamma Distributed")
Fresh.Gg<-histDist(data$Fresh, family=GG, xlab="Annual Spending on Fresh Product", nbins=50, main="Spending on Fresh Product_Generalized Gamma Distributed")
Fresh.Ig<-histDist(data$Fresh, family=IG, xlab="Annual Spending on Fresh Product", nbins=50, main="Spending on Fresh Product_Invers Gaussian Distributed")
Fresh.Logn<-histDist(data$Fresh, family=LOGNO, xlab="Annual Spending on Fresh Product", nbins=50, main="Spending on Fresh Product_Log-Normal Distributed")
Fresh.Wei<-histDist(data$Fresh, family=WEI, xlab="Annual Spending on Fresh Product", nbins=50, main="Spending on Fresh Product_Weibull Distributed")
```

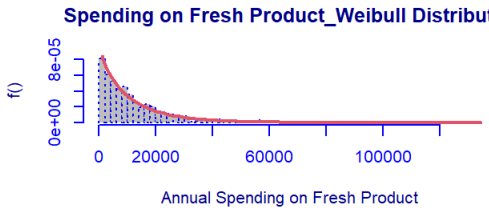












```
data.frame(row.names=c("Exponential","Gamma","Generalized Gamma","Inverse Gaussian","Log-normal","Weibull"),
           df=c(Fresh.Exp$df.fit,Fresh.Ga$df.fit,Fresh.Gg$df.fit,Fresh.Ig$df.fit,Fresh.Logn$df.fit,Fresh.Wei$df.fit),
           AIC=c(AIC(Fresh.Exp),AIC(Fresh.Ga),AIC(Fresh.Gg),AIC(Fresh.Ig),AIC(Fresh.Logn),AIC(Fresh.Wei)),
           SBC=c(Fresh.Exp$sbc,Fresh.Ga$sbc,Fresh.Gg$sbc,Fresh.Ig$sbc,Fresh.Logn$sbc,Fresh.Wei$sbc),
           LogLik=c(logLik(Fresh.Exp),logLik(Fresh.Ga),logLik(Fresh.Gg),logLik(Fresh.Ig),logLik(Fresh.Logn),logLik(Fresh.Wei)))
```

```
##                   df       AIC       SBC
## Exponential        1  9147.564  9151.651
## Gamma              2  9145.068  9153.241
## Generalized Gamma  3  9146.001  9158.261
## Inverse Gaussian   2 10125.260 10133.434
## Log-normal         2  9279.583  9287.757
## Weibull            2  9146.456  9154.630
##                     LogLik
## Exponential       -4572.782
## Gamma             -4570.534
## Generalized Gamma -4570.000
## Inverse Gaussian  -5060.630
## Log-normal        -4637.791
## Weibull           -4571.228
```

```
#According to this dataframe the best Distribution that we can obtain from the AIC criterion is the "Gamma Distribution",
#while for the SBC criterion is the "Exponential Distribution, and for the maximization of the LogLik is the "Generilize Gamma Distribution".
#But it's possible compare other model in order to find the model which best fit our data using the function:
fitDist(data$Fresh,type="realplus")
```

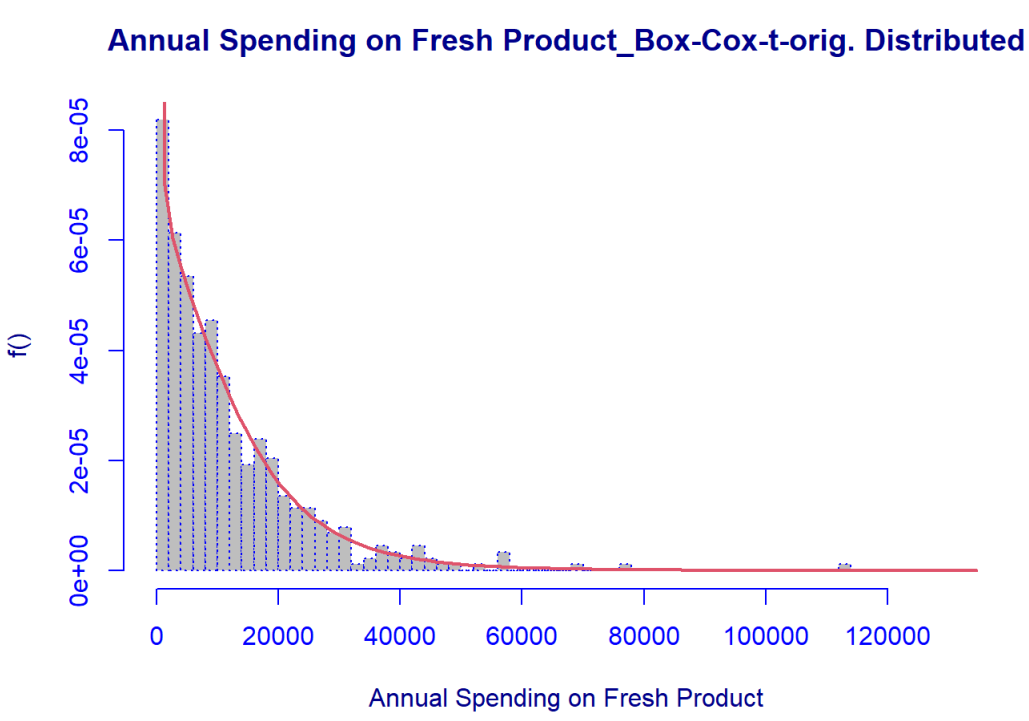
```
##
  |
  |                      |   0%
  |
  |==                    |   4%
  |
  |===                   |   9%
  |
  |=====                 |  13%
  |
  |======                |  17%
  |
  |========              |  22%
  |
  |=========             |  26%
  |
  |===========           |  30%
  |
  |============          |  35%
  |
  |==============        |  39%
  |
  |===============       |  43%
  |
  |=================     |  48%
  |
  |==================    |  52%
  |
  |====================  |  57%
  |
  |====================  |  61%
  |
  |======================|  65%
  |
  |======================|  70%
  |
  |=========================|  74%
```

```
## Warning in qnorm(cdf(q = y, ...)): Si è
## prodotto un NaN
```

```
##
  |
  |============================       | 78%
  |
  |=============================      | 83%
  |
  |==============================     | 87%Error in solve.default(oout$hessian) :
##   Lapack routine dgesv: system is exactly singular: U[4,4] = 0
##
  |
  |================================   | 91%
  |
  |================================== | 96%
  |
  |====================================| 100%
```

```
##
## Family:  c("BCTo", "Box-Cox-t-orig.")
## Fitting method: "nlminb"
##
## Call:
## gamlssML(formula = y, family = DIST[i])
##
## Mu Coefficients:
## [1]  8.556
## Sigma Coefficients:
## [1]  0.6329
## Nu Coefficients:
## [1]  0.671
## Tau Coefficients:
## [1]  1.901
##
##  Degrees of Freedom for the fit: 4 Residual Deg. of Freedom   436
## Global Deviance:     9135.14
##            AIC:     9143.14
##            SBC:     9159.48
```

```
#So, according to "fitDist" function, the best model is "Box-Cox-t-orig."
par(mfrow= c(1,1))
Fresh.BCTo<-histDist(data$Fresh, family=BCTo, xlab="Annual Spending on Fresh Product", nbins=50, main="Annual Spending on Fresh Product_Box-Cox-t-orig. Distributed")
```




```
data.frame(row.names="Box-Cox-t-orig.", df=Fresh.BCTo$df.fit, AIC=AIC(Fresh.BCTo),
           SBC=Fresh.BCTo$sbc, LogLik=logLik(Fresh.BCTo))
```

```
##                df      AIC      SBC
## Box-Cox-t-orig.  4 9143.137 9159.484
##                   LogLik
## Box-Cox-t-orig. -4567.568
```

```
#The "BCTo" distribution minimize the AIC while maximize the LogLik, but the best outcome from SBC is obtained by the Exponential Distribution



##About Annual Spending on Milk Product
#is a numerical continuous variable, defined on the interval [0,+∞)
summary(data$Milk)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.
##      55    1533    3627    5796    7190
##    Max.
##   73498
```

```
kurtosis(data$Milk)
```

```
## [1] 27.37635
```

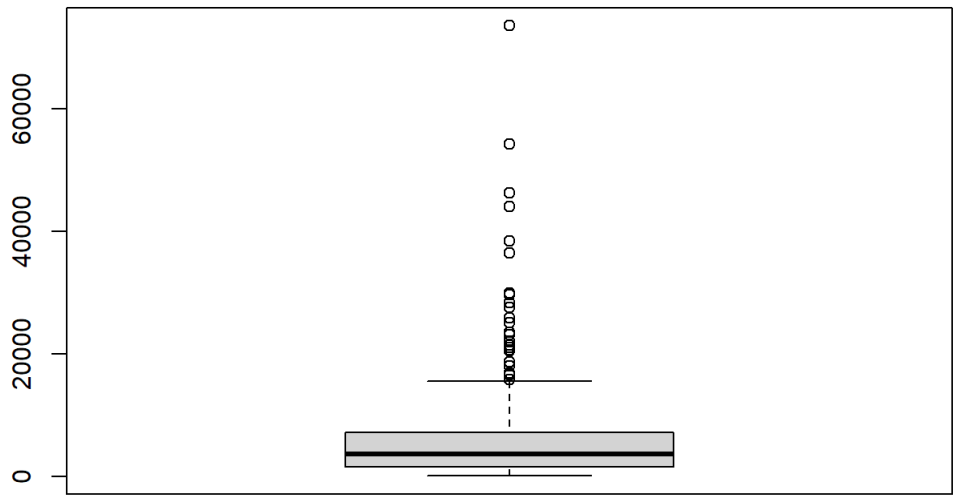
```
#Since the kurtosis is greater than 3, this indicates that the distribution has more values in the tails compared to a normal distribution.
#The "skinniness" of a leptokurtic distribution is a consequence of the outliers, which stretch the horizontal axis of the histogram graph,
#making the bulk of the data appear in a narrow ("skinny") vertical range.
skewness(data$Milk)
```

```
## [1] 4.039922
```

```
#Since the skewness is positive, this indicates that the distribution is right-skewed.
#This confirms what we have seen in the histogram.
par(mfrow= c(1,1))
jarque.test(data$Milk)
```

```
##
##  Jarque-Bera Normality Test
##
## data:  data$Milk
## JB = 12091, p-value < 2.2e-16
## alternative hypothesis: greater
```

```
box<-boxplot(data$Milk)
```



```
box$out
```

```
##  [1] 36423 20484 15729 22044 54259 21412
##  [7] 29892 38369 20959 46197 73498 27472
## [13] 16729 15726 25862 29627 43950 28326
## [19] 16599 23133 17972 23527 20655 25071
## [25] 16784 18664 21858 16687
```

```
data[which(data$Milk%in% box$out),]
```

```
## # A tibble: 28 × 8
##    Channel Region Fresh  Milk Grocery Frozen
##    <fct>   <fct>  <dbl> <dbl>   <dbl>  <dbl>
##  1 Retail  Other  26373 36423   22019   5154
##  2 Retail  Other   4113 20484   25957   1158
##  3 Retail  Other   4591 15729   16709     33
##  4 Retail  Other   5181 22044   21531   1740
##  5 Retail  Other  44466 54259   55571   7782
##  6 Retail  Other   4967 21412   28921   1798
##  7 Retail  Other   4098 29892   26866   2616
##  8 Retail  Other  35942 38369   59598   3254
##  9 Retail  Other     85 20959   45828     36
## 10 Retail  Other  16117 46197   92780   1026
## # … with 18 more rows, and 2 more variables:
## #   Detergents_Paper <dbl>, Delicassen <dbl>
```

```
#Most outliers come from Channel Retail,
#this might be related to the fact that the Retail is characterized by high values
#of annual spending score on Milk Product than the other Channel.
table(data$Milk)
```

```
##
##   55  112  134  201  254  258  286
##    1    1    1    1    1    1    1
##  295  333  337  346  367  475  489
##    1    1    1    1    1    1    1
##  521  540  542  555  577  584  589
##    1    1    1    1    2    1    1
##  594  596  607  640  659  685  713
##    1    1    1    1    2    1    1
##  717  727  735  780  793  803  820
##    1    1    1    1    1    1    1
##  829  848  865  871  873  879  891
##    2    1    1    1    1    1    1
##  894  899  906  918  922  925  926
##    1    2    1    1    1    1    1
##  928  944  961  997 1012 1014 1020
##    1    2    1    1    2    1    1
## 1032 1042 1046 1080 1095 1099 1106
##    2    1    1    1    1    1    1
## 1110 1114 1115 1124 1137 1162 1172
##    1    1    2    1    1    1    1
## 1175 1181 1188 1196 1200 1208 1222
##    1    1    1    2    1    1    1
## 1227 1266 1275 1289 1304 1316 1318
##    1    1    1    1    1    1    1
## 1347 1364 1366 1371 1372 1375 1377
##    1    1    1    1    1    1    1
## 1431 1433 1449 1450 1461 1475 1486
##    1    1    1    1    1    1    1
## 1492 1511 1530 1534 1563 1596 1601
##    1    1    1    1    1    1    1
## 1610 1642 1648 1666 1698 1703 1750
##    2    1    1    1    1    1    1
## 1780 1786 1795 1801 1825 1840 1860
##    1    1    1    1    1    1    1
## 1882 1887 1891 1895 1897 1916 1917
##    1    1    1    1    2    1    1
## 1931 1936 1970 1979 1981 1990 1993
##    1    1    1    1    1    1    1
## 2013 2024 2032 2037 2096 2100 2102
##    1    1    1    1    1    1    1
## 2132 2154 2160 2182 2204 2209 2217
##    1    1    1    1    1    1    1
```

```
## 2247 2256 2280 2295 2317 2335 2344
##    1    1    1    1    1    1    1
## 2374 2380 2408 2428 2495 2521 2527
##    1    1    1    2    1    1    1
## 2567 2602 2616 2703 2713 2746 2761
##    1    1    1    1    1    1    1
## 2762 2770 2801 2820 2872 2884 2920
##    1    1    1    1    1    2    1
## 3045 3065 3086 3088 3090 3154 3195
##    2    1    1    1    1    1    1
## 3199 3216 3218 3234 3243 3259 3289
##    2    1    1    1    1    1    1
## 3294 3327 3328 3354 3373 3473 3485
##    1    1    1    1    1    1    1
## 3504 3575 3576 3587 3605 3610 3613
##    1    1    1    2    1    1    1
## 3620 3634 3648 3651 3677 3683 3686
##    1    1    1    1    1    1    1
## 3688 3696 3737 3748 3749 3783 3795
##    1    1    1    1    1    1    1
## 3801 3836 3838 3880 3916 3922 3944
##    1    1    1    2    1    1    1
## 3965 4025 4051 4180 4230 4257 4280
##    1    1    1    1    2    1    1
## 4332 4339 4362 4411 4519 4560 4563
##    1    1    1    1    1    1    1
## 4591 4613 4737 4753 4786 4847 4859
##    1    1    1    1    1    1    1
## 4885 4888 4956 4959 4979 4980 4984
##    1    1    1    1    1    1    1
## 5007 5008 5010 5025 5139 5164 5266
##    1    1    1    1    2    1    1
## 5279 5291 5302 5332 5360 5403 5410
##    1    1    1    1    1    1    1
## 5479 5491 5499 5506 5758 5763 5878
##    1    1    1    1    1    1    1
## 5921 5970 5989 6036 6046 6128 6152
##    1    1    1    1    1    1    1
## 6154 6157 6200 6208 6243 6245 6250
##    1    1    1    1    1    1    1
## 6257 6264 6327 6337 6343 6380 6448
##    1    1    1    1    1    1    1
## 6459 6551 6570 6602 6721 6730 6759
##    1    1    1    1    1    1    1
## 6817 6939 6958 6964 7027 7075 7097
##    1    1    1    1    1    1    1
## 7108 7152 7184 7209 7243 7260 7330
##    1    1    1    1    1    1    1
## 7393 7435 7460 7503 7504 7555 7603
##    1    1    1    1    1    1    1
## 7639 7677 7704 7775 7779 7845 7961
##    1    1    1    1    1    1    1
## 8002 8053 8080 8259 8323 8384 8397
##    1    1    1    1    1    1    1
## 8494 8533 8579 8630 8675 8808 8816
##    1    1    1    1    1    1    1
## 8847 9232 9250 9347 9465 9540 9656
##    1    1    1    1    1    1    1
## 9679 9763 9776 9810 9933 10044 10473
##    1    1    1    1    1    1    1
## 10556 10646 10678 10690 10765 10769 10810
##    1    1    1    1    1    1    1
## 10940 11006 11093 11095 11103 11114 11487
##    1    1    1    1    1    1    1
## 11577 11601 11711 11991 12051 12220 12319
##    1    1    1    1    1    1    1
## 12653 12697 12729 12844 12867 12939 13240
##    1    1    1    1    1    1    1
## 13252 13316 14069 14399 14641 14881 14982
##    1    1    1    1    1    1    1
## 15488 15726 15729 16599 16687 16729 16784
##    1    1    1    1    1    1    1
## 17972 18664 20484 20655 20959 21412 21858
##    1    1    1    1    1    1    1
## 22044 23133 23527 25071 25862 27472 28326
##    1    1    1    1    1    1    1
## 29627 29892 36423 38369 43950 46197 54259
##    1    1    1    1    1    1    1
```

```
## 73498
##    1
```

```
length(unique(data$Milk))
```

```
## [1] 421
```

```
quantile(data$Milk, 0.25)
```

```
##  25%
## 1533
```

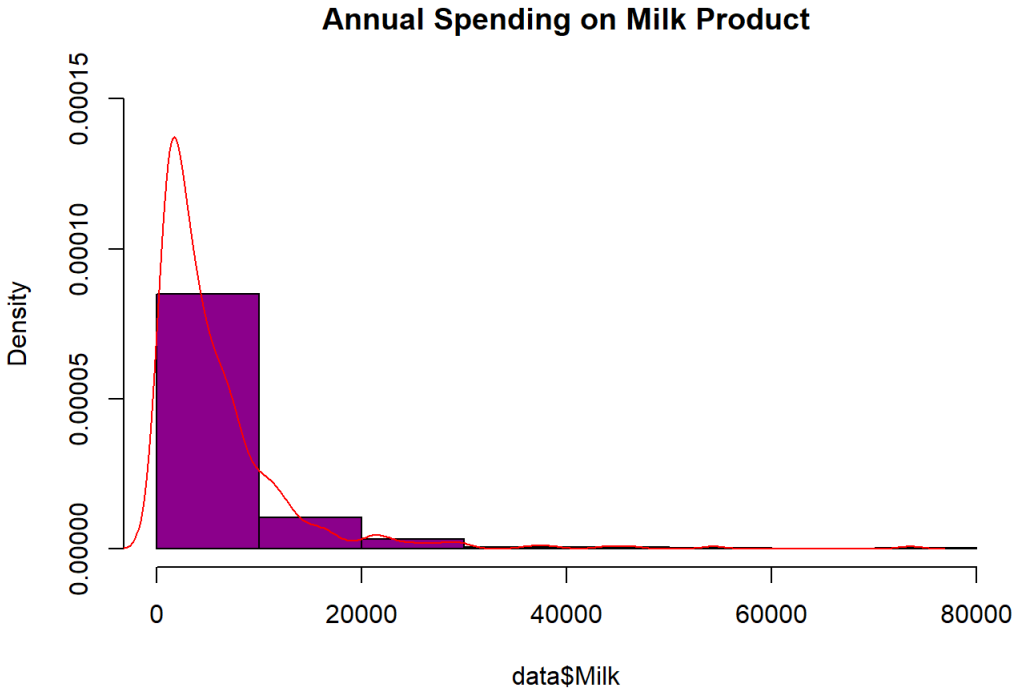
```
quantile(data$Milk, 0.50)
```

```
##  50%
## 3627
```

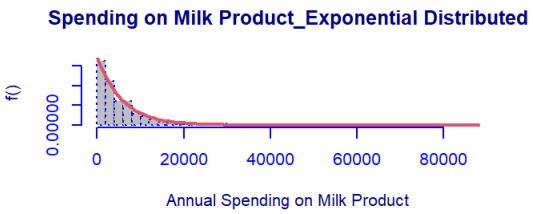
```
quantile(data$Milk, 0.75)
```

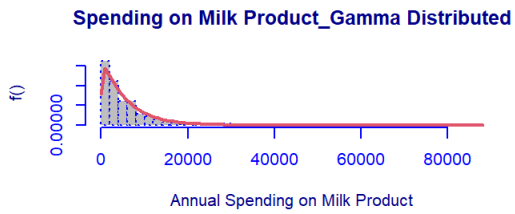
```
##    75%
## 7190.25
```

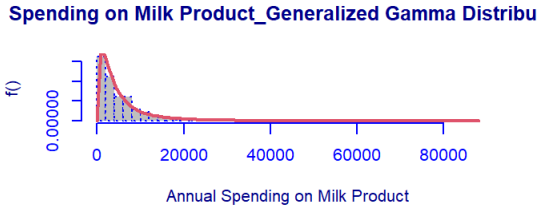
```
par(mfrow= c(1,1))
hist(data$Milk, main="Annual Spending on Milk Product", ylim=c(0,1.5e-04), col="darkmagenta", freq = FALSE)
lines(density(data$Milk), col="red")
```

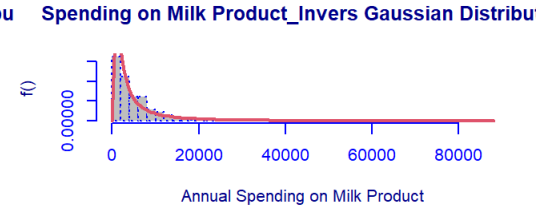


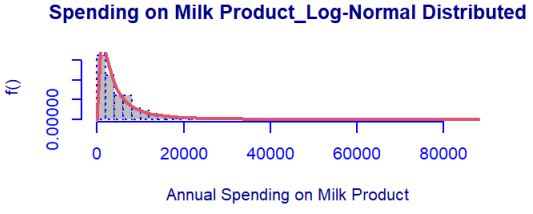

```
#Model Fitting of Milk Product
par(mfrow=c(3,2))
Milk.Exp<-histDist(data$Milk, family=EXP, xlab="Annual Spending on Milk Product", nbins=50, main="Spending on Milk Product_Exponential Distributed")
Milk.Ga<-histDist(data$Milk, family=GA, xlab="Annual Spending on Milk Product", nbins=50, main="Spending on Milk Product_Gamma Distributed")
Milk.Gg<-histDist(data$Milk, family=GG, xlab="Annual Spending on Milk Product", nbins=50, main="Spending on Milk Product_Generalized Gamma Distributed")
Milk.Ig<-histDist(data$Milk, family=IG, xlab="Annual Spending on Milk Product", nbins=50, main="Spending on Milk Product_Invers Gaussian Distributed")
Milk.Logn<-histDist(data$Milk, family=LOGNO, xlab="Annual Spending on Milk Product", nbins=50, main="Spending on Milk Product_Log-Normal Distributed")
Milk.Wei<-histDist(data$Milk, family=WEI, xlab="Annual Spending on Milk Product", nbins=50, main="Spending on Milk Product_Weibull Distributed")
```

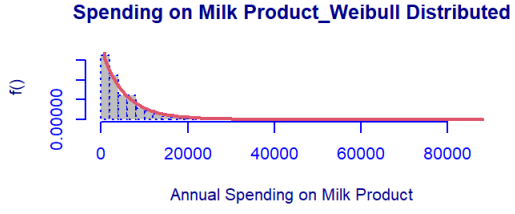













```
data.frame(row.names=c("Exponential","Gamma","Generalized Gamma","Inverse Gaussian","Log-normal","Weibull"),
           df=c(Milk.Exp$df.fit,Milk.Ga$df.fit,Milk.Gg$df.fit,Milk.Ig$df.fit,Milk.Logn$df.fit,Milk.Wei$df.fit),
           AIC=c(AIC(Milk.Exp),AIC(Milk.Ga),AIC(Milk.Gg),AIC(Milk.Ig),AIC(Milk.Logn),AIC(Milk.Wei)),
           SBC=c(Milk.Exp$sbc,Milk.Ga$sbc,Milk.Gg$sbc,Milk.Ig$sbc,Milk.Logn$sbc,Milk.Wei$sbc),
           LogLik=c(logLik(Milk.Exp),logLik(Milk.Ga),logLik(Milk.Gg),logLik(Milk.Ig),logLik(Milk.Logn),logLik(Milk.Wei)))
```

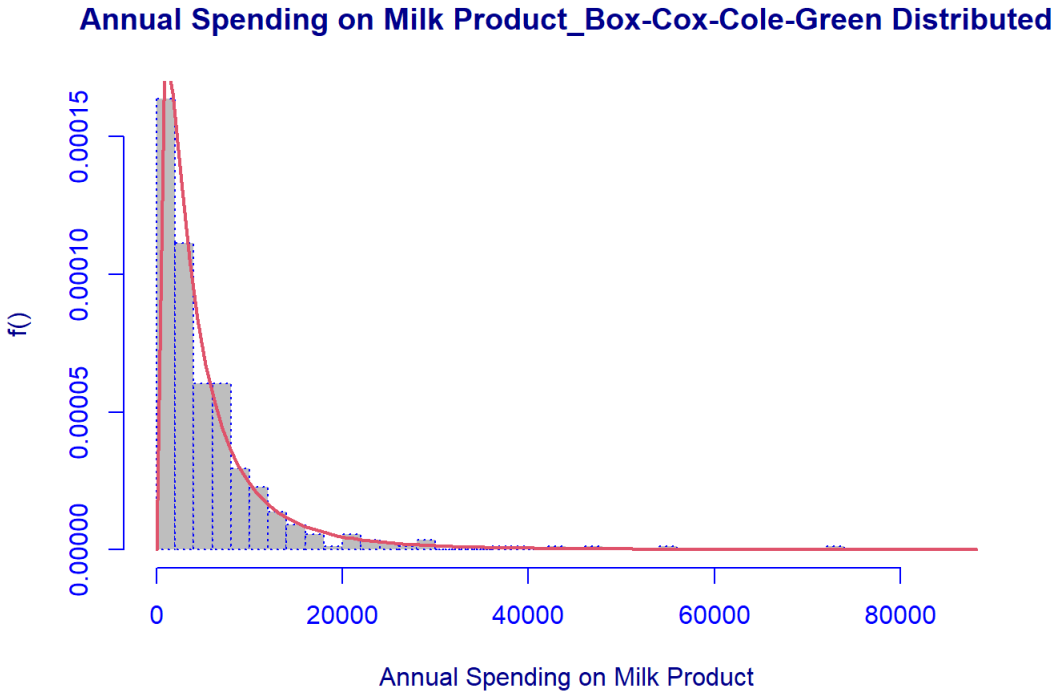
```
##                   df      AIC      SBC
## Exponential        1 8507.173 8511.260
## Gamma              2 8508.387 8516.561
## Generalized Gamma  3 8465.476 8477.736
## Inverse Gaussian   2 8526.163 8534.337
## Log-normal         2 8467.024 8475.197
## Weibull            2 8508.514 8516.687
##                     LogLik
## Exponential       -4252.586
## Gamma             -4252.194
## Generalized Gamma -4229.738
## Inverse Gaussian  -4261.082
## Log-normal        -4231.512
## Weibull           -4252.257
```

```
#According to the AIC and the LogLik criterions the best model is the "Generalized Gamma Distribution",
#while for the SBC criterion the best model is Log-normal.

#But it's possible compare other model in order to find the model which best fit our data using the function:
fitDist(data$Milk,type="realplus")
```

```
##
  |
  |                      |  0%
  |
  |==                    |  4%
  |
  |===                   |  9%
  |
  |=====                 | 13%
  |
  |======                | 17%
  |
  |========              | 22%
  |
  |=========             | 26%
  |
  |===========           | 30%
  |
  |============          | 35%
  |
  |==============        | 39%
  |
  |===============       | 43%
  |
  |=================     | 48%
  |
  |==================    | 52%
  |
  |====================  | 57%
  |
  |====================  | 61%
  |
  |======================| 65%
  |
  |======================| 70%
  |
  |=========================| 74%
  |
  |=========================| 78%
  |
  |============================| 83%
  |
  |============================| 87%Error in solve.default(oout$hessian) :
##   Lapack routine dgesv: system is exactly singular: U[4,4] = 0
##
  |
  |================================| 91%
  |
  |================================| 96%
  |
  |================================| 100%
```

```
##
## Family:  c("BCCG", "Box-Cox-Cole-Green")
## Fitting method: "nlminb"
##
## Call:
## gamlssML(formula = y, family = DIST[i])
##
## Mu Coefficients:
## [1]  3498
## Sigma Coefficients:
## [1]  0.07042
## Nu Coefficients:
## [1]  0.06693
##
##  Degrees of Freedom for the fit: 3 Residual Deg. of Freedom   437
## Global Deviance:     8459.45
##            AIC:     8465.45
##            SBC:     8477.71
```

```
#So, according to "fitDist" function, the best model is "Box-Cox-Cole-Green"
par(mfrow= c(1,1))
Milk.BCCG<-histDist(data$Milk, family=BCCG, xlab="Annual Spending on Milk Product", nbins=50, main="Annual Spending on Milk Product_Box-Cox-Cole-Green Distributed")
```




```
data.frame(row.names="Box-Cox-Cole-Green", df=Milk.BCCG$df.fit, AIC=AIC(Milk.BCCG),
           SBC=Milk.BCCG$sbc, LogLik=logLik(Milk.BCCG))
```

```
##                      df      AIC      SBC
## Box-Cox-Cole-Green  3 8465.448 8477.708
##                        LogLik
## Box-Cox-Cole-Green -4229.724
```

```
#The "BCCG" distribution minimize the AIC while maximize the LogLik,
#but the best outcome according to SBC criterion is obtained by the Log-normal Distribution


##About Annual Spending on Grocery Product
#is a numerical continuous variable, defined on the interval [0,+∞)
summary(data$Grocery)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.
##       3    2153    4756    7951   10656
##    Max.
##   92780
```

```
kurtosis(data$Grocery)
```

```
## [1] 23.66415
```

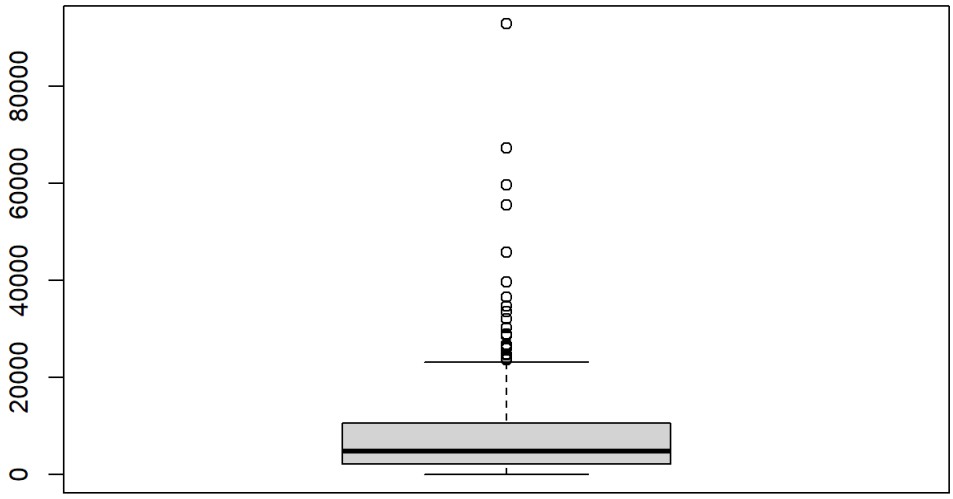
```
skewness(data$Grocery)
```

```
## [1] 3.575187
```

```
jarque.test(data$Grocery)
```

```
##
##  Jarque-Bera Normality Test
##
## data:  data$Grocery
## JB = 8765.8, p-value < 2.2e-16
## alternative hypothesis: greater
```

```
par(mfrow= c(1,1))
box<-boxplot(data$Grocery)
```




```
box$out
```

```
##  [1] 25957 23998 55571 28921 26866 59598
##  [7] 45828 28540 92780 32114 32034 28986
## [13] 34792 26870 24708 23596 39694 36486
## [19] 33586 24773 26839 67298 26316 30243
```

```
data[which(data$Grocery%in% box$out),]
```

```
## # A tibble: 24 × 8
##    Channel Region Fresh  Milk Grocery Frozen
##    <fct>   <fct>  <dbl> <dbl>   <dbl>  <dbl>
##  1 Retail  Other   4113 20484   25957   1158
##  2 Retail  Other    630 11095   23998    787
##  3 Retail  Other  44466 54259   55571   7782
##  4 Retail  Other   4967 21412   28921   1798
##  5 Retail  Other   4098 29892   26866   2616
##  6 Retail  Other  35942 38369   59598   3254
##  7 Retail  Other     85 20959   45828     36
##  8 Retail  Other  12205 12697   28540    869
##  9 Retail  Other  16117 46197   92780   1026
## 10 Retail  Other  22925 73498   32114    987
## # … with 14 more rows, and 2 more variables:
## #   Detergents_Paper <dbl>, Delicassen <dbl>
```

```
#In this case all the outliers come from the Retail channel,
#this might be releated that the Retail is characterized by high values of annual spending score on Grocery Product than the other Channel.
table(data$Grocery)
```

```
##
##    3  137  218  223  245  283  314
##    1    1    1    1    1    1    1
##  471  489  534  542  572  582  585
##    1    1    1    1    1    1    1
##  593  610  683  699  764  778  803
##    1    1    2    1    1    1    1
##  854  864  902  935  950  975 1094
##    1    1    1    1    1    1    1
## 1096 1138 1139 1162 1167 1172 1235
##    1    1    1    1    1    1    1
## 1238 1242 1263 1293 1296 1328 1330
##    1    1    1    1    1    1    1
## 1381 1382 1390 1393 1422 1428 1431
##    1    1    1    1    1    1    1
## 1447 1493 1495 1498 1499 1510 1524
##    1    2    1    1    1    1    1
## 1533 1563 1573 1614 1617 1638 1641
```

```
##    1    2    1    1    1    1    1
## 1648 1651 1658 1660 1664 1668 1677
##    1    1    1    1    2    1    1
## 1694 1733 1763 1765 1777 1783 1799
##    1    1    1    1    1    1    1
## 1841 1883 1902 1909 1939 1947 1980
##    1    1    1    1    1    1    1
## 1981 1988 1997 2000 2006 2010 2012
##    1    1    1    1    1    1    1
## 2013 2022 2028 2046 2060 2062 2067
##    1    1    1    1    1    2    1
## 2070 2109 2112 2124 2128 2146 2147
##    1    1    1    1    1    1    1
## 2155 2157 2174 2177 2201 2216 2223
##    1    1    1    1    1    1    1
## 2232 2251 2262 2280 2313 2362 2368
##    1    1    1    1    1    1    1
## 2405 2406 2431 2453 2464 2469 2474
##    2    2    1    1    1    1    1
## 2475 2479 2500 2501 2510 2530 2543
##    1    1    1    1    1    1    1
## 2548 2576 2591 2593 2609 2611 2642
##    1    1    1    1    1    1    1
## 2648 2661 2707 2743 2824 2828 2842
##    1    1    1    1    1    1    1
## 2856 2857 2861 2886 2914 2933 2961
##    1    1    1    1    1    1    1
## 2974 2988 3007 3009 3045 3047 3133
##    1    1    1    1    1    1    1
## 3135 3202 3204 3250 3261 3268 3281
##    1    1    1    1    1    1    1
## 3315 3316 3343 3355 3389 3412 3417
##    1    1    1    1    1    1    1
## 3444 3445 3450 3558 3580 3600 3643
##    1    1    1    1    1    2    1
## 3655 3779 3810 3821 3823 3828 3833
##    1    1    1    1    1    1    1
## 3838 3842 3970 3993 4042 4157 4172
##    1    1    1    1    1    1    1
## 4221 4252 4329 4438 4469 4523 4533
##    1    1    1    1    1    1    1
## 4563 4583 4602 4604 4657 4710 4740
##    1    1    1    1    1    1    1
## 4748 4754 4757 4814 4897 4910 4945
##    1    1    1    1    1    1    1
## 4955 5005 5026 5034 5091 5109 5119
##    1    1    1    1    1    1    1
## 5126 5160 5167 5189 5226 5230 5234
##    1    1    1    1    1    1    1
## 5241 5249 5265 5330 5332 5380 5428
##    1    1    1    1    1    1    1
## 5429 5615 5838 5876 5923 5956 6089
##    1    1    1    1    1    1    1
## 6100 6114 6192 6235 6252 6360 6407
##    1    1    1    1    1    1    1
## 6532 6536 6544 6550 6633 6684 6824
##    1    2    1    1    1    1    1
## 6861 6869 6964 6975 6981 6986 6996
##    1    1    1    1    1    1    1
## 7021 7030 7041 7102 7198 7305 7326
##    1    1    1    1    1    1    1
## 7336 7398 7417 7561 7595 7647 7677
##    1    1    1    1    1    1    1
## 7684 7854 7994 8025 8040 8117 8118
##    1    1    1    1    1    1    1
## 8253 8280 8282 8335 8469 8552 8584
##    1    1    1    1    1    1    1
## 8713 8814 8852 8887 8906 9053 9170
##    1    1    1    1    1    1    1
## 9212 9345 9426 9464 9490 9568 9618
##    1    1    1    1    1    1    1
## 9670 9694 9785 9794 9819 9965 10099
##    1    1    1    1    1    1    1
## 10391 10471 10487 10518 10646 10685 10704
##    2    1    1    1    1    1    1
## 10790 10817 10868 10908 11009 11055 11091
##    1    1    1    1    1    1    1
## 11107 11238 11323 11364 11522 11532 11593
```

```
##     1     1     1     1     1     1     1
## 11687 11757 11874 11924 12091 12121 12144
##     1     1     1     1     1     1     1
## 12232 12311 12400 12469 12477 12609 12822
##     1     1     1     1     1     1     1
## 12974 13227 13430 13462 13567 13586 13626
##     1     1     1     1     1     1     1
## 13699 13792 13829 13916 14316 14403 14682
##     1     1     1     1     1     1     1
## 14855 14886 14961 14982 15205 15400 15445
##     1     1     1     1     1     1     1
## 15538 15541 15775 16027 16267 16483 16709
##     1     1     1     1     1     1     1
## 16767 16966 17569 17645 18148 18622 18683
##     1     1     1     1     1     1     1
## 18881 19172 19460 19805 19816 19847 19858
##     1     1     1     1     1     1     1
## 20170 20292 20399 20521 21042 21203 21531
##     1     1     1     1     1     1     1
## 21570 21955 22019 22182 22272 22294 23127
##     1     1     1     1     1     1     1
## 23596 23998 24708 24773 25957 26316 26839
##     1     1     1     1     1     1     1
## 26866 26870 28540 28921 28986 30243 32034
##     1     1     1     1     1     1     1
## 32114 33586 34792 36486 39694 45828 55571
##     1     1     1     1     1     1     1
## 59598 67298 92780
##     1     1     1
```

```
length(unique(data$Grocery))
```

```
## [1] 430
```

```
quantile(data$Grocery, 0.25)
```

```
##  25%
## 2153
```

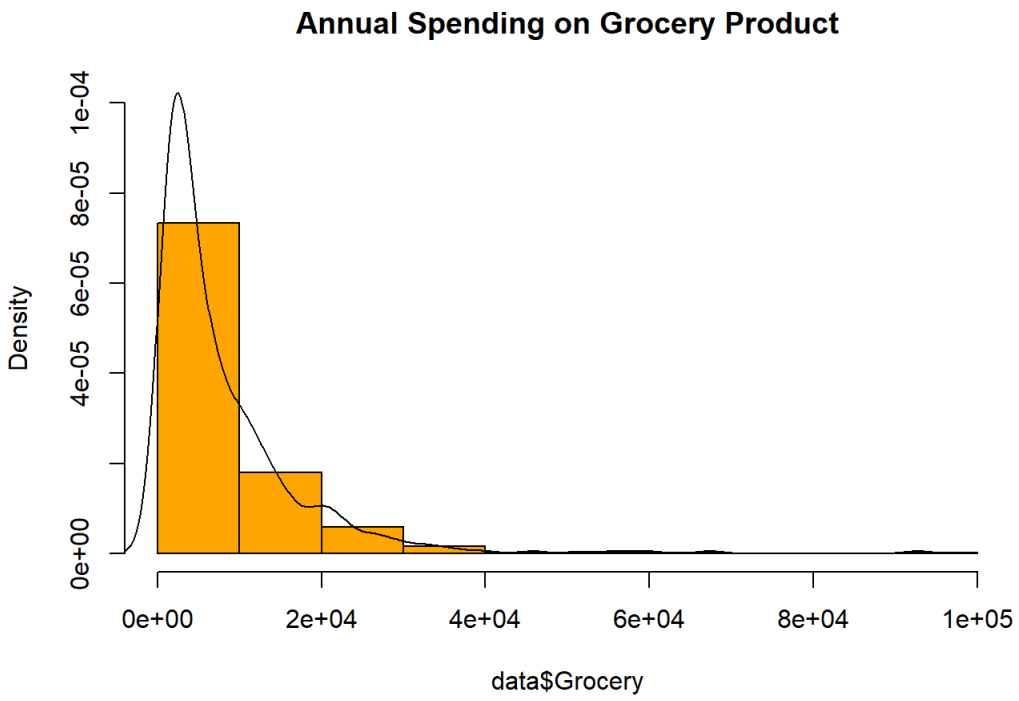
```
quantile(data$Grocery, 0.50)
```

```
##    50%
## 4755.5
```

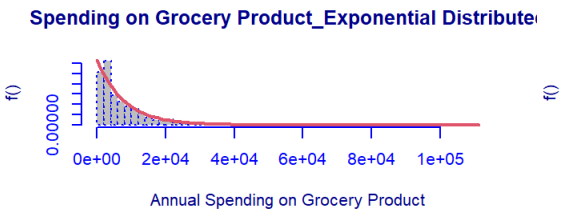
```
quantile(data$Grocery, 0.75)
```

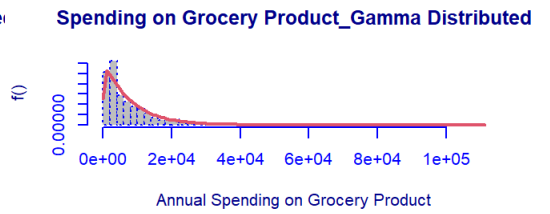
```
##      75%
## 10655.75
```

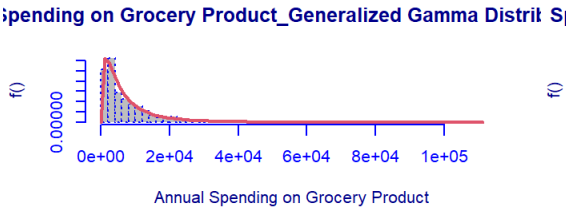
```
par(mfrow= c(1,1))
hist(data$Grocery, main="Annual Spending on Grocery Product", ylim=c(0,1e-04), col="orange", freq = FALSE)
lines(density(data$Grocery), col="black")
```

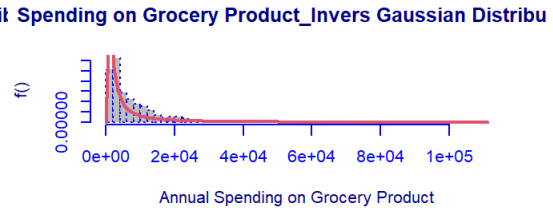


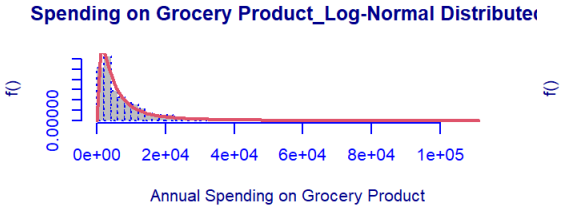

```
#Model Fitting of Grocery Product
par(mfrow=c(3,2))
Grocery.Exp<-histDist(data$Grocery, family=EXP, xlab="Annual Spending on Grocery Product", nbins=50, main="Spending on Grocery Product_Exponential Distributed")
Grocery.Ga<-histDist(data$Grocery, family=GA, xlab="Annual Spending on Grocery Product", nbins=50, main="Spending on Grocery Product_Gamma Distributed")
Grocery.Gg<-histDist(data$Grocery, family=GG, xlab="Annual Spending on Grocery Product", nbins=50, main="Spending on Grocery Product_Generalized Gamma Distributed")
Grocery.Ig<-histDist(data$Grocery, family=IG, xlab="Annual Spending on Grocery Product", nbins=50, main="Spending on Grocery Product_Invers Gaussian Distributed")
Grocery.Logn<-histDist(data$Grocery, family=LOGNO, xlab="Annual Spending on Grocery Product", nbins=50, main="Spending on Grocery Product_Log-Normal Distributed")
Grocery.Wei<-histDist(data$Grocery, family=WEI, xlab="Annual Spending on Grocery Product", nbins=50, main="Spending on Grocery Product_Weibull Distributed")
```

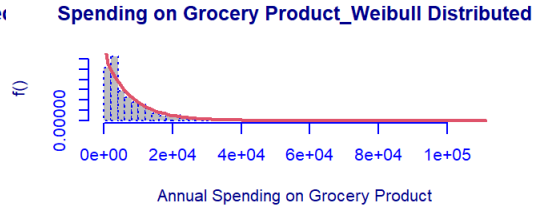













```
data.frame(row.names=c("Exponential","Gamma","Generalized Gamma","Inverse Gaussian","Log-normal","Weibull"),
        df=c(Grocery.Exp$df.fit,Grocery.Ga$df.fit,Grocery.Gg$df.fit,Grocery.Ig$df.fit,Grocery.Logn$df.fit,Grocery.Wei$df.fit),
        AIC=c(AIC(Grocery.Exp),AIC(Grocery.Ga),AIC(Grocery.Gg),AIC(Grocery.Ig),AIC(Grocery.Logn),AIC(Grocery.Wei)),
        SBC=c(Grocery.Exp$sbc,Grocery.Ga$sbc,Grocery.Gg$sbc,Grocery.Ig$sbc,Grocery.Logn$sbc,Grocery.Wei$sbc),
        LogLik=c(logLik(Grocery.Exp),logLik(Grocery.Ga),logLik(Grocery.Gg),logLik(Grocery.Ig),logLik(Grocery.Logn),logLik(Grocery.Wei)))
```

```
##                  df      AIC      SBC
## Exponential       1 8785.357 8789.444
## Gamma             2 8786.366 8794.540
## Generalized Gamma  3 8759.067 8771.327
## Inverse Gaussian   2 9361.916 9370.090
## Log-normal        2 8776.609 8784.783
## Weibull           2 8787.198 8795.371
##                       LogLik
## Exponential       -4391.679
## Gamma             -4391.183
## Generalized Gamma -4376.533
## Inverse Gaussian  -4678.958
## Log-normal        -4386.305
## Weibull           -4391.599
```

```
#According to the criterions the best model is the "Generalized Gamma".
#But it's possible compare other model in order to find the model which best fit our data using the function:
fitDist(data$Grocery, type="realplus")
```

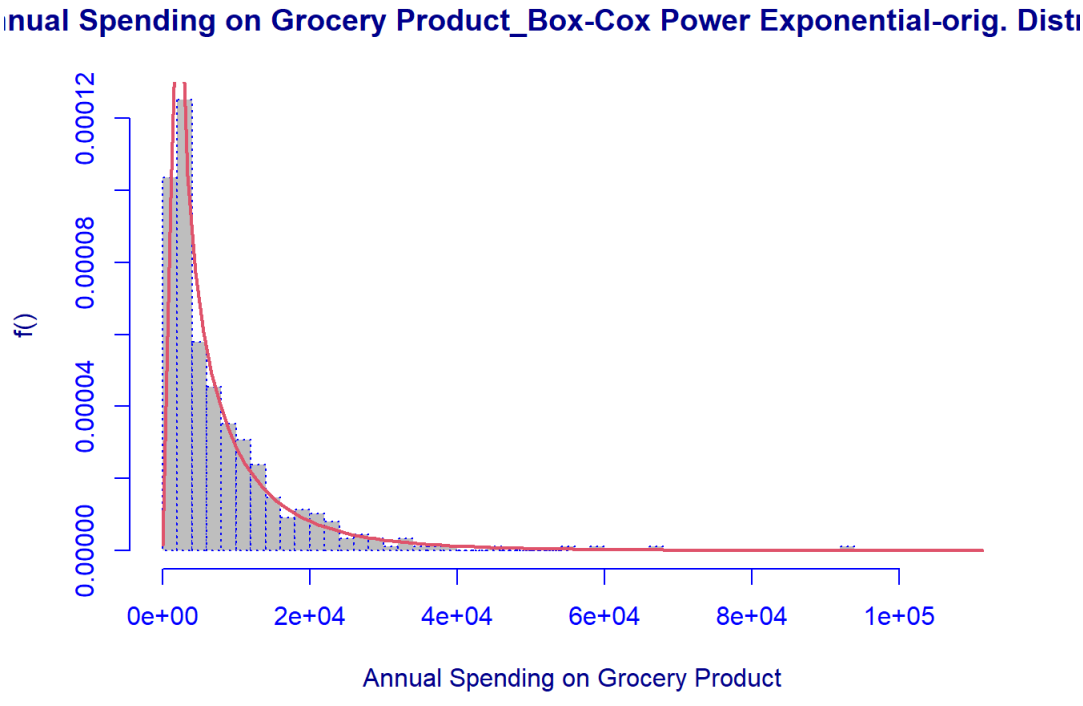
```
##
  |
  |                            |   0%
  |
  |==                          |   4%
  |
  |===                         |   9%
  |
  |=====                       |  13%
  |
  |======                      |  17%
  |
  |========                    |  22%
  |
  |=========                   |  26%
  |
  |===========                 |  30%
  |
  |============                |  35%
  |
  |==============              |  39%
  |
  |================            |  43%
  |
  |==================          |  48%
  |
  |===================         |  52%
  |
  |=====================       |  57%
  |
  |======================      |  61%
  |
  |========================    |  65%
  |
  |=========================   |  70%
  |
  |============================|  74%
```

```
## Warning in qnorm(cdf(q = y, ...)): Si è
## prodotto un NaN
```

```
##
  |
  |============================       | 78%
  |
  |=============================      | 83%
  |
  |==============================     | 87%Error in solve.default(oout$hessian) :
##   Lapack routine dgesv: system is exactly singular: U[4,4] = 0
##
  |
  |================================   | 91%
  |
  |================================== | 96%
  |
  |==================================| 100%
```

```
##
## Family:  c("BCPEo", "Box-Cox Power Exponential-orig.")
## Fitting method: "nlminb"
##
## Call:
## gamlssML(formula = y, family = DIST[i])
##
## Mu Coefficients:
## [1]  7.606
## Sigma Coefficients:
## [1]  2.44
## Nu Coefficients:
## [1]  1.272
## Tau Coefficients:
## [1]  -0.8186
##
##  Degrees of Freedom for the fit: 4 Residual Deg. of Freedom   436
## Global Deviance:     8728.48
##            AIC:     8736.48
##            SBC:     8752.83
```

```
#So according to the function "fitDist" the best model is "Box-Cox Power Exponential-orig."

par(mfrow= c(1,1))
Grocery.BCPEo<-histDist(data$Grocery, family=BCPEo, xlab="Annual Spending on Grocery Product", nbins=50, main="Annual Spending on Grocery Product_Box-Cox Power Exponential-orig. Distributed")
```




```
data.frame(row.names="Box-Cox Power Exponential-orig.", df=Grocery.BCPEo$df.fit, AIC=AIC(Grocery.BCPEo),
           SBC=Grocery.BCPEo$sbc, LogLik=logLik(Grocery.BCPEo))
```

```
##                                df      AIC
## Box-Cox Power Exponential-orig.  4 8736.485
##                                    SBC
## Box-Cox Power Exponential-orig. 8752.832
##                                   LogLik
## Box-Cox Power Exponential-orig. -4364.242
```

```
#So according to the all criterions and the "fitDist" function:
#the best model is the "Box-Cox Power Exponential-orig."


##About Annual Spending on Frozen Product
#is a numerical continuous variable, defined on the interval [0,+∞)
summary(data$Frozen)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.
##    25.0   742.2  1526.0  3071.9  3554.2
##    Max.
## 60869.0
```

```
kurtosis(data$Frozen)
```

```
## [1] 57.05618
```

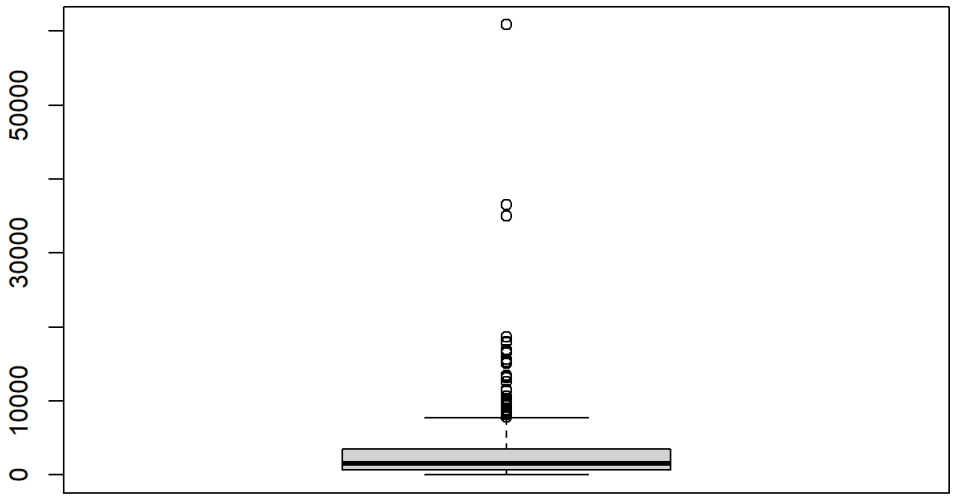
```
#Since the kurtosis is greater than 3, this indicates that the distribution has more values in the tails compared to a normal distribution.
#The "skinniness" of a leptokurtic distribution is a consequence of the outliers, which stretch the horizontal axis of the histogram graph,
#making the bulk of the data appear in a narrow ("skinny") vertical range.

skewness(data$Frozen)
```

```
## [1] 5.887826
```

```
#A positive skewness would indicate a distribution would be biased towards higher values, such that the mean of the distribution will
#exceed the median of the distribution. Right Skewed distributions greater than 0.8 often indicate the presence of a handful of exceptionally high outliers.

jarque.test(data$Frozen)
```

```
##
##  Jarque-Bera Normality Test
##
## data:  data$Frozen
## JB = 56114, p-value < 2.2e-16
## alternative hypothesis: greater
```

```
#For this test the Null Hypothesis is: the dataset has a skewness and kurtosis that matches a normal distribution.
#While Alternative Hypothesis is: the dataset has a skewness and kurtosis that does not match a normal distribution.

par(mfrow= c(1,1))
box<-boxplot(data$Frozen)
```




```
box$out
```

```
##  [1]  9408 10002  9510 10643  8872  8132
##  [7]  9735  8693 35009 18028  8853 16538
## [13]  8195  8425 16745 36534  7888 18711
## [19]  8321 11422 10155 16919 10303  8692
## [25]  8366 12569 60869  7849 11559  8170
## [31] 15601  9584  8164  9927  8620 13223
## [37]  9806 17866 15348 15082 13486 13135
```

```
data[which(data$Frozen%in% box$out),]
```

```
## # A tibble: 42 × 8
##    Channel  Region Fresh  Milk Grocery Frozen
##    <fct>    <fct>  <dbl> <dbl>   <dbl>  <dbl>
##  1 Ho.re.ca Other  31276  1917    4469   9408
##  2 Ho.re.ca Other  56159   555     902  10002
##  3 Ho.re.ca Other  24025  4332    4757   9510
##  4 Ho.re.ca Other  16705  2037    3202  10643
##  5 Ho.re.ca Other   4420  5139    2661   8872
##  6 Retail   Other  19899  5332    8713   8132
##  7 Ho.re.ca Other   7864   542    4042   9735
##  8 Ho.re.ca Other  12754  2762    2530   8693
##  9 Ho.re.ca Other  11314  3090    2062  35009
## 10 Ho.re.ca Other  56082  3504    8906  18028
## # … with 32 more rows, and 2 more variables:
## #   Detergents_Paper <dbl>, Delicassen <dbl>
```

```
#Most outliers come from Channel Ho.re.ca, this might be related to the fact that
#Ho.re.ca is characterized by high values of annual spending on Frozen Product than other channel.

table(data$Frozen)
```

```
##
##   25   33   36   38   42   47   52
##    1    1    1    1    1    1    1
##   61   65   74   75   91   96   98
##    1    1    1    1    1    1    1
##  118  127  129  130  131  133  134
##    1    1    1    1    1    2    1
##  137  142  145  155  175  179  188
##    1    1    1    1    1    1    1
##  191  201  206  214  220  228  230
##    1    1    1    1    1    1    1
##  233  239  245  247  264  266  269
##    1    1    1    1    1    1    1
##  275  282  283  287  294  317  321
```

```
##    1    1    1    1    1    1    1
##  327  333  340  346  349  364  388
##    1    1    1    1    1    2    1
##  397  398  402  405  414  416  417
##    1    1    2    1    1    1    1
##  425  430  432  436  437  440  443
##    2    1    1    1    1    1    1
##  453  469  480  485  492  502  514
##    1    1    1    1    1    1    1
##  520  529  531  532  541  547  559
##    1    1    1    1    1    1    1
##  560  561  575  576  584  596  597
##    1    1    1    1    1    1    1
##  617  633  638  650  651  659  660
##    1    1    1    1    1    1    1
##  661  662  666  669  673  688  720
##    1    1    1    1    1    1    1
##  737  744  767  774  779  784  787
##    1    2    1    1    2    1    1
##  799  800  805  806  824  825  830
##    1    1    1    1    2    1    2
##  833  839  848  853  862  864  868
##    1    1    2    1    1    1    1
##  869  874  876  878  880  890  895
##    1    1    1    1    1    1    1
##  902  907  909  910  913  915  926
##    1    1    1    1    1    1    1
##  930  934  937  950  955  959  977
##    1    1    2    1    1    1    1
##  978  982  987  993 1026 1031 1034
##    1    1    1    1    1    1    1
## 1038 1042 1057 1059 1066 1069 1089
##    1    1    1    1    1    1    1
## 1092 1093 1103 1112 1116 1120 1127
##    1    1    1    1    1    1    1
## 1128 1148 1149 1152 1158 1159 1170
##    1    1    1    1    1    1    1
## 1173 1178 1183 1200 1206 1218 1234
##    1    1    1    1    1    1    1
## 1256 1274 1285 1286 1291 1293 1329
##    1    1    2    1    1    1    1
## 1336 1341 1364 1365 1374 1383 1388
##    1    1    1    1    1    1    1
## 1389 1393 1398 1420 1439 1455 1456
##    1    1    1    1    1    1    1
## 1457 1465 1483 1492 1504 1517 1535
##    1    1    1    1    1    1    1
## 1541 1593 1601 1619 1646 1647 1668
##    1    1    1    2    1    1    1
## 1669 1677 1691 1718 1729 1730 1740
##    1    1    1    1    1    1    1
## 1741 1752 1759 1762 1765 1777 1796
##    1    1    1    1    1    1    1
## 1798 1801 1809 1840 1859 1920 1945
##    1    1    1    1    1    1    1
## 1960 1991 2005 2033 2046 2069 2077
##    1    1    1    1    1    1    1
## 2088 2096 2121 2181 2194 2198 2205
##    1    1    1    1    1    1    1
## 2208 2216 2221 2234 2276 2279 2286
##    1    1    1    1    1    1    1
## 2306 2312 2320 2349 2367 2369 2395
##    1    1    1    1    1    1    1
## 2399 2405 2417 2436 2471 2507 2515
##    1    1    1    1    1    1    1
## 2516 2532 2540 2555 2561 2583 2601
##    1    1    2    1    1    1    1
## 2616 2644 2665 2679 2693 2714 2715
##    1    1    1    1    1    1    1
## 2758 2809 2854 2896 2915 2921 2946
##    1    1    1    1    1    1    1
## 2995 3001 3012 3019 3045 3046 3095
##    1    1    1    1    1    1    1
## 3141 3151 3157 3220 3232 3242 3252
##    1    1    1    1    1    1    1
## 3254 3347 3378 3383 3443 3470 3498
##    1    1    1    1    1    1    1
## 3527 3549 3570 3576 3635 3703 3724
```

```
##    1    1    1    1    1    1    1
##  3752  3838  3860  3881  3896  3915  3941
##    1    1    1    1    1    1    1
##  3975  4006  4052  4154  4260  4324  4400
##    1    1    1    1    1    2    1
##  4407  4425  4447  4477  4479  4510  4575
##    1    1    1    1    1    1    1
##  4634  4686  4736  4787  4802  5004  5065
##    1    1    1    1    1    1    1
##  5127  5154  5243  5343  5373  5390  5500
##    1    1    1    1    1    1    1
##  5502  5612  5639  5641  5679  5845  5870
##    1    1    1    1    1    1    1
##  5970  6114  6130  6269  6312  6316  6340
##    1    1    1    1    1    1    1
##  6386  6404  6422  6746  6818  6838  6845
##    1    1    1    1    1    1    1
##  7332  7368  7496  7530  7683  7782  7849
##    1    1    1    1    1    1    1
##  7888  8132  8164  8170  8195  8321  8366
##    1    1    1    1    1    1    1
##  8425  8620  8692  8693  8853  8872  9408
##    1    1    1    1    1    1    1
##  9510  9584  9735  9806  9927 10002 10155
##    1    1    1    1    1    1    1
## 10303 10643 11422 11559 12569 13135 13223
##    1    1    1    1    1    1    1
## 13486 15082 15348 15601 16538 16745 16919
##    1    1    1    1    1    1    1
## 17866 18028 18711 35009 36534 60869
##    1    1    1    1    1    1
```

```
length(unique(data$Frozen))
```

```
## [1] 426
```

```
quantile(data$Frozen, 0.25)
```

```
##    25%
## 742.25
```

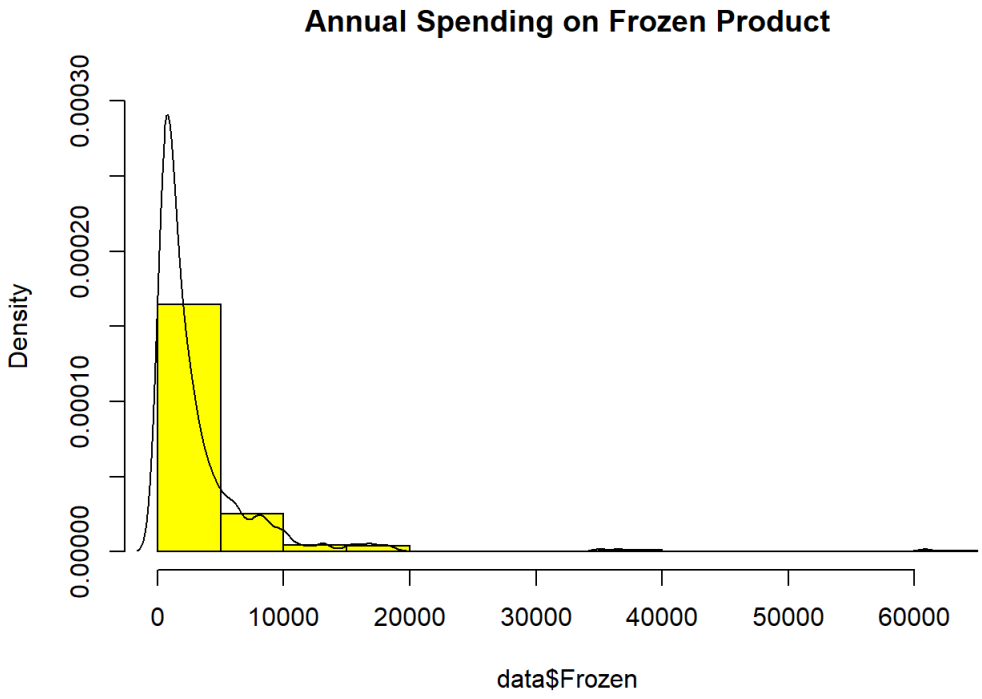
```
quantile(data$Frozen, 0.50)
```

```
##  50%
## 1526
```

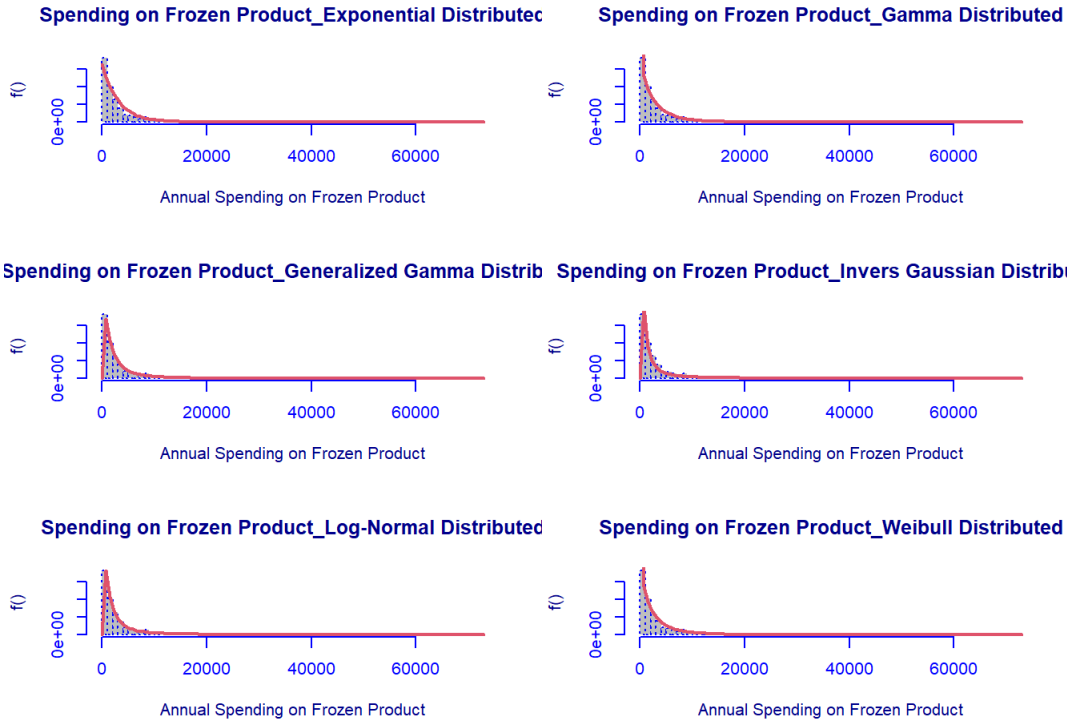
```
quantile(data$Frozen, 0.75)
```

```
##    75%
## 3554.25
```

```
par(mfrow= c(1,1))
hist(data$Frozen, main="Annual Spending on Frozen Product", ylim=c(0,3e-04), col="yellow", freq = FALSE)
lines(density(data$Frozen), col="black")
```




```
#Model Fitting of Frozen Product
par(mfrow=c(3,2))
Frozen.Exp<-histDist(data$Frozen, family=EXP, xlab="Annual Spending on Frozen Product", nbins=50, main="Spending on Frozen Product_Exponential Distributed")
Frozen.Ga<-histDist(data$Frozen, family=GA, xlab="Annual Spending on Frozen Product", nbins=50, main="Spending on Frozen Product_Gamma Distributed")
Frozen.Gg<-histDist(data$Frozen, family=GG, xlab="Annual Spending on Frozen Product", nbins=50, main="Spending on Frozen Product_Generalized Gamma Distributed")
Frozen.Ig<-histDist(data$Frozen, family=IG, xlab="Annual Spending on Frozen Product", nbins=50, main="Spending on Frozen Product_Invers Gaussian Distributed")
Frozen.Logn<-histDist(data$Frozen, family=LOGNO, xlab="Annual Spending on Frozen Product", nbins=50, main="Spending on Frozen Product_Log-Normal Distributed")
Frozen.Wei<-histDist(data$Frozen, family=WEI, xlab="Annual Spending on Frozen Product", nbins=50, main="Spending on Frozen Product_Weibull Distributed")
```




```
data.frame(row.names=c("Exponential","Gamma","Generalized Gamma","Inverse Gaussian","Log-normal","Weibull"),
           df=c(Frozen.Exp$df.fit,Frozen.Ga$df.fit,Frozen.Gg$df.fit,Frozen.Ig$df.fit,Frozen.Logn$df.fit,Frozen.Wei$df.fit),
           AIC=c(AIC(Frozen.Exp),AIC(Frozen.Ga),AIC(Frozen.Gg),AIC(Frozen.Ig),AIC(Frozen.Logn),AIC(Frozen.Wei)),
           SBC=c(Frozen.Exp$sbc,Frozen.Ga$sbc,Frozen.Gg$sbc,Frozen.Ig$sbc,Frozen.Logn$sbc,Frozen.Wei$sbc),
           LogLik=c(logLik(Frozen.Exp),logLik(Frozen.Ga),logLik(Frozen.Gg),logLik(Frozen.Ig),logLik(Frozen.Logn),logLik(Frozen.Wei)))
```

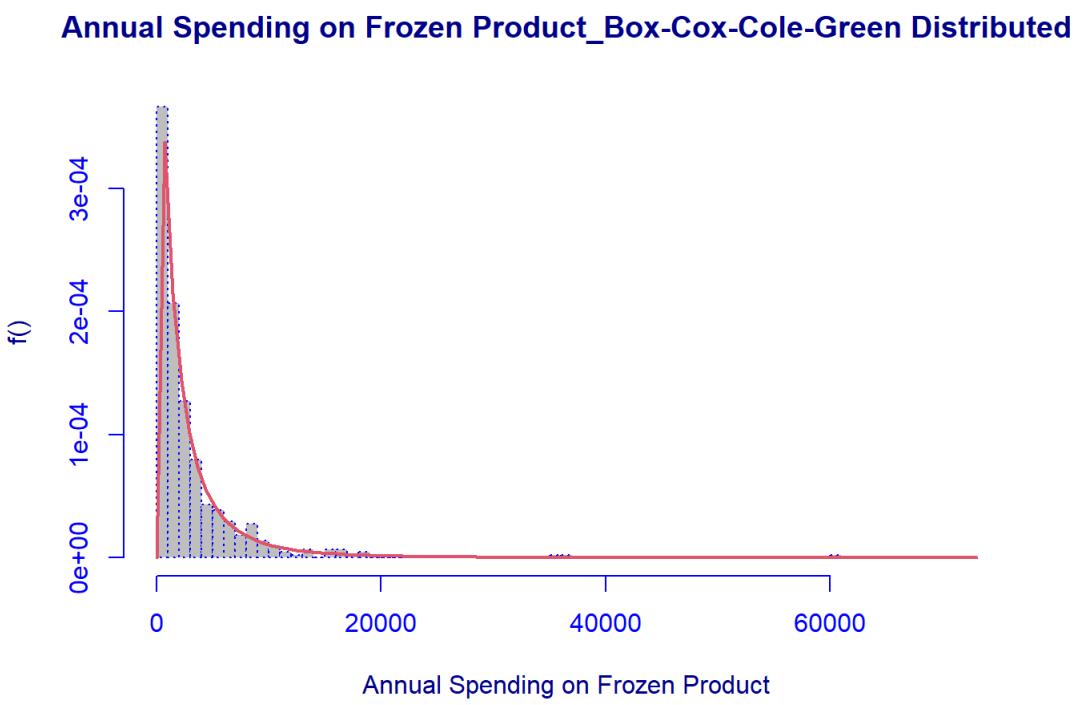
```
##                  df      AIC      SBC
## Exponential       1 7948.454 7952.541
## Gamma             2 7937.041 7945.214
## Generalized Gamma  3 7890.293 7902.553
## Inverse Gaussian   2 8003.945 8012.118
## Log-normal        2 7897.246 7905.419
## Weibull           2 7922.062 7930.235
##                   LogLik
## Exponential       -3973.227
## Gamma             -3966.520
## Generalized Gamma -3942.146
## Inverse Gaussian  -3999.972
## Log-normal        -3946.623
## Weibull           -3959.031
```

```
#Accordind to the 3 criterions the best model is "Generalized Gamma".

#But it's possible compare other model in order to find the model which best fit our data using the function:
fitDist(data$Frozen,type="realplus")
```

```
##
  |
  |                      |   0%
  |
  |==                    |   4%
  |
  |===                   |   9%
  |
  |=====                 |  13%
  |
  |======                |  17%
  |
  |========              |  22%
  |
  |=========             |  26%
  |
  |===========           |  30%
  |
  |============          |  35%
  |
  |==============        |  39%
  |
  |================      |  43%
  |
  |==================    |  48%
  |
  |====================  |  52%
  |
  |======================|  57%
  |
  |======================|  61%
  |
  |======================|  65%
  |
  |======================|  70%
  |
  |======================|  74%
  |
  |======================|  78%
  |
  |======================|  83%
  |
  |======================|  87%Error in solve.default(oout$hessian) :
##   Lapack routine dgesv: system is exactly singular: U[4,4] = 0
##
  |
  |======================|  91%
  |
  |======================|  96%
  |
  |======================| 100%
```

```
##
## Family:  c("BCCG", "Box-Cox-Cole-Green")
## Fitting method: "nlminb"
##
## Call:
## gamlssML(formula = y, family = DIST[i])
##
## Mu Coefficients:
## [1]  1594
## Sigma Coefficients:
## [1]  0.2325
## Nu Coefficients:
## [1]  0.08949
##
##  Degrees of Freedom for the fit: 3 Residual Deg. of Freedom   437
## Global Deviance:     7884.22
##             AIC:     7890.22
##             SBC:     7902.48
```

```
#So, according to "fitDist" function, the best model is "Box-Cox-Cole-Green"

par(mfrow= c(1,1))
Frozen.BCCG<-histDist(data$Frozen, family=BCCG, xlab="Annual Spending on Frozen Product", nbins=50, main="Annual Spending on Frozen Product_Box-Cox-Cole-Green Distributed")
```




```
data.frame(row.names="Box-Cox-Cole-Green", df=Frozen.BCCG$df.fit, AIC=AIC(Frozen.BCCG),
           SBC=Frozen.BCCG$sbc, LogLik=logLik(Frozen.BCCG))
```

```
##                       df      AIC      SBC
## Box-Cox-Cole-Green  3 7890.216 7902.477
##                        LogLik
## Box-Cox-Cole-Green -3942.108
```

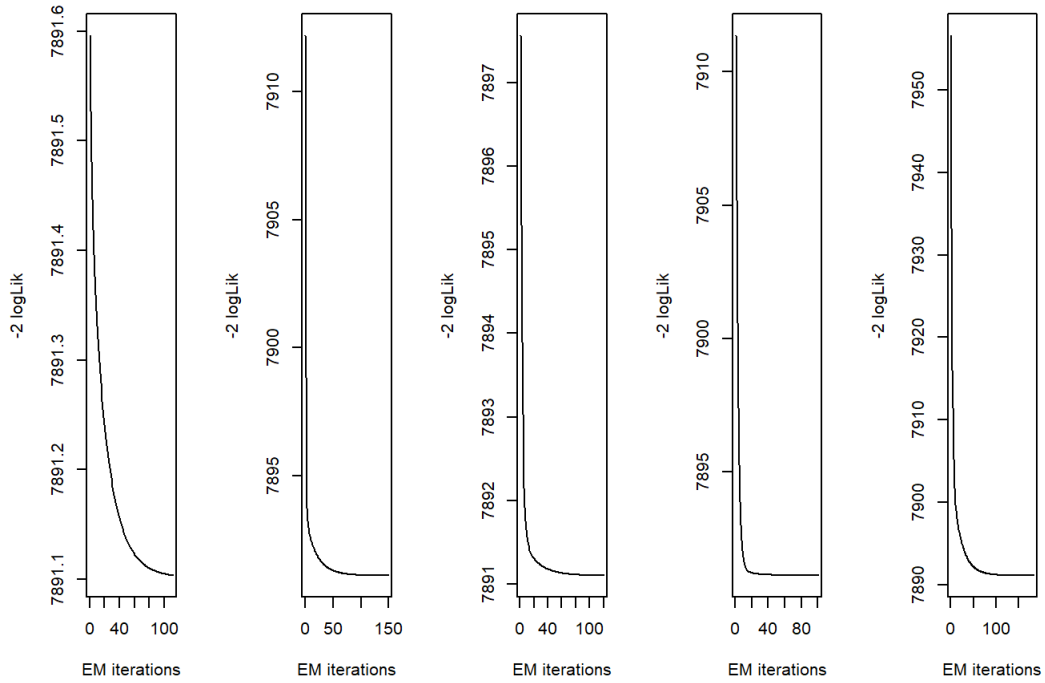
```
#According to the criterions and the "fitDist" function, the potentially best model is "Box-Cox-Cole-Green".

#MIXTURE MODEL for Frozen with k=2
#It's possible to combine two or more distribution and compute a mixture model that can fit better the
#distribution. In this case two gaussian distribution are used, so K=2.
set.seed(123)
par(mfrow= c(1,5))
F.mix.GA<-gamlssMXfits(n=5,data$Frozen~1,family=GA,K=2,data=NULL)
```

```
## model= 1
```

```
## model= 2
```

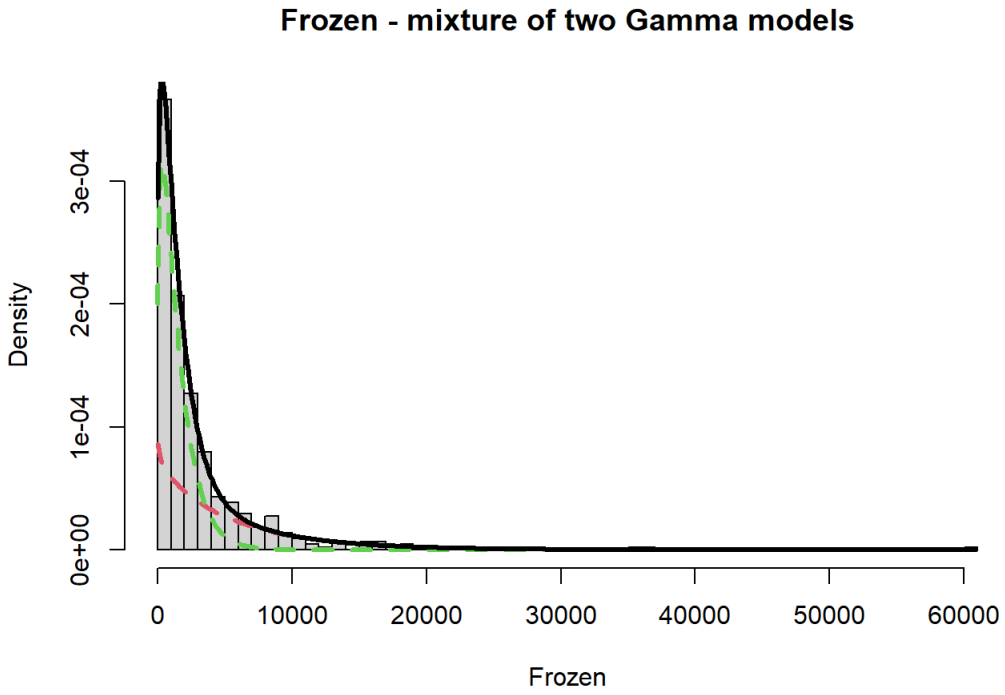
```
## model= 3
```

```
## model= 4
```




```
## model= 5
```

```
#estimate mu and sigma in the group 1 and 2
mu1<-exp(F.mix.GA[["models"]][[1]][["mu.coefficients"]])
mu2<-exp(F.mix.GA[["models"]][[2]][["mu.coefficients"]])
sigma1<-exp(F.mix.GA[["models"]][[1]][["sigma.coefficients"]])
sigma2<-exp(F.mix.GA[["models"]][[2]][["sigma.coefficients"]])

par(mfrow= c(1,1))
hist(data$Frozen,breaks=50,freq=FALSE,xlab="Frozen",main="Frozen - mixture of two Gamma models")
lines(seq(min(data$Frozen),max(data$Frozen),length=length(data$Frozen))
      ,F.mix.GA[["prob"]][1]*dGA(seq(min(data$Frozen), max(data$Frozen),length=length(data$Frozen)), mu=mu1, sigma=sigma1),lty=2,lwd=3,col=2)
lines(seq(min(data$Frozen),max(data$Frozen),length=length(data$Frozen))
      ,F.mix.GA[["prob"]][2]*dGA(seq(min(data$Frozen),max(data$Frozen),length=length(data$Frozen)), mu=mu2, sigma=sigma2), lty=2,lwd=3,col=3)
lines(seq(min(data$Frozen),max(data$Frozen),length=length(data$Frozen))
      ,F.mix.GA[["prob"]][1]*dGA(seq(min(data$Frozen), max(data$Frozen),length=length(data$Frozen)), mu=mu1, sigma=sigma1)+
        F.mix.GA[["prob"]][2]*dGA(seq(min(data$Frozen),max(data$Frozen),length=length(data$Frozen)), mu=mu2, sigma= sigma2),
      lty=1,lwd=3,col=1)
```



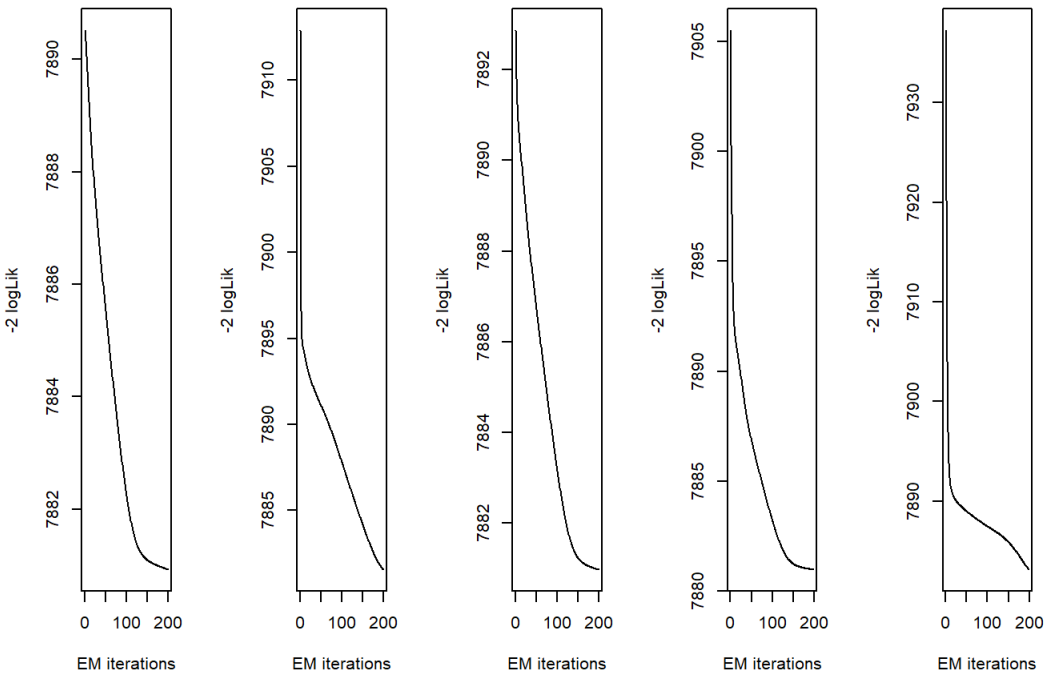

```
#MIXTURE MODEL for Frozen with k=3
set.seed(123)
par(mfrow= c(1,5))
F3.mix.GA<-gamlssMXfits(n=5,data$Frozen~1,family=GA,K=3,data=NULL)
```

```
## model= 1
```

```
## model= 2
```

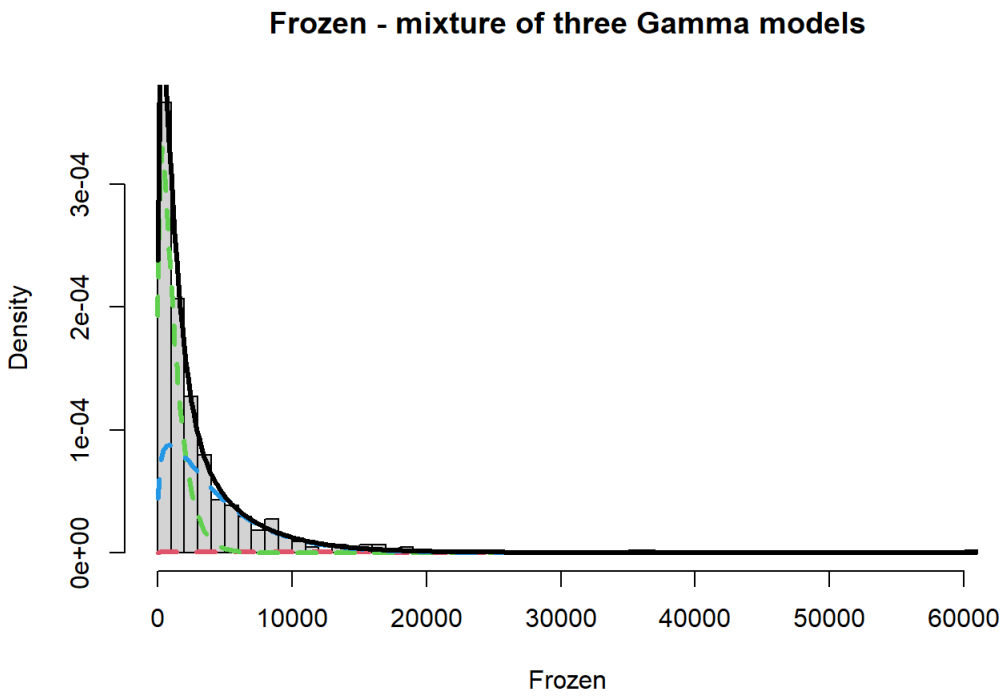
```
## model= 3
```

```
## model= 4
```




```
## model= 5
```

```
#estimate mu and sigma in the group 1 and 2
mu1<-exp(F3.mix.GA[["models"]][[1]][["mu.coefficients"]])
mu2<-exp(F3.mix.GA[["models"]][[2]][["mu.coefficients"]])
mu3<-exp(F3.mix.GA[["models"]][[3]][["mu.coefficients"]])
sigma1<-exp(F3.mix.GA[["models"]][[1]][["sigma.coefficients"]])
sigma2<-exp(F3.mix.GA[["models"]][[2]][["sigma.coefficients"]])
sigma3<-exp(F3.mix.GA[["models"]][[3]][["sigma.coefficients"]])

par(mfrow= c(1,1))
hist(data$Frozen,breaks=50,freq=FALSE,xlab="Frozen",main="Frozen - mixture of three Gamma models")
lines(seq(min(data$Frozen),max(data$Frozen),length=length(data$Frozen))
    ,F3.mix.GA[["prob"]][1]*dGA(seq(min(data$Frozen), max(data$Frozen),length=length(data$Frozen)), mu=mu1, sigma=sigma1),lty=2,lwd=3,col=2)
lines(seq(min(data$Frozen),max(data$Frozen),length=length(data$Frozen))
    ,F3.mix.GA[["prob"]][2]*dGA(seq(min(data$Frozen),max(data$Frozen),length=length(data$Frozen)), mu=mu2, sigma=sigma2), lty=2,lwd=3,col=3)
lines(seq(min(data$Frozen),max(data$Frozen),length=length(data$Frozen))
    ,F3.mix.GA[["prob"]][2]*dGA(seq(min(data$Frozen),max(data$Frozen),length=length(data$Frozen)), mu=mu3, sigma=sigma3), lty=2,lwd=3,col=4)
lines(seq(min(data$Frozen),max(data$Frozen),length=length(data$Frozen))
    ,F3.mix.GA[["prob"]][1]*dGA(seq(min(data$Frozen), max(data$Frozen),length=length(data$Frozen)), mu=mu1, sigma=sigma1)+
     F3.mix.GA[["prob"]][2]*dGA(seq(min(data$Frozen),max(data$Frozen),length=length(data$Frozen)), mu=mu2, sigma=sigma2)+
     F3.mix.GA[["prob"]][2]*dGA(seq(min(data$Frozen),max(data$Frozen),length=length(data$Frozen)), mu=mu3, sigma=sigma3),
    lty=1,lwd=3,col=1)
```




```
data.frame(row.names=c("Gamma mixtures with K=2","Gamma mixtures with K=3"), AIC=c(AIC(F.mix.GA),AIC(F3.mix.GA)),
          SBC=c(F.mix.GA$sbc,F3.mix.GA$sbc), LogLik=c(logLik(F.mix.GA),logLik(F3.mix.GA)))
```

```
##                             AIC      SBC
## Gamma mixtures with K=2 7901.103 7921.537
## Gamma mixtures with K=3 7896.921 7929.616
##                            LogLik
## Gamma mixtures with K=2 -3945.552
## Gamma mixtures with K=3 -3940.461
```

```
LR.test(F.mix.GA,F3.mix.GA)
```

```
##  Likelihood Ratio Test for nested GAMLSS models.
##  (No check whether the models are nested is performed).
##
##        Null model: deviance= 7891.103 with  5 deg. of freedom
##  Altenative model: deviance= 7880.921 with  8 deg. of freedom
##
##  LRT = 10.18203 with 3 deg. of freedom and p-value= 0.0170805
```

```
#Using the LR.test in order to check if the mixture model of three gamma distributions brings enough
#information to justify the higher number of parameters. Due to the p-value<0.05 the model with K=3 is
#effectively the best one for the LogLik criterion.

LR.test(Frozen.BCCG,F3.mix.GA)
```

```
##  Likelihood Ratio Test for nested GAMLSS models.
##  (No check whether the models are nested is performed).
##
##        Null model: deviance= 7884.216 with  3 deg. of freedom
##  Altenative model: deviance= 7880.921 with  8 deg. of freedom
##
##  LRT = 3.295112 with 5 deg. of freedom and p-value= 0.6545901
```

```
#In this case the single "Box-Cox-Cole-Green" is the best model according the LR test


##About Annual Spending on Frozen Product
#is a numerical continuous variable, defined on the interval [0,+∞)
summary(data$Detergents_Paper)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.
##     3.0   256.8   816.5  2881.5  3922.0
##    Max.
## 40827.0
```

```
kurtosis(data$Detergents_Paper)
```

```
## [1] 21.78053
```

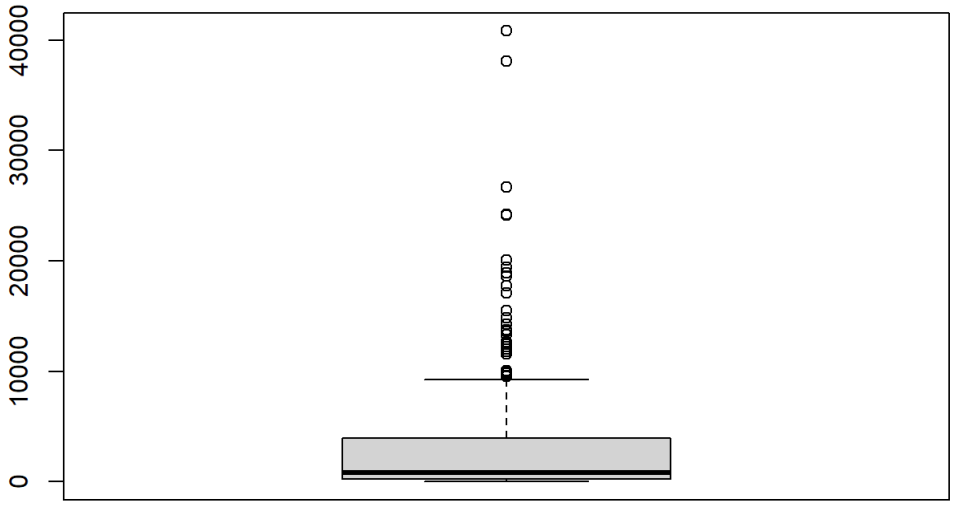
```
skewness(data$Detergents_Paper)
```

```
## [1] 3.619458
```

```
jarque.test(data$Detergents_Paper)
```

```
##
##  Jarque-Bera Normality Test
##
## data:  data$Detergents_Paper
## JB = 7427, p-value < 2.2e-16
## alternative hypothesis: greater
```

```
par(mfrow= c(1,1))
box<-boxplot(data$Detergents_Paper)
```




```
box$out
```

```
##  [1]  9529 24171 13583 17740 26701 24231
##  [7] 12034 40827 20070 18906 12591 11577
## [13] 13726  9836  9959 14235 12420 19410
## [19] 13308 18594 10069 11783 17120 12408
## [25]  9606 38102 15469 12218 12638 14841
```

```
data[which(data$Detergents_Paper%in% box$out),]
```

```
## # A tibble: 30 × 8
##    Channel Region Fresh  Milk Grocery Frozen
##    <fct>   <fct>  <dbl> <dbl>   <dbl>  <dbl>
##  1 Retail  Other    630 11095   23998    787
##  2 Retail  Other  44466 54259   55571   7782
##  3 Retail  Other   4967 21412   28921   1798
##  4 Retail  Other   4098 29892   26866   2616
##  5 Retail  Other  35942 38369   59598   3254
##  6 Retail  Other     85 20959   45828     36
##  7 Retail  Other  12205 12697   28540    869
##  8 Retail  Other  16117 46197   92780   1026
##  9 Retail  Other  22925 73498   32114    987
## 10 Retail  Other   9198 27472   32034   3232
## # … with 20 more rows, and 2 more variables:
## #   Detergents_Paper <dbl>, Delicassen <dbl>
```

```
#In this case all the outliers come from the Retail channel,
#this might be releated that Retail is characterized by
#high values of annual spending score on Detergents and Paper Product than the other Channel.

table(data$Detergents_Paper)
```

```
##
##    3    5    7    9   10   15   20
##    2    1    1    1    1    1    2
##   25   28   32   41   43   44   47
##    1    1    1    1    1    1    1
##   49   51   54   56   58   64   68
##    1    1    1    2    1    1    1
##   69   70   71   72   73   74   79
##    2    2    1    1    1    1    1
##   83   84   86   88   90   92   93
##    1    1    1    1    1    1    2
##   95   96  100  101  108  111  113
##    1    2    1    1    1    1    1
##  116  118  120  122  130  139  140
##    1    2    1    1    1    1    1
```

```
##  146  147  149  153  159  165  167
##    1    1    1    2    1    1    1
##  168  169  170  173  174  178  179
##    1    1    1    1    1    1    1
##  182  183  184  187  192  197  199
##    2    1    1    1    1    1    1
##  200  204  205  210  212  215  217
##    1    1    1    2    2    1    1
##  222  223  227  228  231  232  234
##    1    1    2    1    1    2    1
##  235  239  240  241  242  244  246
##    1    1    1    1    1    1    1
##  249  252  255  256  257  263  264
##    1    1    1    2    1    1    1
##  266  273  274  275  276  282  283
##    1    1    1    1    1    1    1
##  284  288  290  299  301  302  310
##    2    1    1    1    1    1    1
##  311  314  319  325  330  332  333
##    2    1    1    1    1    1    1
##  334  343  349  351  352  353  355
##    1    1    1    1    1    1    1
##  356  361  363  370  371  375  385
##    1    1    1    1    1    1    1
##  386  387  392  395  397  399  402
##    1    1    1    1    2    1    1
##  409  410  411  412  415  429  430
##    1    1    1    1    1    1    1
##  436  439  442  445  454  468  469
##    1    1    1    1    1    1    1
##  476  477  483  492  500  507  514
##    1    1    2    1    1    1    1
##  516  523  529  536  549  550  562
##    1    1    1    1    1    1    1
##  573  585  586  592  593  600  603
##    1    1    1    1    1    1    1
##  609  610  621  627  632  637  656
##    1    1    1    1    1    1    1
##  694  706  710  716  721  730  737
##    1    1    1    1    1    1    1
##  751  759  761  763  764  778  780
##    1    1    1    1    1    1    1
##  788  811  813  820  821  825  828
##    2    2    1    1    1    1    1
##  830  836  841  850  857  862  912
##    1    1    1    1    1    1    1
##  914  918  948  949  954  955  960
##    1    2    1    1    1    2    1
##  964  965  967 1041 1062 1082 1092
##    1    1    1    1    1    1    1
## 1107 1135 1145 1184 1202 1216 1226
##    1    1    1    1    1    1    1
## 1247 1272 1321 1328 1333 1335 1377
##    1    1    1    1    1    1    1
## 1382 1470 1480 1491 1532 1538 1547
##    1    1    1    1    1    1    1
## 1566 1579 1580 1679 1680 1682 1711
##    1    1    1    1    1    1    1
## 1716 1777 1795 1803 1860 1901 1916
##    1    1    1    1    1    1    1
## 1976 2024 2123 2134 2208 2259 2328
##    1    1    1    1    1    1    1
## 2371 2381 2386 2447 2505 2518 2568
##    1    1    1    1    1    1    1
## 2575 2587 2662 2674 2730 2767 2840
##    1    1    1    1    1    1    1
## 2894 2970 3084 3140 3143 3213 3293
##    1    1    1    1    1    1    1
## 3321 3378 3381 3415 3459 3468 3485
##    1    1    1    1    1    1    1
## 3516 3537 3540 3593 3620 3674 3712
##    1    1    1    1    1    1    1
## 3837 3843 3874 3881 3891 3909 3961
##    1    1    1    1    1    1    1
## 4003 4004 4027 4074 4095 4111 4167
##    1    1    1    1    1    1    1
## 4173 4196 4239 4314 4337 4424 4482
##    1    1    1    1    1    1    1
```

```
## 4508 4515 4538 4573 4595 4618 4621
##    1    1    1    1    1    1    1
## 4666 4762 4797 4882 4948 4973 5038
##    1    1    1    1    1    1    1
## 5058 5079 5121 5141 5162 5316 5611
##    1    1    1    1    1    1    1
## 5618 5828 5952 5957 5977 5980 6236
##    1    1    1    1    1    1    1
## 6374 6457 6506 6600 6694 6707 6728
##    1    1    1    1    1    1    1
## 6740 6747 6766 6792 6818 6830 6839
##    1    1    1    1    1    1    1
## 6846 6899 6907 6956 7015 7108 7271
##    1    1    1    1    1    1    1
## 7353 7425 7558 7572 7677 7818 7883
##    1    1    1    1    1    1    1
## 8035 8077 8604 8682 8752 8773 8933
##    1    1    1    1    1    1    1
## 8969 9265 9529 9606 9836 9959 10069
##    1    1    1    1    1    1    1
## 11577 11783 12034 12218 12408 12420 12591
##    1    1    1    1    1    1    1
## 12638 13308 13583 13726 14235 14841 15469
##    1    1    1    1    1    1    1
## 17120 17740 18594 18906 19410 20070 24171
##    1    1    1    1    1    1    1
## 24231 26701 38102 40827
##    1    1    1    1
```

```
length(unique(data$Detergents_Paper))
```

```
## [1] 417
```

```
quantile(data$Detergents_Paper, 0.25)
```

```
##    25%
## 256.75
```

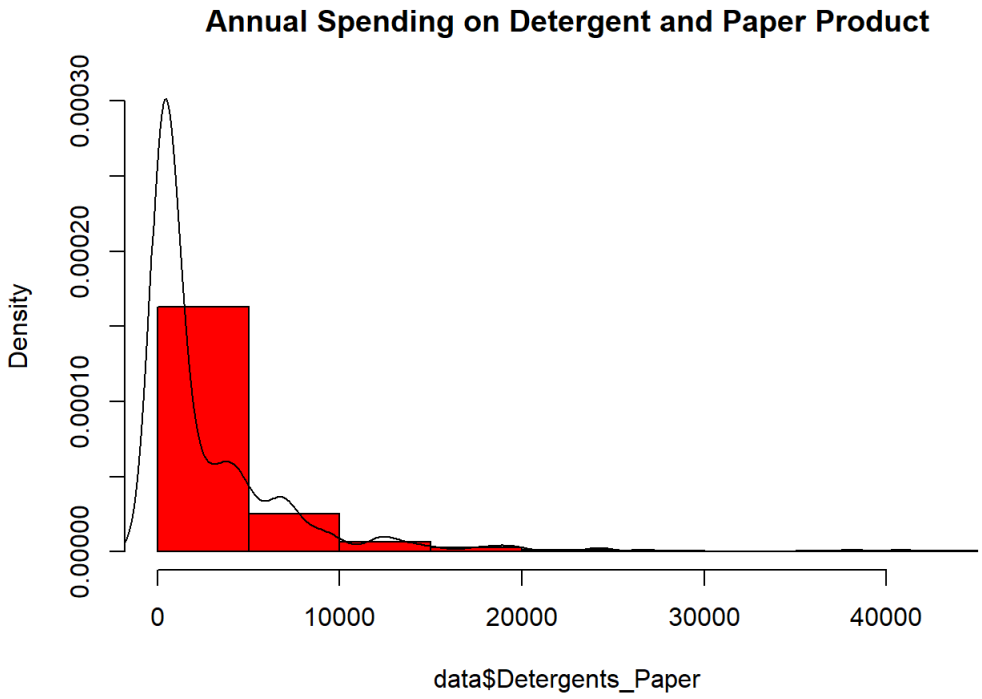
```
quantile(data$Detergents_Paper, 0.50)
```

```
##   50%
## 816.5
```

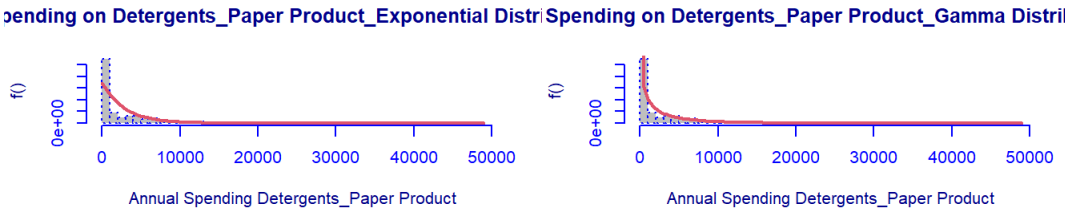
```
quantile(data$Detergents_Paper, 0.75)
```

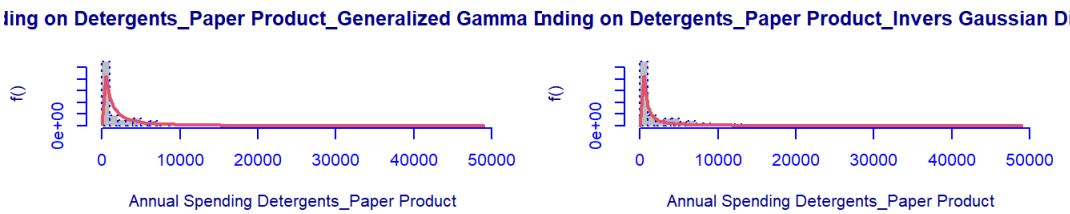
```
##  75%
## 3922
```

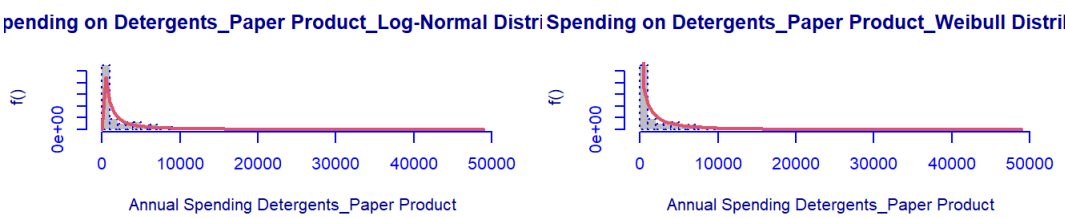
```
par(mfrow= c(1,1))
hist(data$Detergents_Paper, main="Annual Spending on Detergent and Paper Product", ylim=c(0,3e-04), col="red", freq = FALSE)
lines(density(data$Detergents_Paper), col="black")
```




```
#Model Fitting of Annual Spending on Detergent and Paper
par(mfrow=c(3,2))
DP.Exp<-histDist(data$Detergents_Paper, family=EXP, xlab="Annual Spending Detergents_Paper Product", nbins=50, main="Spending on Detergents_Paper Product_Exponential Distributed")
DP.Ga<-histDist(data$Detergents_Paper, family=GA, xlab="Annual Spending Detergents_Paper Product", nbins=50, main="Spending on Detergents_Paper Product_Gamma Distributed")
DP.Gg<-histDist(data$Detergents_Paper, family=GG, xlab="Annual Spending Detergents_Paper Product", nbins=50, main="Spending on Detergents_Paper Product_Generalized Gamma Distributed")
DP.Ig<-histDist(data$Detergents_Paper, family=IG, xlab="Annual Spending Detergents_Paper Product", nbins=50, main="Spending on Detergents_Paper Product_Invers Gaussian Distributed")
DP.Logn<-histDist(data$Detergents_Paper, family=LOGNO, xlab="Annual Spending Detergents_Paper Product", nbins=50, main="Spending on Detergents_Paper Product_Log-Normal Distributed")
DP.Wei<-histDist(data$Detergents_Paper, family=WEI, xlab="Annual Spending Detergents_Paper Product", nbins=50, main="Spending on Detergents_Paper Product_Weibull Distributed")
```








```
data.frame(row.names=c("Exponential","Gamma","Generalized Gamma","Inverse Gaussian","Log-normal","Weibull"),
           df=c(DP.Exp$df.fit,DP.Ga$df.fit,DP.Gg$df.fit,DP.Ig$df.fit,DP.Logn$df.fit,DP.Wei$df.fit),
           AIC=c(AIC(DP.Exp),AIC(DP.Ga),AIC(DP.Gg),AIC(DP.Ig),AIC(DP.Logn),AIC(DP.Wei)),
           SBC=c(DP.Exp$sbc,DP.Ga$sbc,DP.Gg$sbc,DP.Ig$sbc,DP.Logn$sbc,DP.Wei$sbc),
           LogLik=c(logLik(DP.Exp),logLik(DP.Ga),logLik(DP.Gg),logLik(DP.Ig),logLik(DP.Logn),logLik(DP.Wei)))
```

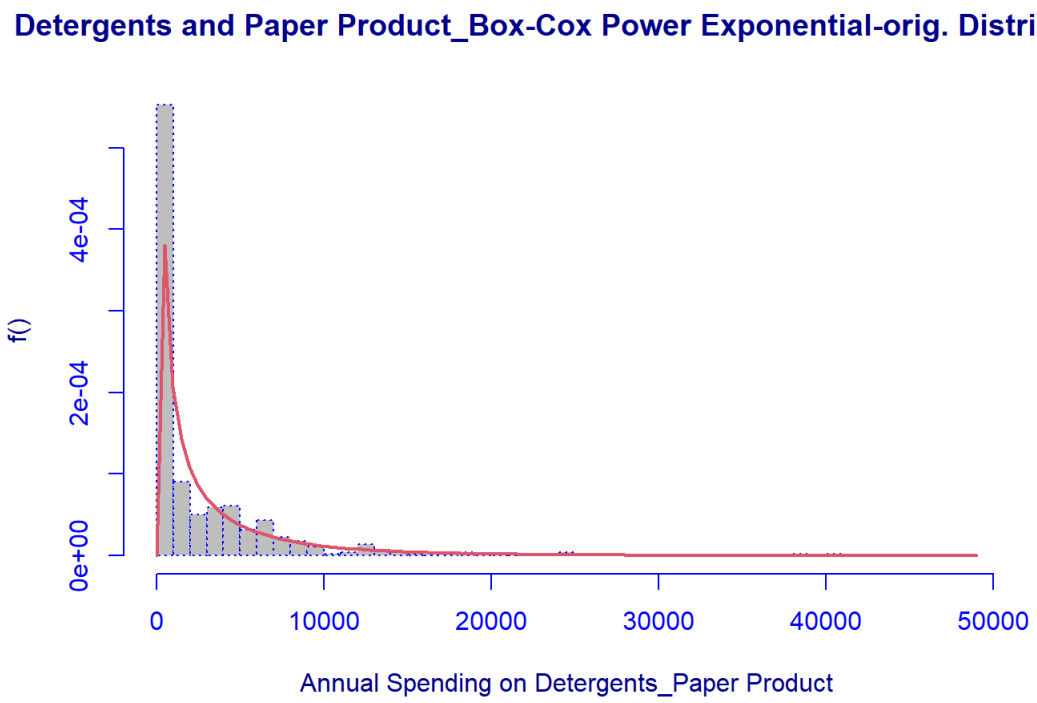
```
##                    df      AIC      SBC
## Exponential         1 7892.136 7896.223
## Gamma               2 7742.282 7750.456
## Generalized Gamma   3 7696.460 7708.721
## Inverse Gaussian    2 7942.793 7950.967
## Log-normal          2 7701.087 7709.261
## Weibull             2 7715.556 7723.729
##                       LogLik
## Exponential       -3945.068
## Gamma             -3869.141
## Generalized Gamma -3845.230
## Inverse Gaussian  -3969.396
## Log-normal        -3848.544
## Weibull           -3855.778
```

```
#According to the criterions the best model is the "Generalized Gamma".

#But it's possible compare other model in order to find the model which best fit our data using the function:
fitDist(data$Detergents_Paper, type="realplus")
```

```
##
  |
  |                            |   0%
  |
  |==                          |   4%
  |
  |===                         |   9%
  |
  |=====                       |  13%
  |
  |======                      |  17%
  |
  |========                    |  22%
  |
  |=========                   |  26%
  |
  |===========                 |  30%
  |
  |============                |  35%
  |
  |==============              |  39%
  |
  |================            |  43%
  |
  |==================          |  48%
  |
  |====================        |  52%
  |
  |======================      |  57%
  |
  |=======================     |  61%
  |
  |=========================   |  65%
  |
  |=========================   |  70%
  |
  |============================|  74%
  |
  |============================|  78%
  |
  |============================|  83%
  |
  |============================|  87%
  |
  |============================|  91%
  |
  |============================|  96%
  |
  |============================| 100%
```

```
##
## Family:  c("BCPEo", "Box-Cox Power Exponential-orig.")
## Fitting method: "nlminb"
##
## Call:
## gamlssML(formula = y, family = DIST[i])
##
## Mu Coefficients:
## [1]  6.968
## Sigma Coefficients:
## [1]  0.5152
## Nu Coefficients:
## [1]  0.08988
## Tau Coefficients:
## [1]  1.103
##
##  Degrees of Freedom for the fit: 4 Residual Deg. of Freedom   436
## Global Deviance:     7676.47
##             AIC:     7684.47
##             SBC:     7700.82
```

```
par(mfrow= c(1,1))
DP.BCPEo<-histDist(data$Detergents_Paper, family=BCPEo, xlab="Annual Spending on Detergents_Paper Product", nbins=50, main="Detergents and Paper Product_Box-Cox Power Exponential-orig. Distributed")
```




```
data.frame(row.names="Box-Cox Power Exponential-orig.", df=DP.BCPEo$df.fit, AIC=AIC(DP.BCPEo),
           SBC=DP.BCPEo$sbc, LogLik=logLik(DP.BCPEo))
```

```
##                                df      AIC
## Box-Cox Power Exponential-orig.  4 7684.469
##                                    SBC
## Box-Cox Power Exponential-orig. 7700.817
##                                  LogLik
## Box-Cox Power Exponential-orig. -3838.235
```

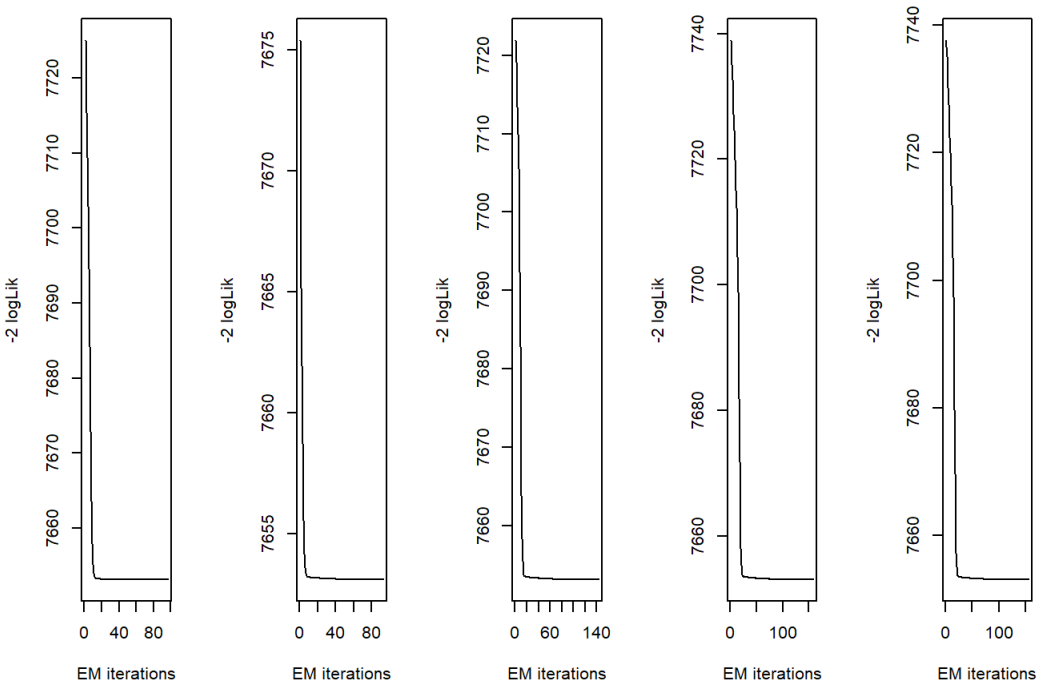
```
#So according to the all criterion and to the "fitDist" function, the potentially best model is the "Box-Cox Power Exponential-orig."


#MIXTURE MODEL for Detergents_Paper with k=2
#It's possible to combine two or more distribution and compute a mixture model that can fit better the
#distribution. In this case two gaussian distribution are used, so K=2.
set.seed(123)
par(mfrow= c(1,5))
DP.mix.GA<-gamlssMXfits(n=5,data$Detergents_Paper~1,family=GA,K=2,data=NULL)
```

```
## model= 1
```

```
## model= 2
```

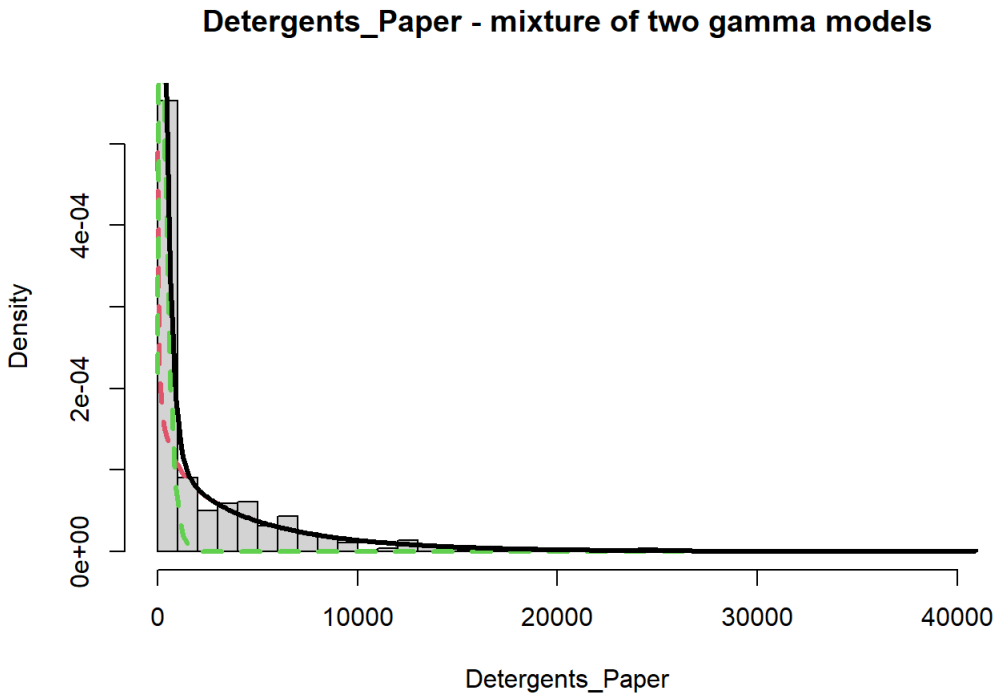
```
## model= 3
```

```
## model= 4
```




```
## model= 5
```

```
#estimate mu and sigma in the group 1 and 2
mu1<-exp(DP.mix.GA[["models"]][[1]][["mu.coefficients"]])
sigma1<-exp(DP.mix.GA[["models"]][[1]][["sigma.coefficients"]])
mu2<-exp(DP.mix.GA[["models"]][[2]][["mu.coefficients"]])
sigma2<-exp(DP.mix.GA[["models"]][[2]][["sigma.coefficients"]])

par(mfrow= c(1,1))
hist(data$Detergents_Paper,breaks=50,freq=FALSE,xlab="Detergents_Paper",main="Detergents_Paper - mixture of two gamma models")
lines(seq(min(data$Detergents_Paper),max(data$Detergents_Paper),length=length(data$Detergents_Paper))
     ,DP.mix.GA[["prob"]][1]*dGA(seq(min(data$Detergents_Paper), max(data$Detergents_Paper),length=length(data$Detergents_Paper)), mu=mu1, sigm
a=sigma1),lty=2,lwd=3,col=2)
lines(seq(min(data$Detergents_Paper),max(data$Detergents_Paper),length=length(data$Detergents_Paper))
     ,DP.mix.GA[["prob"]][2]*dGA(seq(min(data$Detergents_Paper),max(data$Detergents_Paper),length=length(data$Detergents_Paper)), mu=mu2, sigm
a=sigma2), lty=2,lwd=3,col=3)
lines(seq(min(data$Detergents_Paper),max(data$Detergents_Paper),length=length(data$Detergents_Paper))
     ,DP.mix.GA[["prob"]][1]*dGA(seq(min(data$Detergents_Paper), max(data$Detergents_Paper),length=length(data$Detergents_Paper)), mu=mu1, sigm
a=sigma1)+
      DP.mix.GA[["prob"]][2]*dGA(seq(min(data$Detergents_Paper),max(data$Detergents_Paper),length=length(data$Detergents_Paper)), mu=mu2, sigm
a=sigma2),
     lty=1,lwd=3,col=1)
```



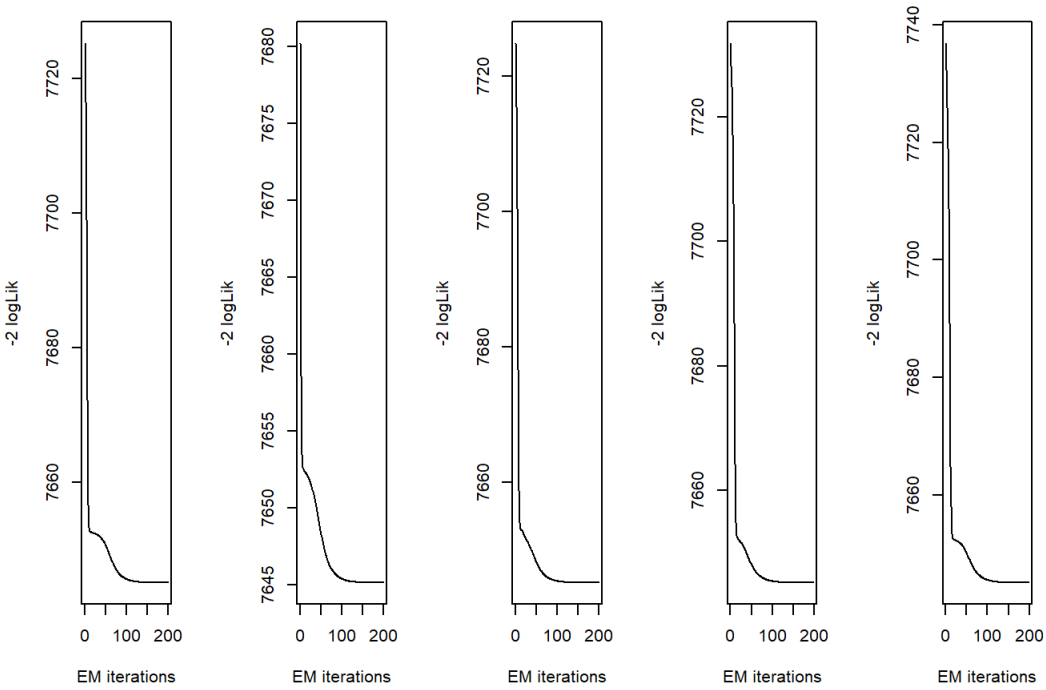

```
#MIXTURE MODEL for Detergents_Paper with k=3
set.seed(123)
par(mfrow= c(1,5))
DP3.mix.GA<-gamlssMXfits(n=5,data$Detergents_Paper~1,family=GA,K=3,data=NULL)
```

```
## model= 1
```

```
## model= 2
```

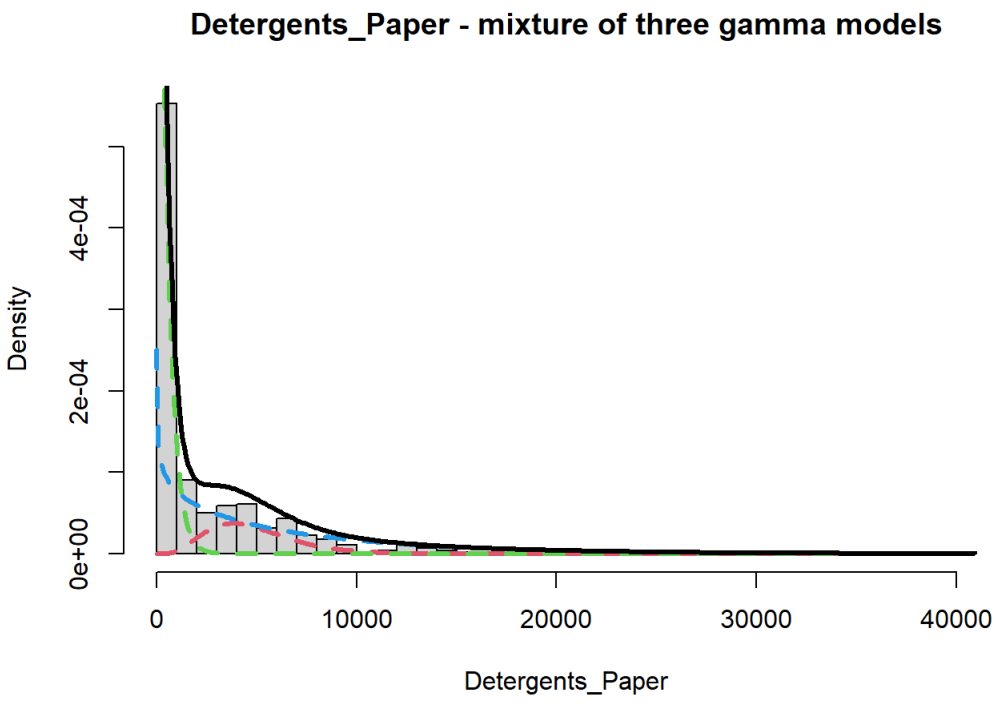
```
## model= 3
```

```
## model= 4
```




```
## model= 5
```

```
#estimate mu and sigma in the group 1 and 2
mu1<-exp(DP3.mix.GA[["models"]][[1]][["mu.coefficients"]])
sigma1<-exp(DP3.mix.GA[["models"]][[1]][["sigma.coefficients"]])
mu2<-exp(DP3.mix.GA[["models"]][[2]][["mu.coefficients"]])
sigma2<-exp(DP3.mix.GA[["models"]][[2]][["sigma.coefficients"]])
mu3<-exp(DP3.mix.GA[["models"]][[3]][["mu.coefficients"]])
sigma3<-exp(DP3.mix.GA[["models"]][[3]][["sigma.coefficients"]])

par(mfrow= c(1,1))
hist(data$Detergents_Paper,breaks=50,freq=FALSE,xlab="Detergents_Paper",main="Detergents_Paper - mixture of three gamma models")
lines(seq(min(data$Detergents_Paper),max(data$Detergents_Paper),length=length(data$Detergents_Paper))
    ,DP3.mix.GA[["prob"]][1]*dGA(seq(min(data$Detergents_Paper), max(data$Detergents_Paper),length=length(data$Detergents_Paper)), mu=mu1, sig
ma=sigma1),lty=2,lwd=3,col=2)
lines(seq(min(data$Detergents_Paper),max(data$Detergents_Paper),length=length(data$Detergents_Paper))
    ,DP3.mix.GA[["prob"]][2]*dGA(seq(min(data$Detergents_Paper),max(data$Detergents_Paper),length=length(data$Detergents_Paper)), mu=mu2, sig
ma=sigma2), lty=2,lwd=3,col=3)
lines(seq(min(data$Detergents_Paper),max(data$Detergents_Paper),length=length(data$Detergents_Paper))
    ,DP3.mix.GA[["prob"]][2]*dGA(seq(min(data$Detergents_Paper),max(data$Detergents_Paper),length=length(data$Detergents_Paper)), mu=mu3, sig
ma=sigma3), lty=2,lwd=3,col=4)
lines(seq(min(data$Detergents_Paper),max(data$Detergents_Paper),length=length(data$Detergents_Paper))
    ,DP3.mix.GA[["prob"]][1]*dGA(seq(min(data$Detergents_Paper), max(data$Detergents_Paper),length=length(data$Detergents_Paper)), mu=mu1, sig
ma=sigma1)+
     DP3.mix.GA[["prob"]][2]*dGA(seq(min(data$Detergents_Paper),max(data$Detergents_Paper),length=length(data$Detergents_Paper)), mu=mu2, sig
ma=sigma2)+
     DP3.mix.GA[["prob"]][2]*dGA(seq(min(data$Detergents_Paper),max(data$Detergents_Paper),length=length(data$Detergents_Paper)), mu=mu3, sig
ma=sigma3),
    lty=1,lwd=3,col=1)
```




```
data.frame(row.names=c("Gamma mixtures with K=2","Gamma mixtures with K=3"), AIC=c(AIC(DP.mix.GA),AIC(DP3.mix.GA)),
          SBC=c(DP.mix.GA$sbc,DP3.mix.GA$sbc), LogLik=c(logLik(DP.mix.GA),logLik(DP3.mix.GA)))
```

```
##                             AIC      SBC
## Gamma mixtures with K=2 7663.088 7683.522
## Gamma mixtures with K=3 7661.122 7693.816
##                          LogLik
## Gamma mixtures with K=2 -3826.544
## Gamma mixtures with K=3 -3822.561
```

```
LR.test(DP.mix.GA,DP3.mix.GA)
```

```
##  Likelihood Ratio Test for nested GAMLSS models.
##  (No check whether the models are nested is performed).
##
##        Null model: deviance= 7653.088 with  5 deg. of freedom
##  Altenative model: deviance= 7645.122 with  8 deg. of freedom
##
##  LRT = 7.966695 with 3 deg. of freedom and p-value= 0.04670505
```

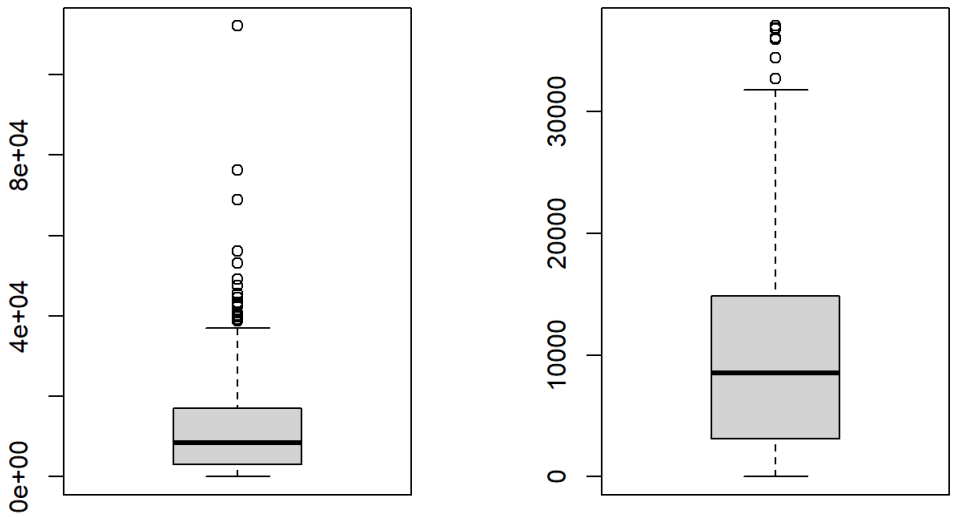
```
#Using the LR.test in order to check if the mixture model of three gamma distributions brings enough
#information to justify the higher number of parameters. Due to the p-value<0.05 the model with K=3 is
#effectively the best method according to the AIC and LogLik criterions.


##Management of Outliers
data <- read_csv("C:/Users/acer/Downloads/Wholesale customers data.csv")
```

```
## Rows: 440 Columns: 8
## ── Column specification ────────────────────────────────────────
## Delimiter: ","
## dbl (8): Channel, Region, Fresh, Milk, Gr...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#Outliers from Fresh
par(mfrow= c(1,2))
boxF<-boxplot(data$Fresh)
#Outliers
boxF$out
```

```
##  [1]  43088  56159  44466  40721  43265
##  [6]  56082  76237  42312  45640 112151
## [11]  47493  56083  53205  49063  68951
## [16]  40254  42786  39679  38793  39228
```

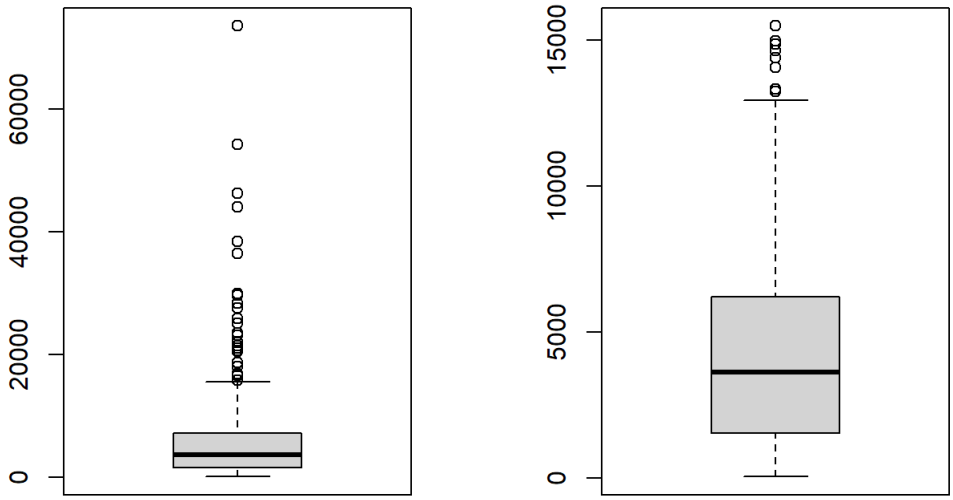
```
#Located
#sapply(data$Fresh, function(x) x %in% boxF$out)
#Replace Outliers with the median
data$Fresh[data$Fresh %in% boxF$out]<- median(data$Fresh)
boxplot(data$Fresh)
```




```
#Outliers from Milk
par(mfrow= c(1,2))
boxM<-boxplot(data$Milk)
boxM$out
```

```
##  [1] 36423 20484 15729 22044 54259 21412
##  [7] 29892 38369 20959 46197 73498 27472
## [13] 16729 15726 25862 29627 43950 28326
## [19] 16599 23133 17972 23527 20655 25071
## [25] 16784 18664 21858 16687
```

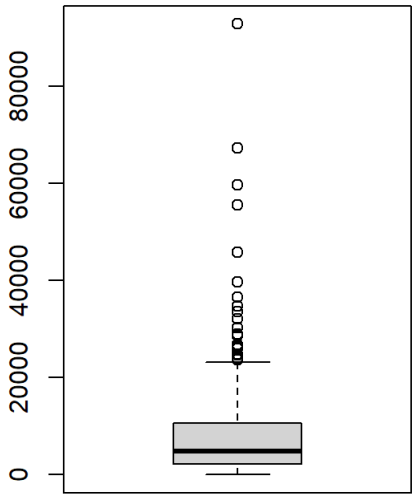
```
data$Milk[data$Milk %in% boxM$out]<- median(data$Milk)
boxplot(data$Milk)
```

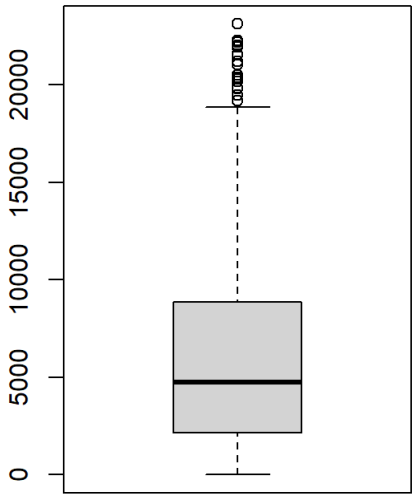



```
#Outliers from Grocery
par(mfrow= c(1,2))
boxG<-boxplot(data$Grocery)
boxG$out
```

```
##  [1] 25957 23998 55571 28921 26866 59598
##  [7] 45828 28540 92780 32114 32034 28986
## [13] 34792 26870 24708 23596 39694 36486
## [19] 33586 24773 26839 67298 26316 30243
```

```
data$Grocery[data$Grocery %in% boxG$out]<- median(data$Grocery)
boxplot(data$Grocery)
```

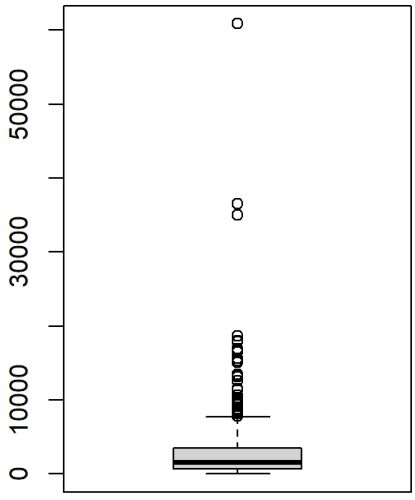


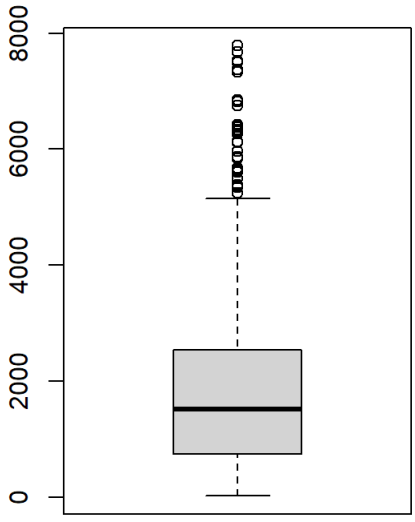



```
#Outliers from Frozen
par(mfrow= c(1,2))
boxFro<-boxplot(data$Frozen)
boxFro$out
```

```
##  [1]  9408 10002  9510 10643  8872  8132
##  [7]  9735  8693 35009 18028  8853 16538
## [13]  8195  8425 16745 36534  7888 18711
## [19]  8321 11422 10155 16919 10303  8692
## [25]  8366 12569 60869  7849 11559  8170
## [31] 15601  9584  8164  9927  8620 13223
## [37]  9806 17866 15348 15082 13486 13135
```

```
data$Frozen[data$Frozen %in% boxFro$out]<- median(data$Frozen)
boxplot(data$Frozen)
```

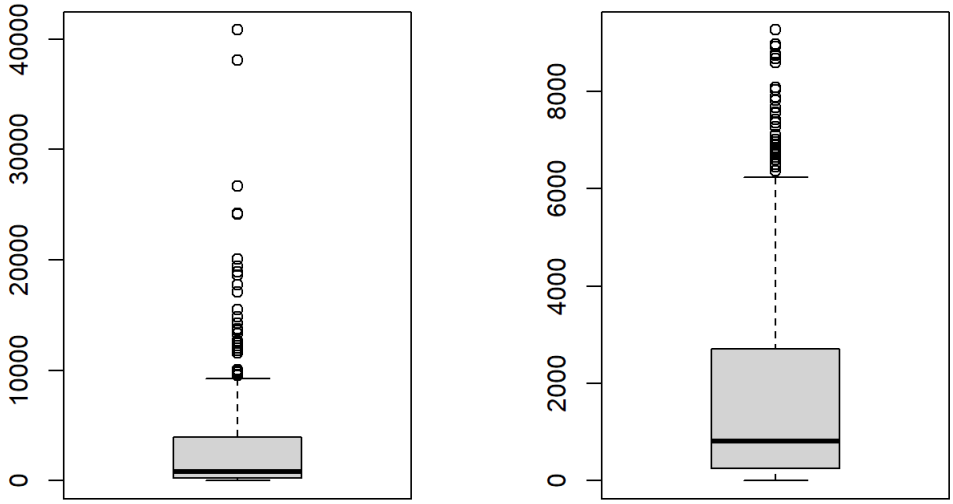





```
#Outliers from Detergents_Paper
par(mfrow= c(1,2))
boxD<-boxplot(data$Detergents_Paper)
boxD$out
```

```
##  [1]  9529 24171 13583 17740 26701 24231
##  [7] 12034 40827 20070 18906 12591 11577
## [13] 13726  9836  9959 14235 12420 19410
## [19] 13308 18594 10069 11783 17120 12408
## [25]  9606 38102 15469 12218 12638 14841
```

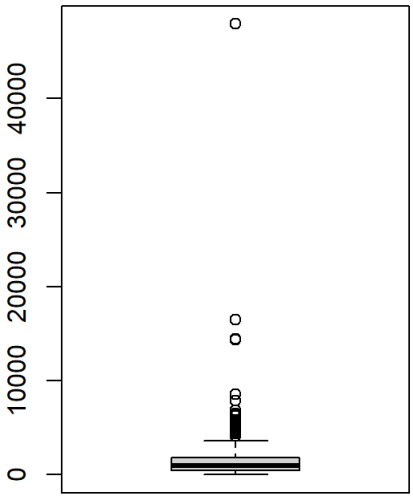
```
data$Detergents_Paper[data$Detergents_Paper %in% boxD$out]<- median(data$Detergents_Paper)
boxplot(data$Detergents_Paper)
```

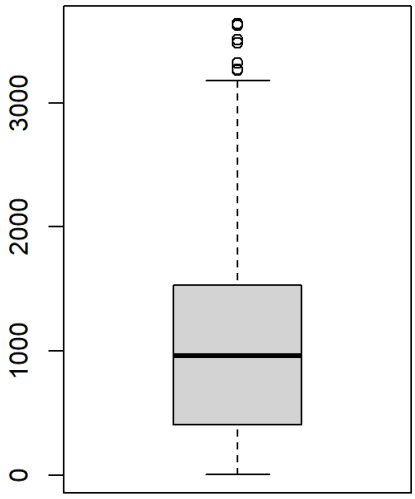



```
#Outliers from Delicassen
par(mfrow= c(1,2))
boxDl<-boxplot(data$Delicassen)
boxDl$out
```

```
##  [1]  7844  5185  4478  4334 16523  5778
##  [7]  5206  4626  5864  4985  6465 14472
## [13] 14351  5130  4430  6250  8550 47943
## [19]  6854  6372  5121  5609  4100  4829
## [25]  5120  5137  4365
```

```
data$Delicassen[data$Delicassen %in% boxDl$out]<- median(data$Delicassen)
boxplot(data$Delicassen)
```






```
#PCA ANALYSIS
#Before to proceed with the PCA analysis,
#we have to select the numeric variable and then scale the data
df<-data[3:8]
summary(df)
```

```
##      Fresh            Milk
##  Min.   :    3   Min.   :   55
##  1st Qu.: 3128   1st Qu.: 1533
##  Median : 8490   Median : 3624
##  Mean   :10033   Mean   : 4282
##  3rd Qu.:14792   3rd Qu.: 6202
##  Max.   :37036   Max.   :15488
##     Grocery          Frozen
##  Min.   :    3   Min.   : 25.0
##  1st Qu.: 2153   1st Qu.: 742.2
##  Median : 4755   Median :1521.5
##  Mean   : 6219   Mean   :1910.9
##  3rd Qu.: 8861   3rd Qu.:2540.0
##  Max.   :23127   Max.   :7782.0
##  Detergents_Paper   Delicassen
##  Min.   :   3.0   Min.   :   3.0
##  1st Qu.: 256.8   1st Qu.: 408.2
##  Median : 814.8   Median : 964.8
##  Mean   :1806.0   Mean   :1084.3
##  3rd Qu.:2688.0   3rd Qu.:1525.0
##  Max.   :9265.0   Max.   :3637.0
```

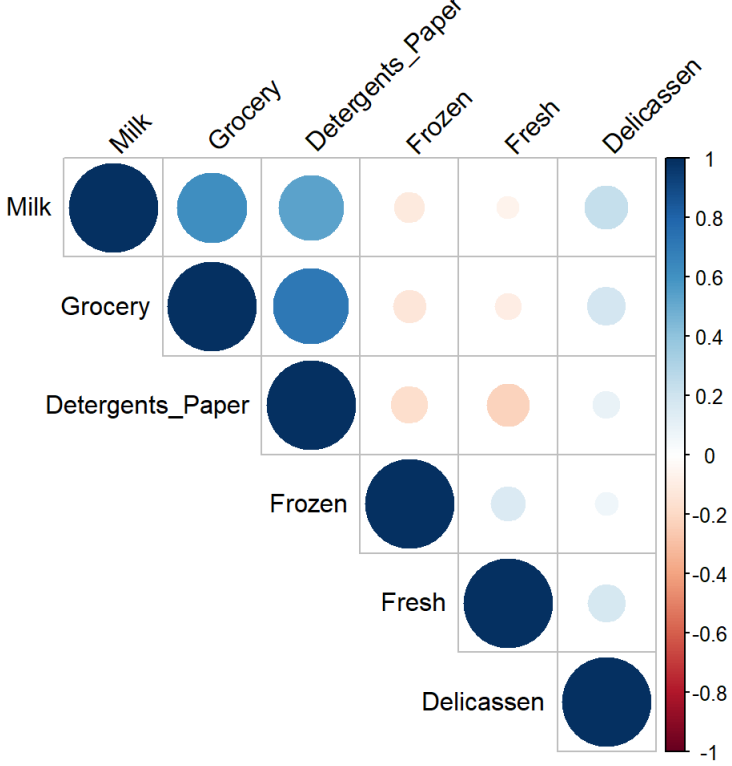
```
#Each originals variable may have different mean,
#so can be usefull centered at zero each variable for the PCA
#in order to compare each principal componenet to the mean straightforward
scale_df<-scale(df)
summary(scale_df)
```

```
##      Fresh              Milk        
##  Min.   :-1.1869   Min.   :-1.2423  
##  1st Qu.:-0.8171   1st Qu.:-0.8079  
##  Median :-0.1827   Median :-0.1935  
##  Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 0.5631   3rd Qu.: 0.5644  
##  Max.   : 3.1954   Max.   : 3.2936  
##     Grocery            Frozen        
##  Min.   :-1.1641   Min.   :-1.1105  
##  1st Qu.:-0.7614   1st Qu.:-0.6882  
##  Median :-0.2742   Median :-0.2293  
##  Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 0.4948   3rd Qu.: 0.3705  
##  Max.   : 3.1665   Max.   : 3.4574  
##  Detergents_Paper    Delicassen       
##  Min.   :-0.7985   Min.   :-1.2866  
##  1st Qu.:-0.6861   1st Qu.:-0.8044  
##  Median :-0.4390   Median :-0.1422  
##  Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 0.3906   3rd Qu.: 0.5244  
##  Max.   : 3.3034   Max.   : 3.0374
```

```
#we can compute the Correlation Matrix
cor(scale_df)
```

```
##                      Fresh       Milk
## Fresh            1.00000000 -0.0635298
## Milk            -0.06352980  1.0000000
## Grocery         -0.09005962  0.6155721
## Frozen           0.15318259 -0.1193893
## Detergents_Paper -0.22581116  0.5368541
## Delicassen       0.17747887  0.2368994
##                     Grocery      Frozen
## Fresh            -0.09005962  0.15318259
## Milk             0.61557206 -0.11938934
## Grocery          1.00000000 -0.13643800
## Frozen          -0.13643800  1.00000000
## Detergents_Paper  0.71138574 -0.17440204
## Delicassen       0.18717077  0.06671472
##                  Detergents_Paper Delicassen
## Fresh                -0.22581116 0.17747887
## Milk                 0.53685411 0.23689940
## Grocery              0.71138574 0.18717077
## Frozen              -0.17440204 0.06671472
## Detergents_Paper     1.00000000 0.09212284
## Delicassen           0.09212284 1.00000000
```

```
par(mfrow= c(1,1))
corrplot(cor(scale_df), type = "upper", order = "hclust", tl.col = "black", tl.srt = 45)
```




```
#The corrplot above show us important stuff:
#•a high positive linear correlation between Detergents_Paper_ and Grocery around 0,71
#•a positive linear correlation between Milk and Grocery around 0,62
#•a positive linear correlation between Milk and Detergents_Paper_ around 0.54

#Calculating the eigenvalues and the eigenvector
cov(scale_df)
```

```
##                      Fresh       Milk
## Fresh            1.00000000 -0.0635298
## Milk            -0.06352980  1.0000000
## Grocery         -0.09005962  0.6155721
## Frozen           0.15318259 -0.1193893
## Detergents_Paper -0.22581116  0.5368541
## Delicassen       0.17747887  0.2368994
##                     Grocery      Frozen
## Fresh           -0.09005962  0.15318259
## Milk             0.61557206 -0.11938934
## Grocery          1.00000000 -0.13643800
## Frozen          -0.13643800  1.00000000
## Detergents_Paper  0.71138574 -0.17440204
## Delicassen       0.18717077  0.06671472
##                  Detergents_Paper Delicassen
## Fresh                -0.22581116 0.17747887
## Milk                  0.53685411 0.23689940
## Grocery               0.71138574 0.18717077
## Frozen               -0.17440204 0.06671472
## Detergents_Paper      1.00000000 0.09212284
## Delicassen            0.09212284 1.00000000
```

```
data.eigen<-eigen(cov(scale_df))
str(data.eigen)
```

```
## List of 2
##  $ values : num [1:6] 2.381 1.292 0.871 0.736 0.453 ...
##  $ vectors: num [1:6, 1:6] 0.15 -0.525 -0.572 0.18 -0.558 ...
##  - attr(*, "class")= chr "eigen"
```

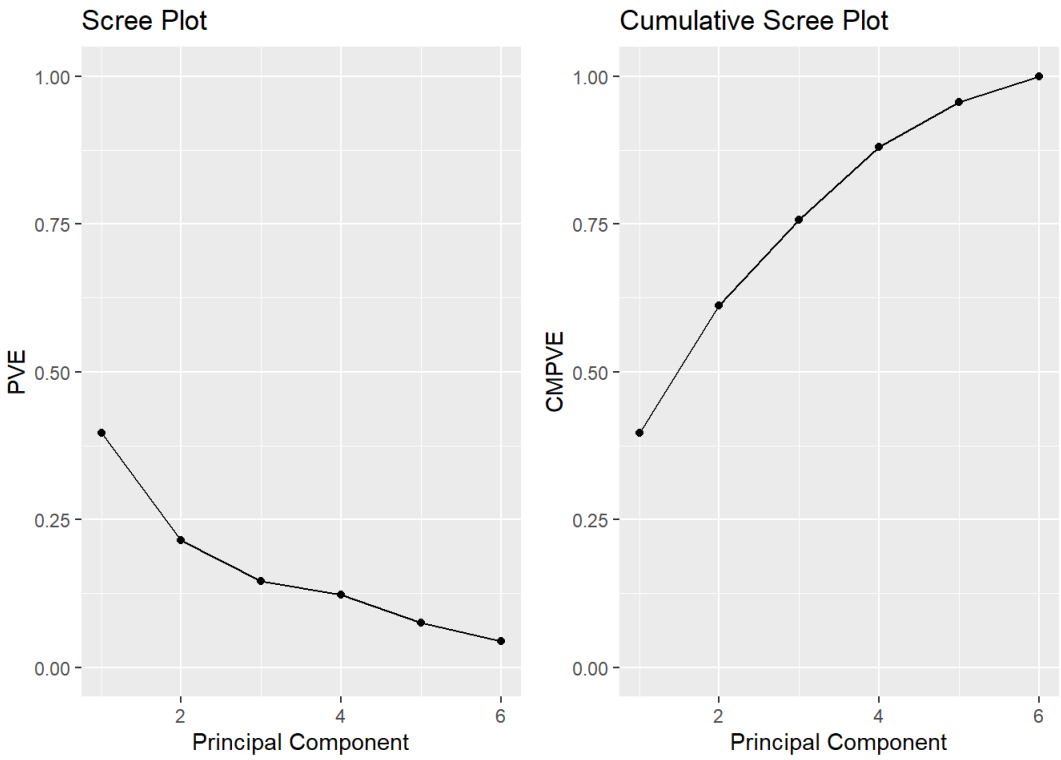
```
##Selecting the PCs
#There are 3 different method to select the right number o PCs

#Proportion of Variance Explained
PVE<- data.eigen$values/sum(data.eigen$values)
round(PVE, 3)
```

```
## [1] 0.397 0.215 0.145 0.123 0.075 0.044
```

```
#The first principal component explain the 39,7% of the variability
#The second principal component explain the 21,5% of the variability
#The third principal component explain the 14,5% of the variability
#Together the first 3 PCs explains the 75,7% of the variability

#It' s convenient to plot the PVE and the CPVE (Cumulative Proportion of Variance Explained) in a scree plot.
PVEplot<-qplot(c(1:6), PVE) + geom_line() + xlab("Principal Component") + ylab("PVE") +ggtitle("Scree Plot") +ylim(0, 1)
```

```
## Warning: `qplot()` was deprecated in ggplot2
## 3.4.0.
```

```
cumPVE <- qplot(c(1:6), cumsum(PVE)) +
  geom_line() +
  xlab("Principal Component") +
  ylab("CMPVE") +
  ggtitle("Cumulative Scree Plot") +
  ylim(0,1)
grid.arrange(PVEplot, cumPVE, ncol = 2)
```




```
#We can estimate the number of PCs by looking for the "elbow point" in the scree plot,
#where the PVE significantly drops off. In this case the perfect number o PCs is equal 3.

#Kaiser's Rule
#This rule suggest to take as many PCs as are the eigenvalues larger than 1.
data.eigen$value[data.eigen$value>1]
```

```
## [1] 2.381492 1.291819
```

```
#According to this rule can be retained the first two PCs

#Principal Component Loadings and Biplot
phi<-data.eigen$vector[,1:3]
phi
```

```
##            [,1]       [,2]       [,3]
## [1,]  0.1498818  0.6266133  0.35499905
## [2,] -0.5245468  0.1505365 -0.02183299
## [3,] -0.5724122  0.0709030 -0.09163610
## [4,]  0.1803474  0.4352485 -0.88172214
## [5,] -0.5576502 -0.1005848 -0.15507480
## [6,] -0.1767244  0.6165296  0.25222943
```

```
#The eigenvector calculated are unique up to a sign flip.
#So is important choice the sign of each eigenvector in order to make the interpretation easier.
#By default eigentvector are computed in the negative direction.

row.names(phi)<- c("Fresh","Milk","Grocery","Frozen","Detergents_Paper","Delicassen")
colnames(phi)<- c("PC1", "PC2","PC3")
phi1<-(phi*-1)
phi1
```

```
##                      PC1        PC2
## Fresh           -0.1498818 -0.6266133
## Milk             0.5245468 -0.1505365
## Grocery          0.5724122 -0.0709030
## Frozen          -0.1803474 -0.4352485
## Detergents_Paper  0.5576502  0.1005848
## Delicassen        0.1767244 -0.6165296
##                      PC3
## Fresh           -0.35499905
## Milk             0.02183299
## Grocery          0.09163610
## Frozen           0.88172214
## Detergents_Paper  0.15507480
## Delicassen       -0.25222943
```

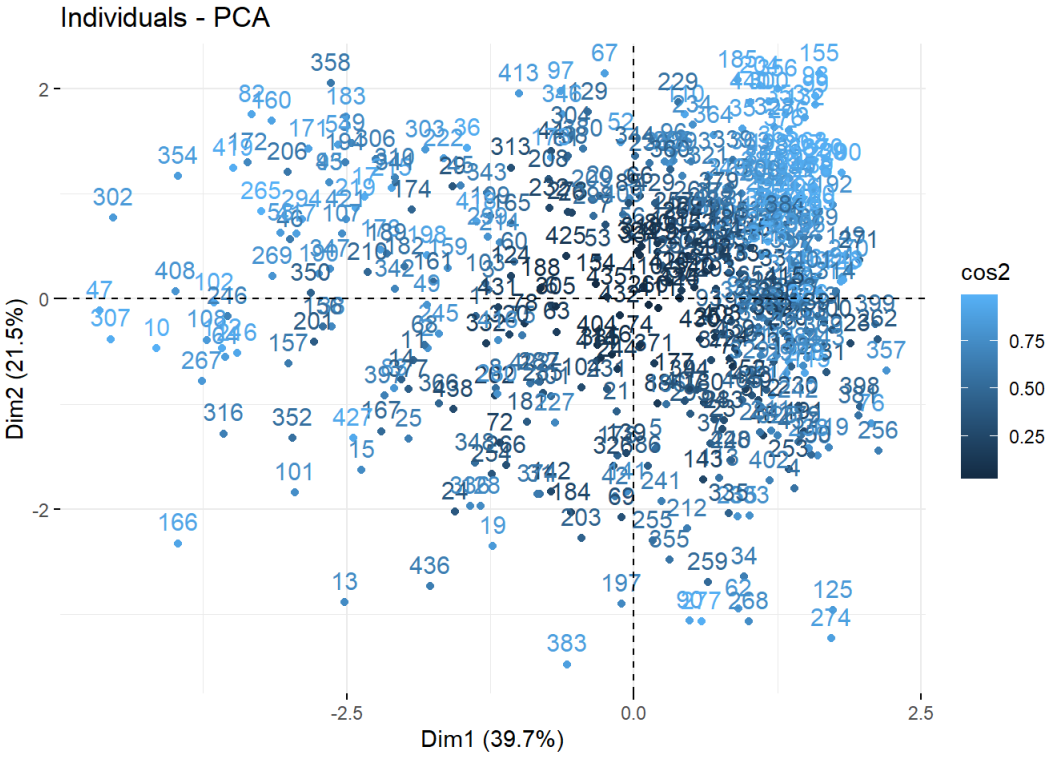
```
#The first loading vector are influenced in equal way from Milk,Grocery and Detergents_Paper,
#while much less by Fresh,Frozen and Delicassen.

#The second loading vector place most of the weight from Fronzen, and also from Fresh and Delicassen,
#with some oppisite influence from Detergents_paper and Grocery.

#The third loading vector take most of the weight from Fresh but in a strong negative way,
#while is positive influenced from Delicassen,Milk,Frozen.

PC1<- scale_df%*%phi1[,1]
PC2<- scale_df%*%phi1[,2]
PC3<- scale_df%*%phi1[,3]
PC<- data.frame(Type_Product=row.names(data), PC1,PC2,PC3)
head(PC)
```

```
##   Type_Product       PC1         PC2
## 1            1  1.3735860 -0.16358349
## 2            2  1.7925886 -0.47141653
## 3            3  1.2649866  0.08984126
## 4            4 -1.3972571 -1.80230698
## 5            5 -0.1892543 -1.42368942
## 6            6  0.7136027 -0.06590935
##          PC3
## 1 -0.95080230
## 2  0.03520021
## 3  0.61843774
## 4  1.84266879
## 5  0.56973833
## 6 -0.72436604
```

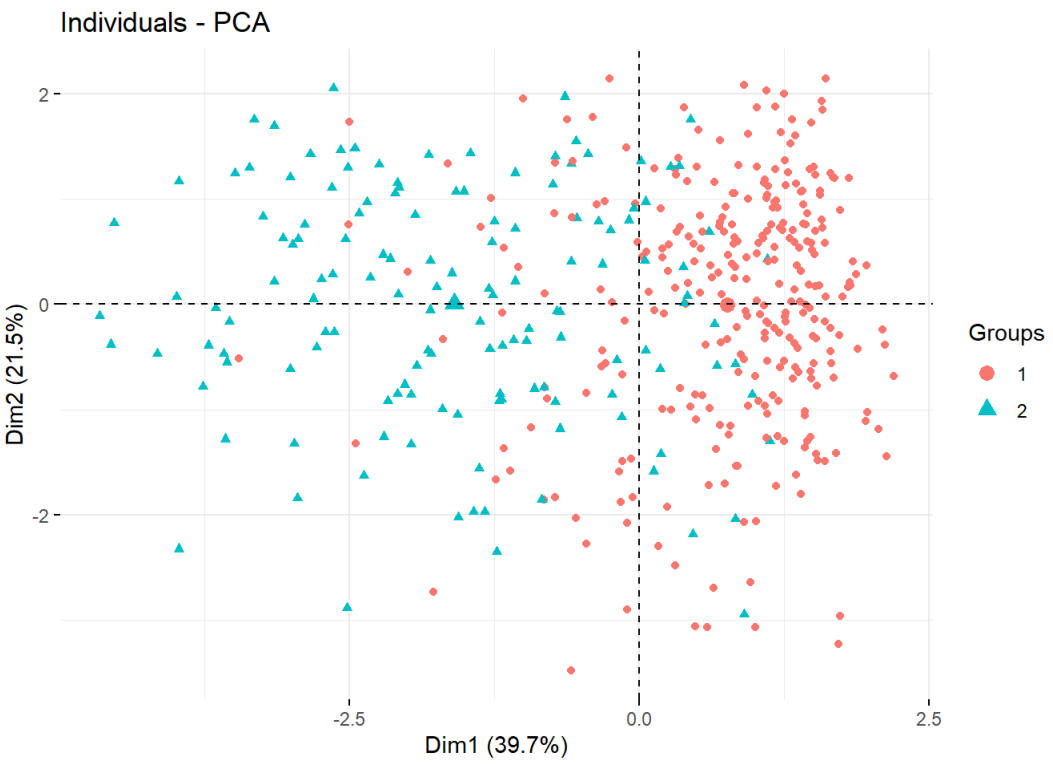
```
#We have computed the first, the second, and the third principal components for each observation.

pr.out=prcomp(df, scale=TRUE)
pr.out$rotation<- -pr.out$rotation
pr.out$rotation[,1:3]
```

```
##                       PC1        PC2
## Fresh            0.1498818 -0.6266133
## Milk            -0.5245468 -0.1505365
## Grocery         -0.5724122 -0.0709030
## Frozen           0.1803474 -0.4352485
## Detergents_Paper -0.5576502  0.1005848
## Delicassen      -0.1767244 -0.6165296
##                       PC3
## Fresh           -0.35499905
## Milk             0.02183299
## Grocery          0.09163610
## Frozen           0.88172214
## Detergents_Paper  0.15507480
## Delicassen      -0.25222943
```

```
pr.out$x<- -pr.out$x


par(mfrow= c(1,1))
fviz_pca_ind(pr.out,col.ind="cos2" )
```




```
par(mfrow= c(1,1))
fviz_pca_ind(pr.out, label="none", habillage=data$Channel)
```




```
pca3d(pr.out, group=data$Region)
```

```
## [1] 0.09306401 0.06945101 0.06558533
## Creating new device
```

```
## Warning: 'rgl.open' è deprecato.
## Usare 'open3d' al suo posto.
## Si veda help("Deprecated")
```

```
## Warning: 'rgl.viewpoint' è deprecato.
## Usare 'view3d' al suo posto.
## Si veda help("Deprecated")
```

```
## Warning: 'rgl.clear' è deprecato.
## Usare 'clear3d' al suo posto.
## Si veda help("Deprecated")
```

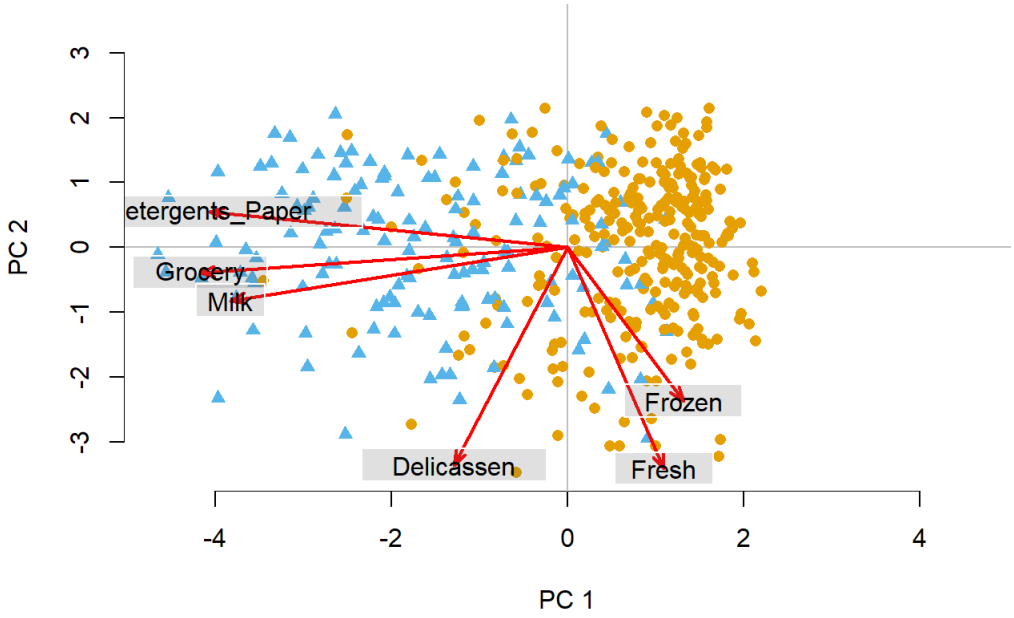
```
#we will use channel for the group
pca3d(pr.out, group=data$Channel, show.ellipses=TRUE,
    ellipse.ci=0.75, show.plane=FALSE)
```

```
## [1] 0.09306401 0.06945101 0.06558533
```

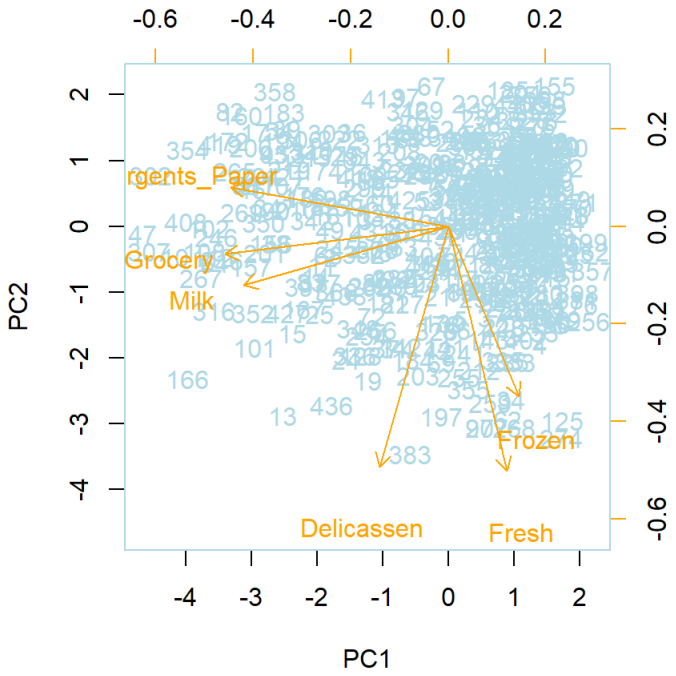
```
## Warning: 'rgl.clear' è deprecato.
## Usare 'clear3d' al suo posto.
## Si veda help("Deprecated")
```

```
pca2d(pr.out, group=data$Channel, biplot=TRUE, biplot.vars=3)
```

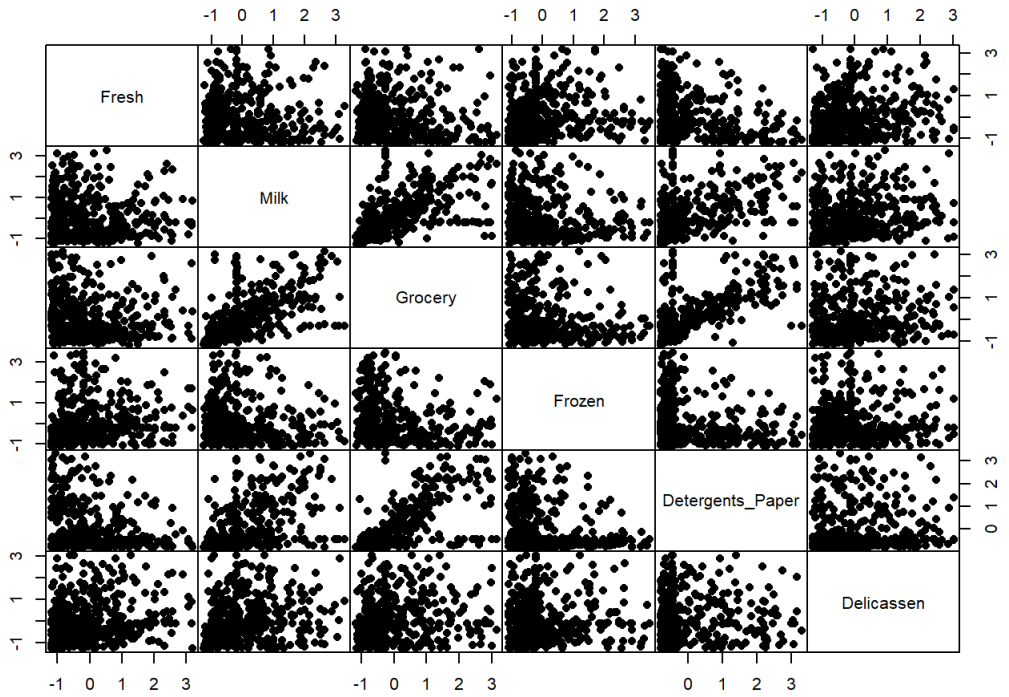
```
## Warning in par(oldpar): argument 1 does not
## name a graphical parameter
```



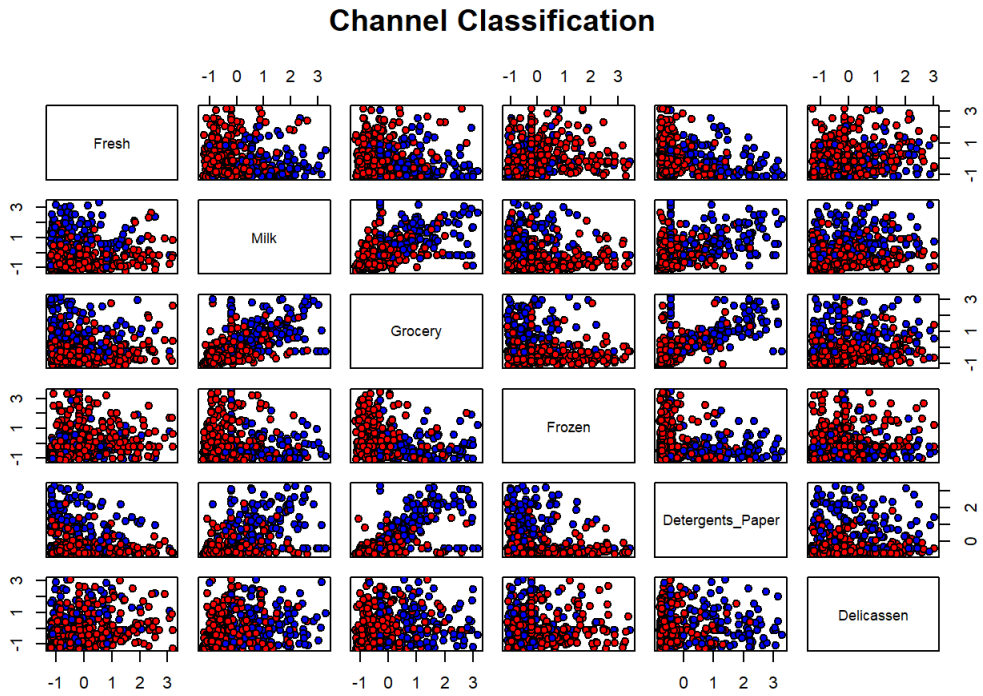

```
biplot(pr.out, scale=0, col=c("lightblue","orange"))
```



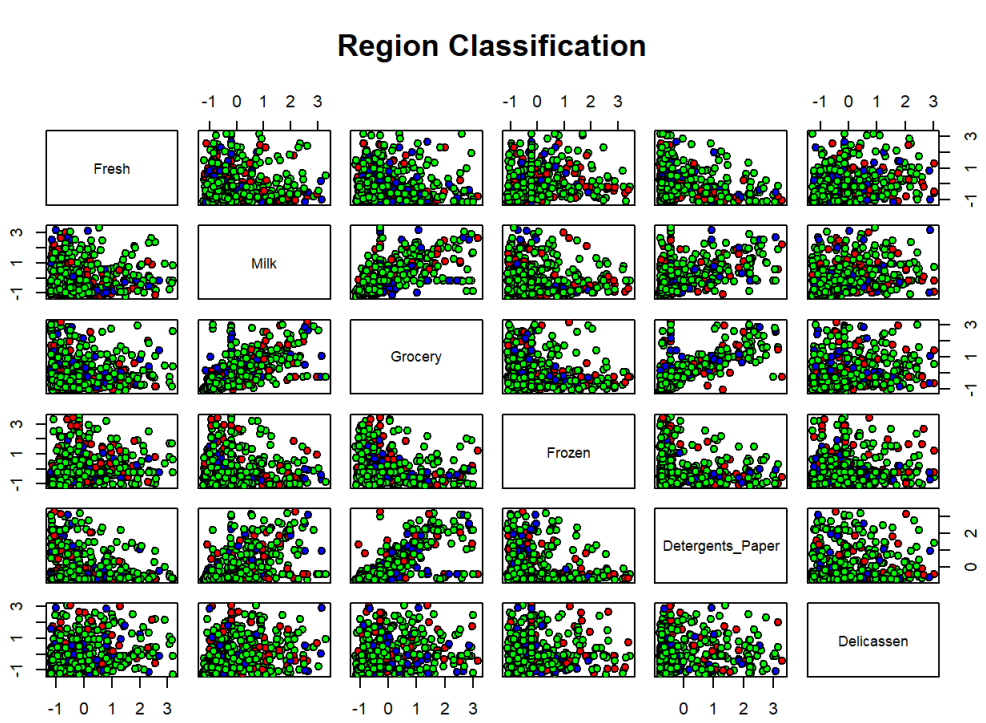

```
## CLUSTER ANALYSIS

#Before applying any clustering method it's important to evaluate whether the data set contains meaningful
#clusters or not.

#Assessing clustering method
#Before analyzing these two method in details, it's necessary to create a a uniform random data set from the
#original data. If the dataset contains clusters it will be possible to see the difference from a dataset created
#through a uniform distribution of data.

random_df<-apply(df,2,function(x){runif(length(x), min(x), max(x))})
random_df<-as.data.frame(random_df)
srandom_df<-scale(random_df)


pairs(scale_df, gap=0, pch=16)
```



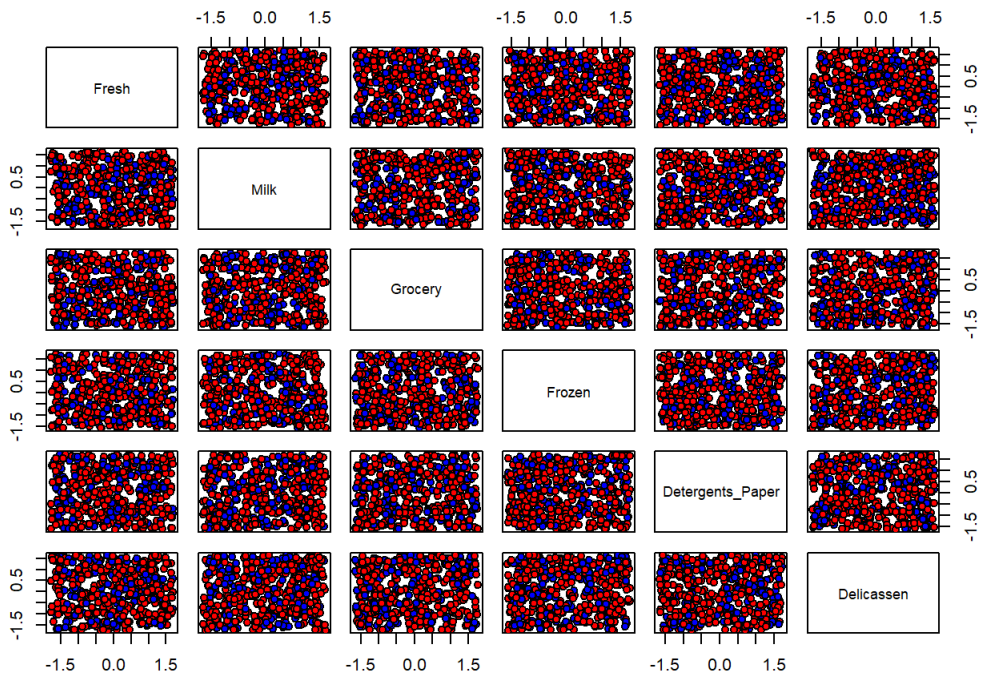

```
pairs(scale_df, main="Channel Classification", pch = 21, bg = c("red", "blue")[unclass(data$Channel)])
```



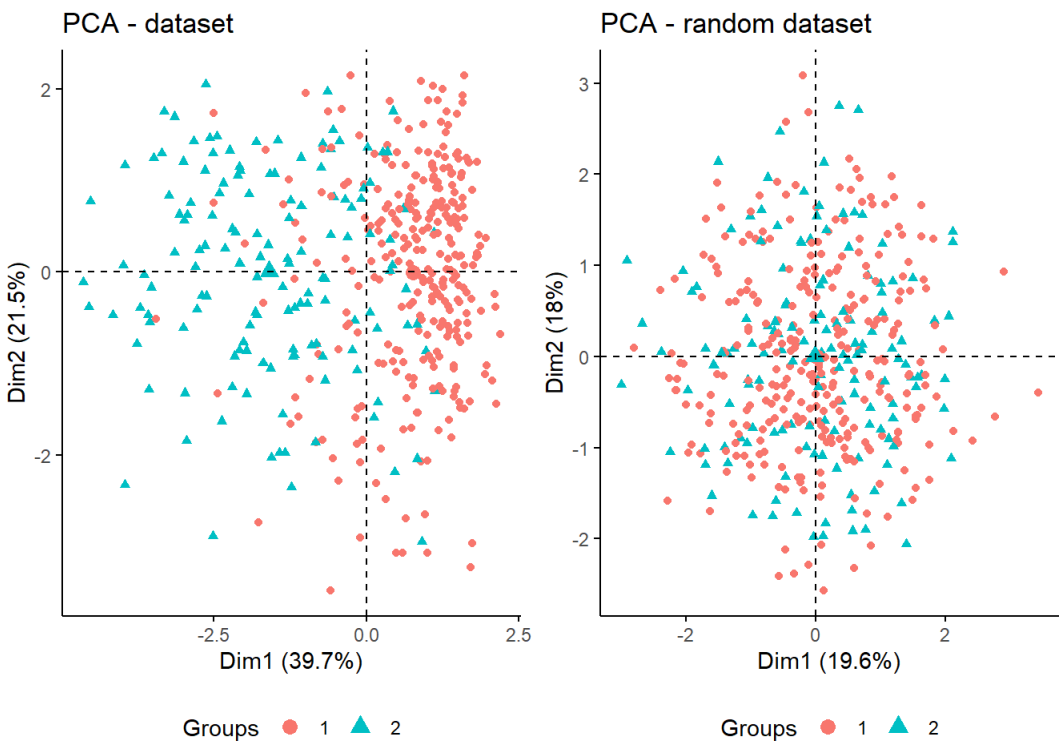

```
pairs(scale_df, main="Region Classification", pch = 21, bg = c("red", "blue","green")[unclass(data$Region)])
```




```
pairs(srandom_df, pch=21 ,bg = c("red", "blue")[unclass(data$Channel)])
```




```
#It can be seen that the standarized randomly generated uniform data do not contain meaningful clusters.

pca_graph<-fviz_pca_ind(pr.out,title="PCA - dataset", habillage=data$Channel, geom = "point", ggtheme=theme_classic(), legend="bottom")
random_pca<-fviz_pca_ind(prcomp(srandom_df),title="PCA - random dataset", habillage=data$Channel, geom = "point", ggtheme=theme_classic(), legend="bottom")
grid.arrange(pca_graph, random_pca, ncol=2)
```




```
pca_3d<-pca3d(pr.out, group=data$Channel)
```

```
## [1] 0.09306401 0.06945101 0.06558533
```

```
## Warning: 'rgl.clear' è deprecato.
## Usare 'clear3d' al suo posto.
## Si veda help("Deprecated")
```

```
pca_r3d<-pca3d(prcomp(srandom_df), group=data$Channel)
```

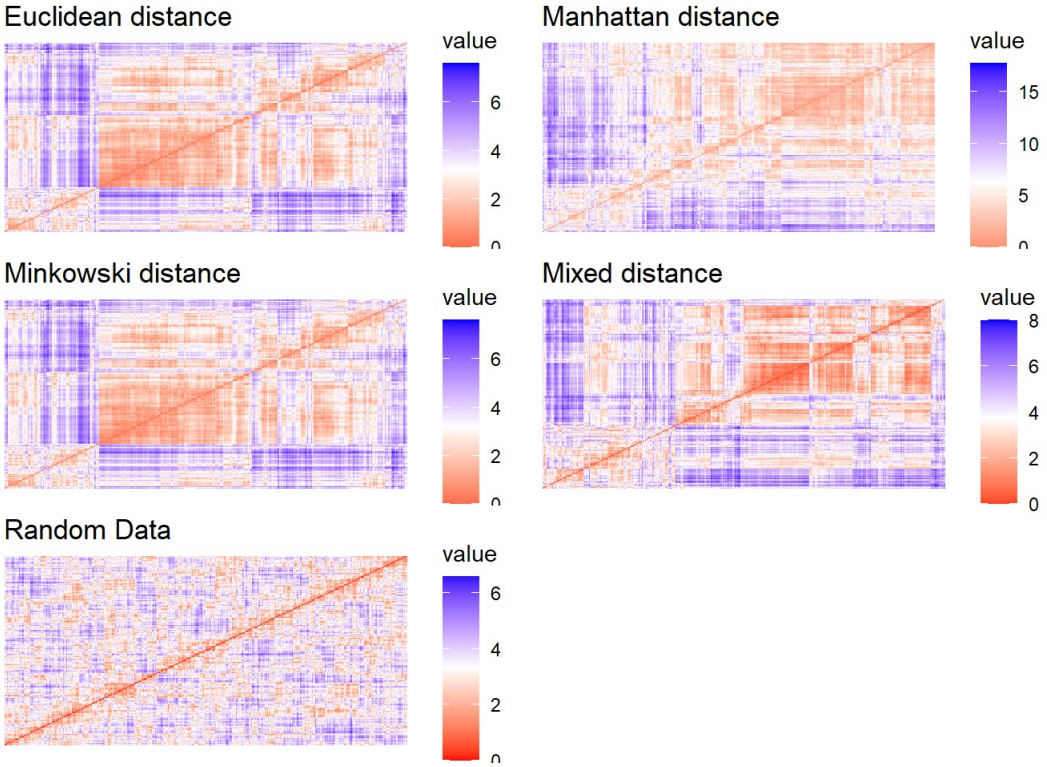
```
## [1] 0.06872988 0.06172070 0.05754572
```

```
## Warning: 'rgl.clear' è deprecato.
## Usare 'clear3d' al suo posto.
## Si veda help("Deprecated")
```

```
## Visual method
D_euc<-dist(scale_df, method="euclidean")
D_man<-dist(scale_df, method="manhattan")
D_min<-dist(scale_df, method="minkowski")
D_mixed<-daisy(scale(data))
```

```
## Warning in daisy(scale(data)): variabili
## binarie 1 trattate come intervallo
## ridimensionato
```

```
VAT.euc<-fviz_dist(D_euc, show_labels=FALSE)+ labs(title="Euclidean distance")
VAT.man<-fviz_dist(D_man, show_labels=FALSE)+ labs(title="Manhattan distance")
VAT.min<-fviz_dist(D_min, show_labels=FALSE)+ labs(title="Minkowski distance")
VAT.mix<-fviz_dist(D_mixed, show_labels=FALSE)+ labs(title="Mixed distance")

VAT.ran<-fviz_dist(dist(srandom_df), show_labels=FALSE)+ labs(title="Random Data")
grid.arrange(VAT.euc,VAT.man,VAT.min,VAT.mix,VAT.ran, ncol=2)
```




```
#the color level is proportional to the value of the dissimilarity btw observations:
#red means high similarity, while blue means low similarity
#in this case the dissimilarity matrix image confirms that there are
#clusters structure in our data but not in the random one

# STATISTICAL METHOD

set.seed(123)
hopkins::hopkins(scale_df, m=nrow(scale_df)-1)
```

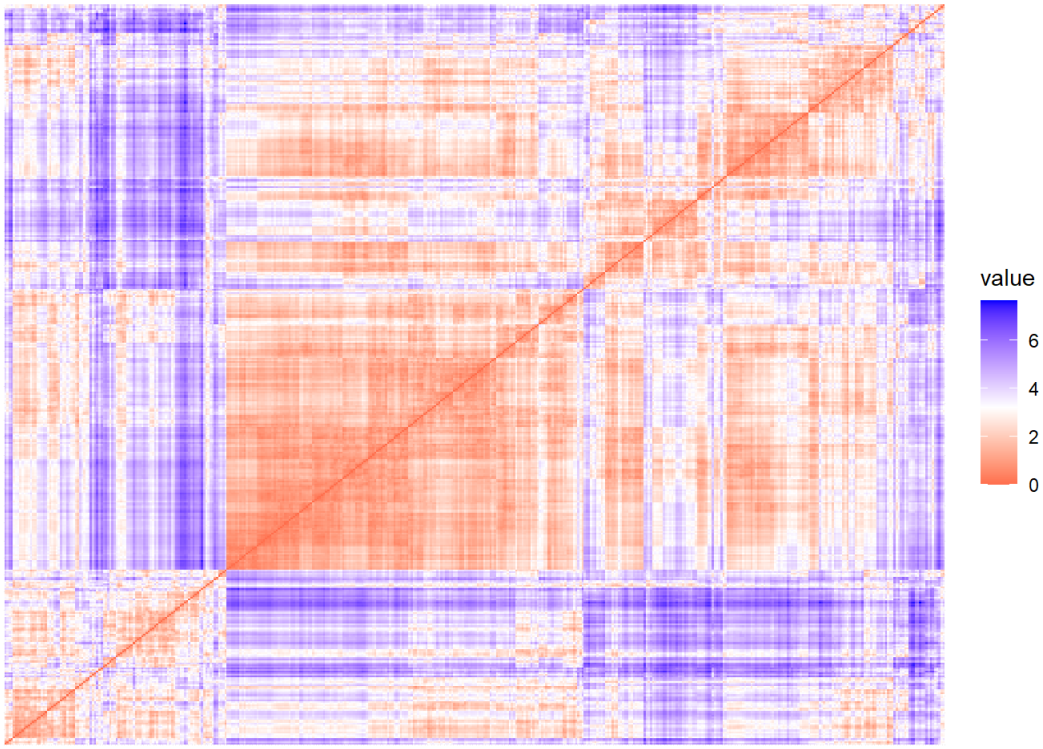
```
## [1] 0.9881067
```

```
hopkins::hopkins(srandom_df, m=nrow(srandom_df)-1)
```

```
## [1] 0.4906487
```

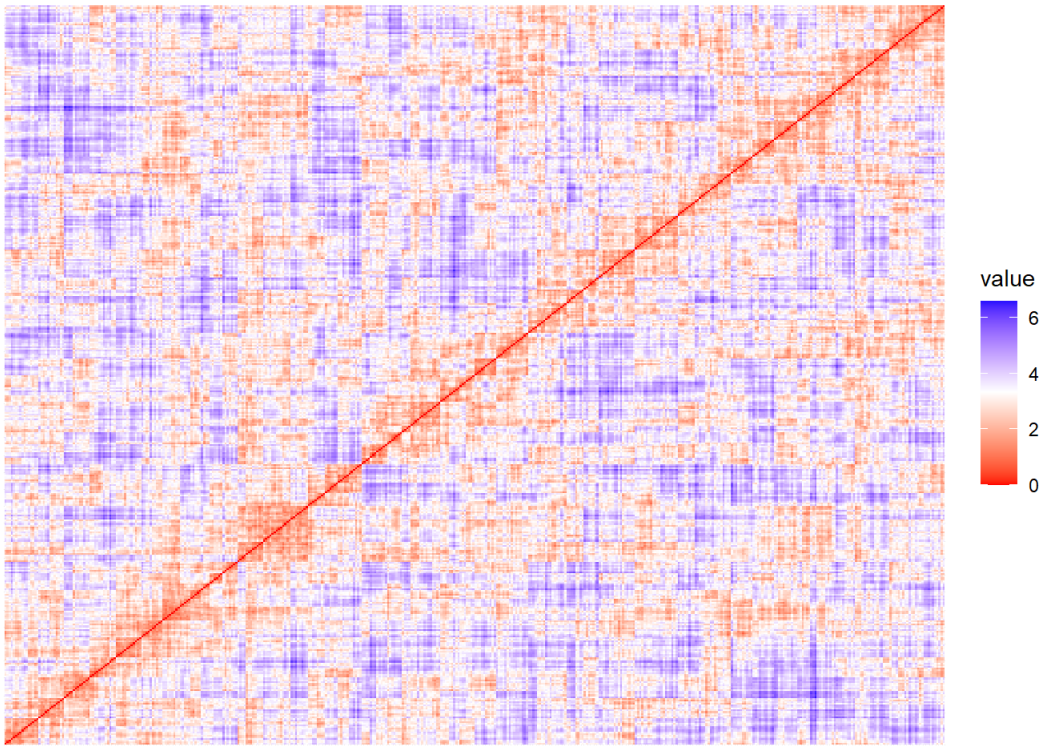
```
#By looking at the Hopkins statistic it's possible to see that the dataset has an high value (close to 1), which
#means that there are clusters. Instead for the random data set the result is a lower value.
#It's also possible to use the "get_clust_tendency" function that provide togheter the Hopkins statistic and the
#VAT algorithm result. Below the function is applied to the scaled data set and to the scaled random data set,
#in order to compare the results:

get_clust_tendency(scale_df, n=nrow(scale_df)-1, graph=TRUE, seed=123)
```

```
## $hopkins_stat
## [1] 0.7118541
##
## $plot
```

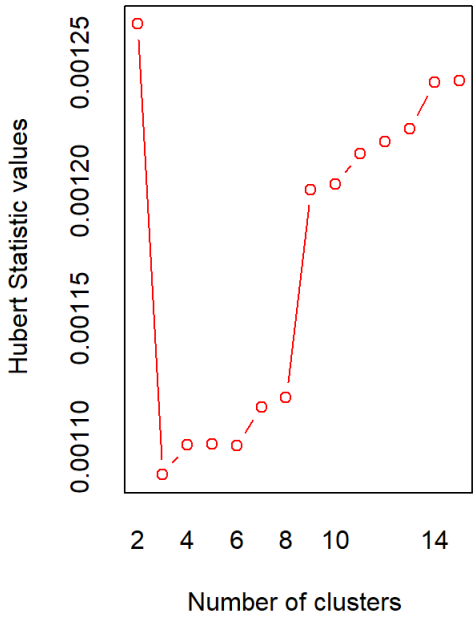


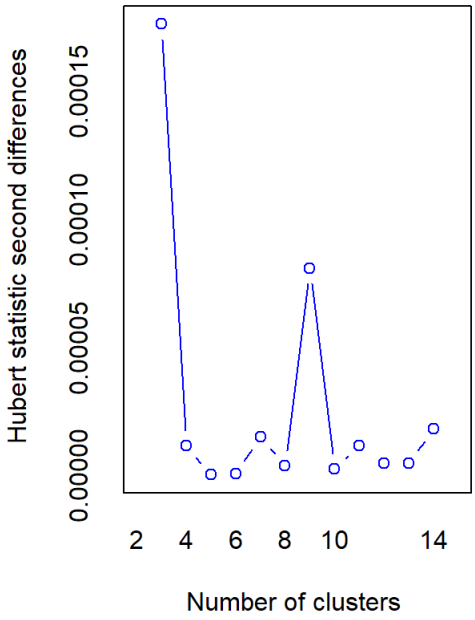

```
get_clust_tendency(srandom_df, n=nrow(srandom_df)-1, graph=TRUE, seed=123)
```

```
## $hopkins_stat
## [1] 0.4935189
##
## $plot
```

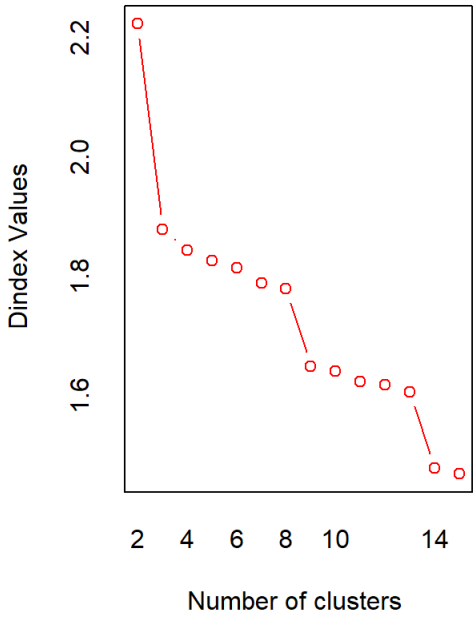


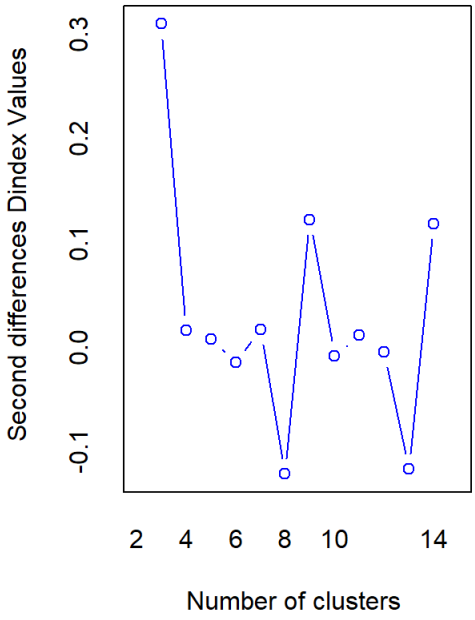

```
## CLUSTERING ANALYSIS
#average linkage-euclidean dist
nb1<-NbClust(scale_df, distance="euclidean", min.nc=2, max.nc=15, method="average")
```






```
## *** : 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.
##
```



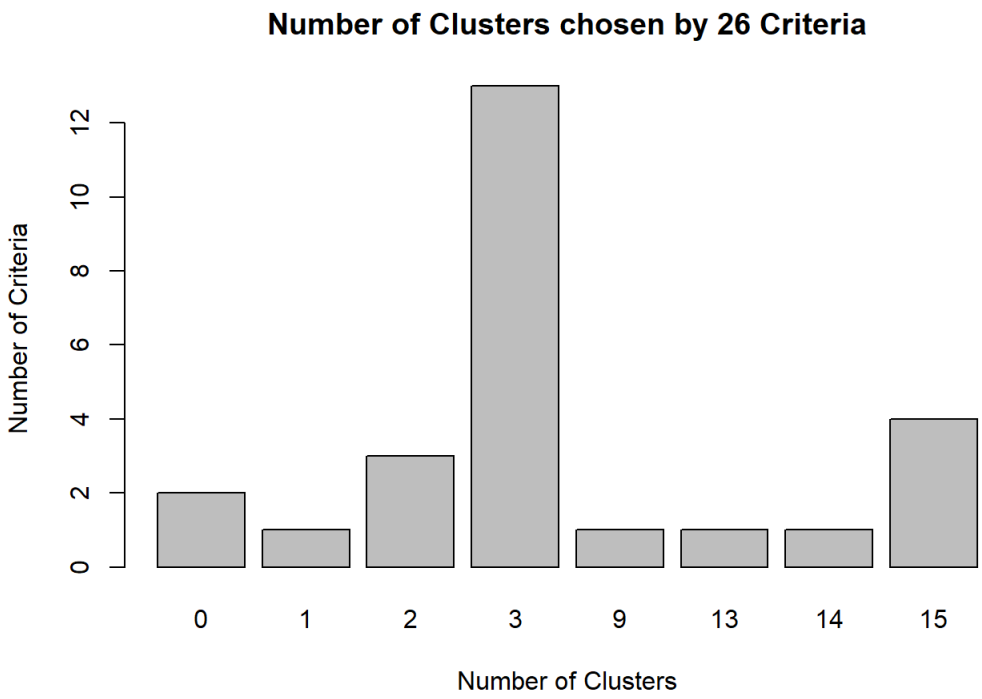


```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 3 proposed 2 as the best number of clusters
## * 13 proposed 3 as the best number of clusters
## * 1 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
## * 4 proposed 15 as the best number of clusters
##
##                   ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *******************************************************************
```

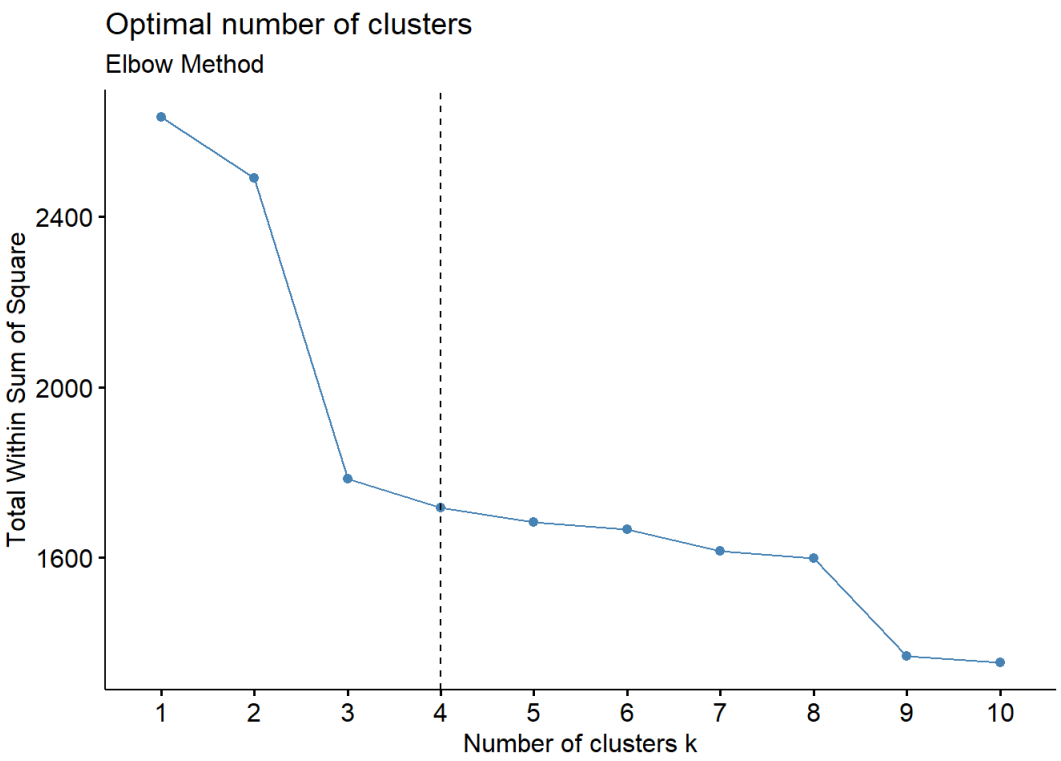
```
summary(nb1)
```

```
##                  Length Class  Mode
## All.index          364    -none- numeric
## All.CriticalValues  42    -none- numeric
## Best.nc             52    -none- numeric
## Best.partition     440    -none- numeric
```

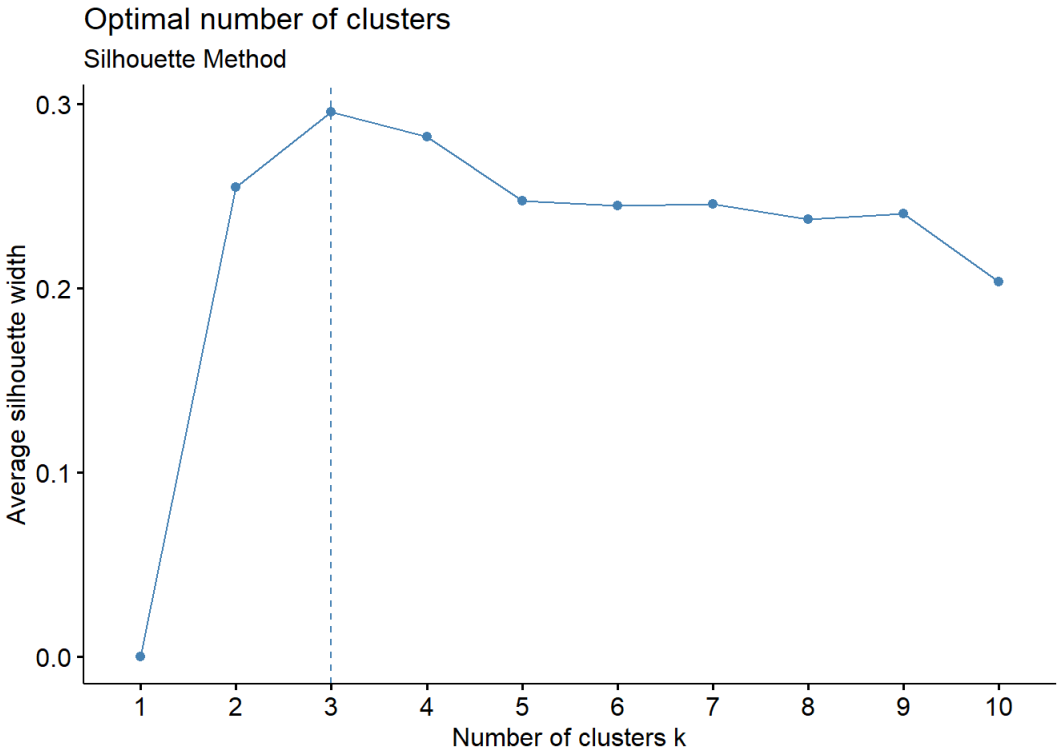
```
t <- table(nb1$Best.nc[1,])

par(mfrow= c(1,1))
barplot(t,xlab="Number of Clusters",ylab="Number of Criteria",
        main="Number of Clusters chosen by 26 Criteria")
```




```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, hcut, method="wss", hc_metric="euclidean", hc_method="average")+geom_vline(xintercept = 4, linetype=2)+labs(subtitle = "Elbow Method")
```

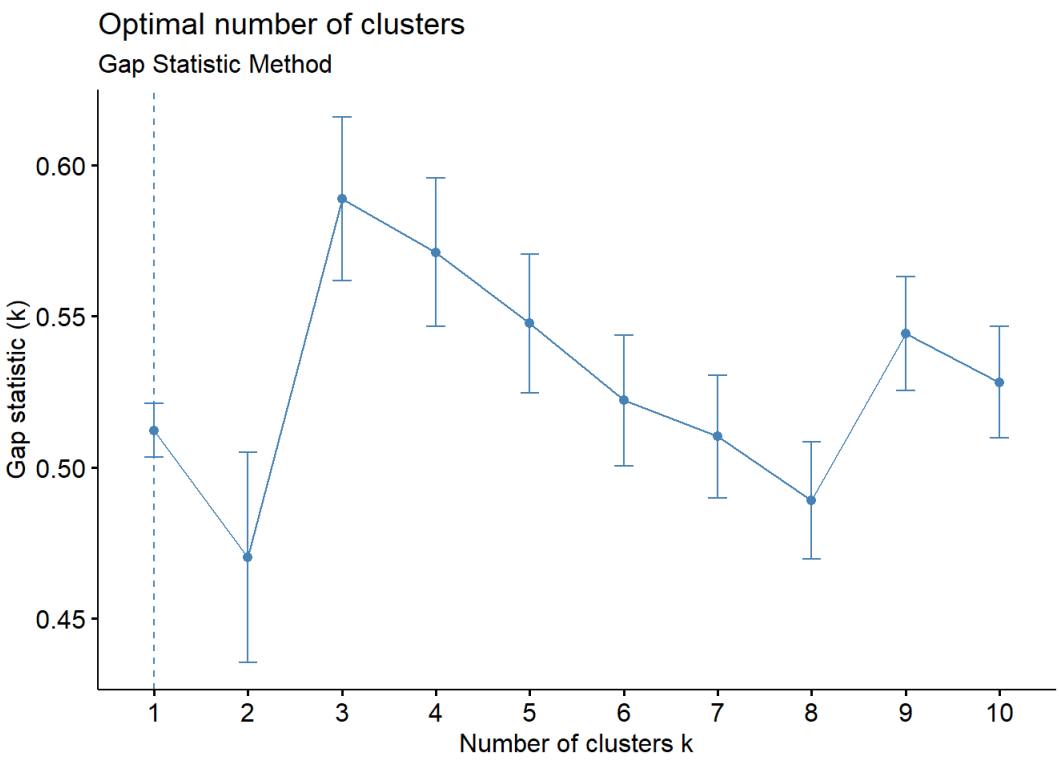



```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, hcut, method="silhouette", hc_metric="euclidean", hc_method="average")+labs(subtitle = "Silhouette Method")
```



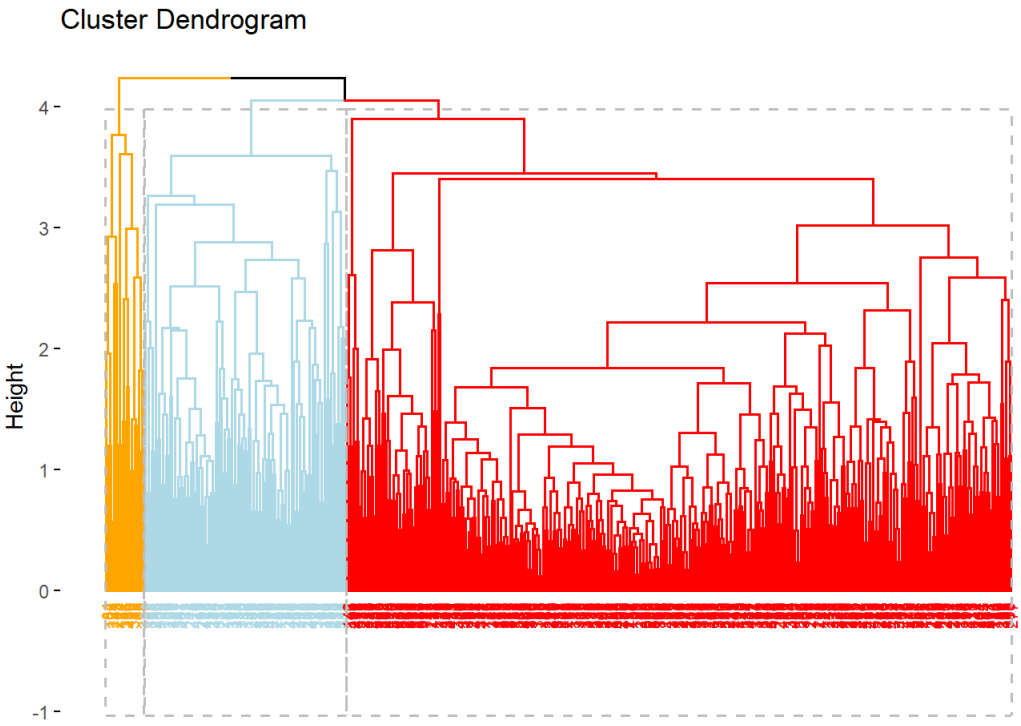

```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, hcut, method="gap_stat", nboot=500, hc_metric="euclidean", hc_method="average")+labs(subtitle = "Gap Statistic Method")
```

```
## 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
```




```
#according with this method, 3 clusters is the most suggested choice
avg.euc.hc<-hclust(d=D_euc, method="average")
par(mfrow=c(1,1))
fviz_dend(avg.euc.hc, k=3, cex=0.5,k_colors=c("orange","lightblue","red"),rect=TRUE)
```

```
## Warning: The `<scale>` argument of `guides()` cannot
## be `FALSE`. Use "none" instead as of ggplot2
## 3.3.4.
## ℹ The deprecated feature was likely used in
##   the factoextra package.
##   Please report the issue at
##   <]8;;https://github.com/kassambara/factoextra/issueshttps://github.com/kassambara/factoextra/issues]8;;>.
```



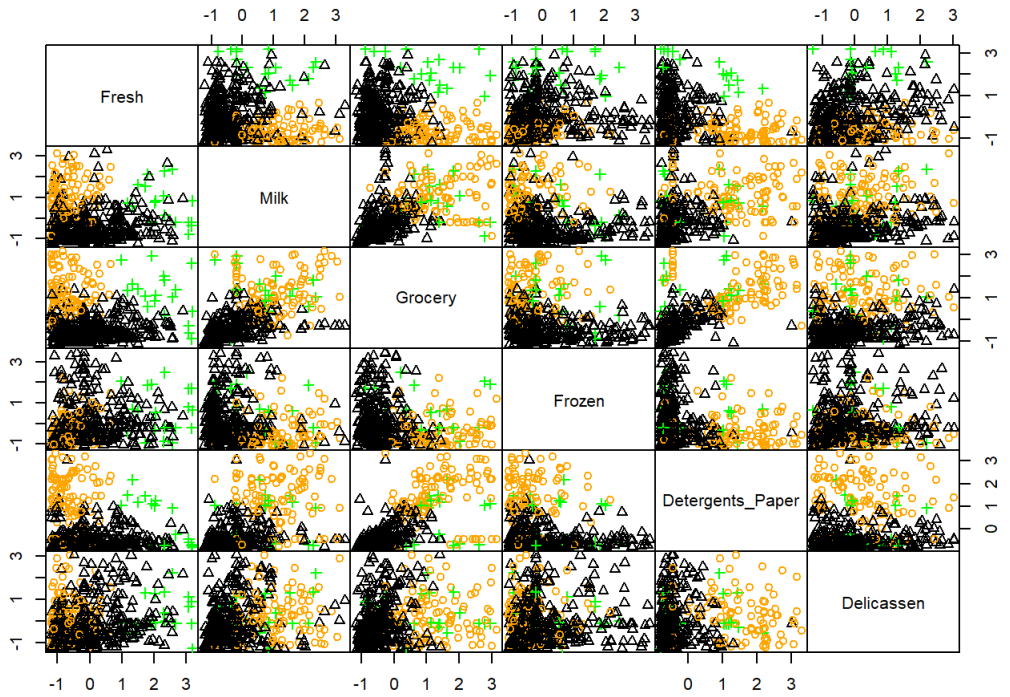

```
avg.euc.coph<-cophenetic(avg.euc.hc)
cor(D_euc, avg.euc.coph)
```

```
## [1] 0.7317355
```

```
grp.avg.euc<-cutree(avg.euc.hc, k=3)
table(grp.avg.euc)
```

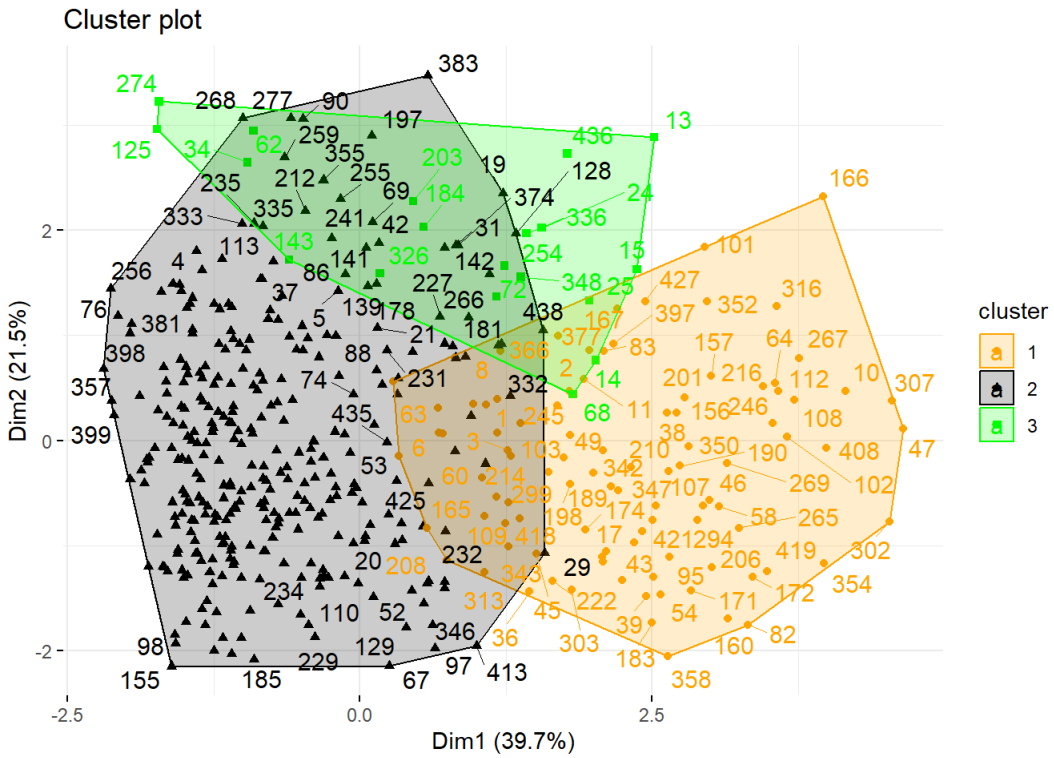
```
## grp.avg.euc
##   1   2   3
##  98 323  19
```

```
par(mfrow=c(1,1))
pairs(scale_df, gap=0, pch=grp.avg.euc, col=c("orange","black","green")[grp.avg.euc])
```



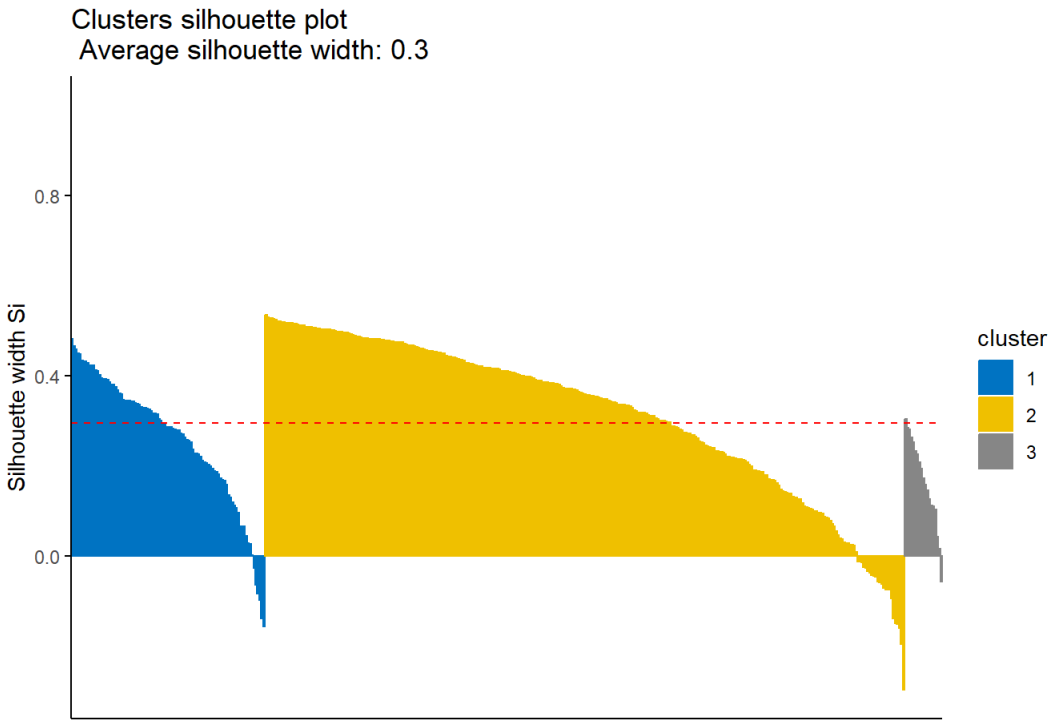

```
par(mfrow=c(1,1))
fviz_cluster(list(data=scale_df, cluster=grp.avg.euc), palette=c("orange","black","green"),
             ellipse.type="convex",repel=TRUE, show.clust.cent=FALSE, ggtheme= theme_minimal())
```

```
## Warning: ggrepel: 282 unlabeled data points
## (too many overlaps). Consider increasing
## max.overlaps
```



```
###internal validation silhouette and dunn index

avg.res<-eclust(scale_df, "hclust", k=3, hc_metric="euclidean", hc_method="average", graph=FALSE)
fviz_silhouette(avg.res, palette="jco", ggtheme=theme_classic())
```

```
##   cluster size ave.sil.width
## 1       1   98          0.26
## 2       2  323          0.31
## 3       3   19          0.17
```




```
silinfo.avg<-avg.res$silinfo
neg_sil.avg<-which(silinfo.avg$widths[,"sil_width"]<0)

silinfo.avg$avg.width
```

```
## [1] 0.2956008
```

```
silinfo.avg$clus.avg.widths
```

```
## [1] 0.2599226 0.3139711 0.1673303
```

```
silinfo.avg$widths[neg_sil.avg,,drop=FALSE]
```

```
##     cluster neighbor   sil_width
## 63        1        2 -0.02563479
## 305       1        2 -0.06412231
## 6         1        2 -0.08347490
## 273       1        2 -0.09634882
## 385       1        2 -0.13796397
## 154       1        2 -0.15591055
## 23        2        3 -0.01207125
## 31        2        3 -0.01263058
## 404       2        3 -0.01401884
## 90        2        3 -0.02396903
## 266       2        1 -0.02539139
## 188       2        1 -0.03295182
## 428       2        3 -0.03700237
## 37        2        3 -0.04267416
## 312       2        3 -0.04406270
## 5         2        3 -0.04517983
## 332       2        1 -0.05557039
## 202       2        1 -0.05876409
## 124       2        1 -0.06118529
## 227       2        3 -0.07059125
## 438       2        1 -0.07357014
## 280       2        1 -0.07380871
## 413       2        1 -0.07452077
## 19        2        3 -0.09355257
## 277       2        3 -0.13796946
## 197       2        3 -0.14862248
## 128       2        3 -0.15028717
## 142       2        3 -0.15855266
## 29        2        1 -0.19497321
## 383       2        3 -0.29574125
## 68        3        1 -0.05625372
```

```
avg.link<-cluster.stats(D_euc, avg.res$cluster)
avg.link$dunn
```

```
## [1] 0.1278857
```

```
#External validation
#confusion matrix
table(data$Channel, avg.res$cluster)
```

```
##
##      1   2   3
##   1  16 272  10
##   2  82  51   9
```

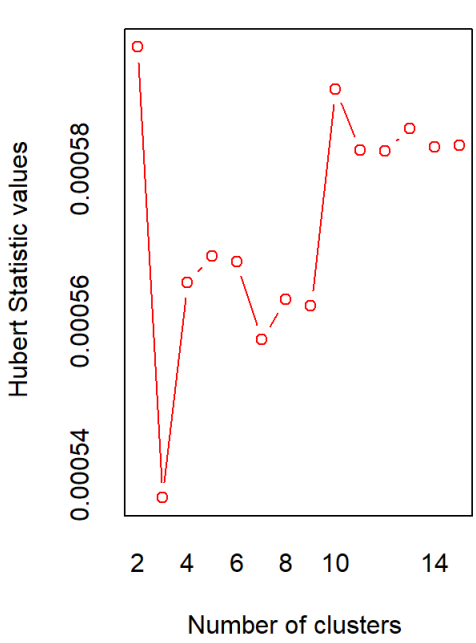
```
#rand index
tipes<-as.numeric(data$Channel)
clust_stats.avg<-cluster.stats(d=D_euc, tipes, avg.res$cluster)
clust_stats.avg$corrected.rand
```

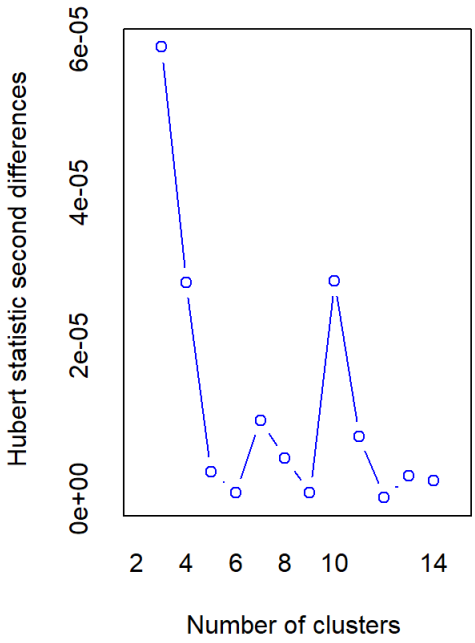
```
## [1] 0.4094852
```

```
#The ARI provides an index that is close to 0 because it takes into account the chance of overlap.
#In addition, note that the ARI is a negative value indicating that the amount of overlap is less than expected.
##Meila's VI Index
clust_stats.avg$vi
```

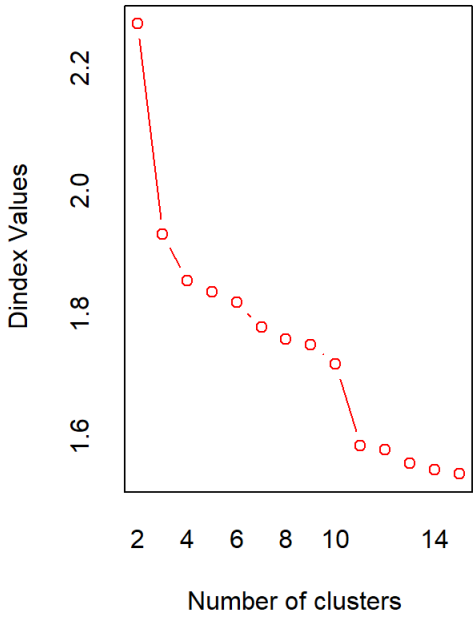
```
## [1] 0.966557
```

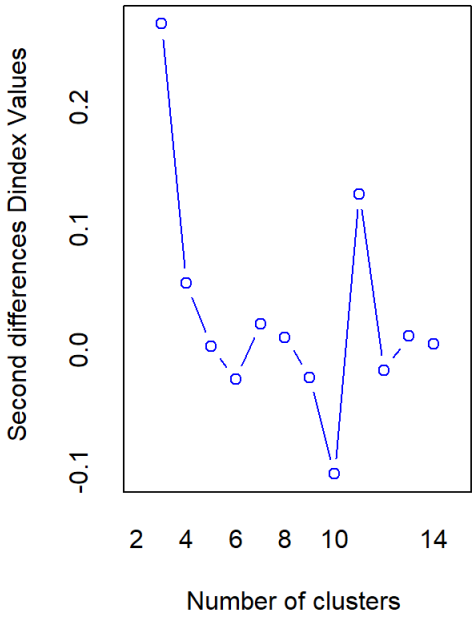
```
#average linkage method (Manhattan dist)
nb2<-NbClust(scale_df, distance="manhattan", method="average")
```






```
## *** : 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.
##
```



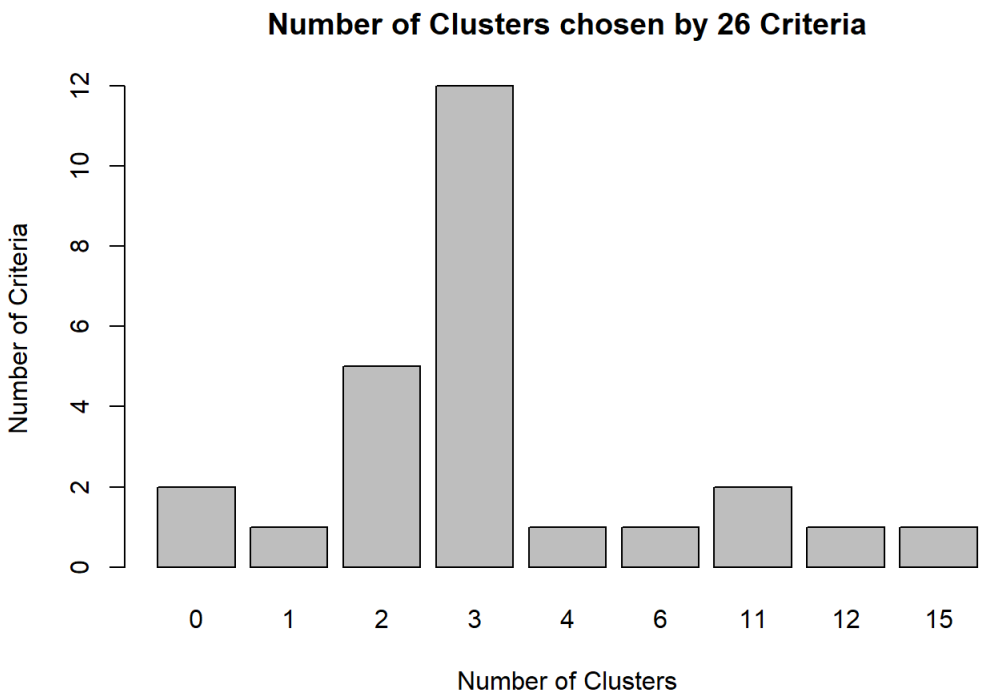


```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 5 proposed 2 as the best number of clusters
## * 12 proposed 3 as the best number of clusters
## * 1 proposed 4 as the best number of clusters
## * 1 proposed 6 as the best number of clusters
## * 2 proposed 11 as the best number of clusters
## * 1 proposed 12 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 is  3
##
##
## *******************************************************************
```

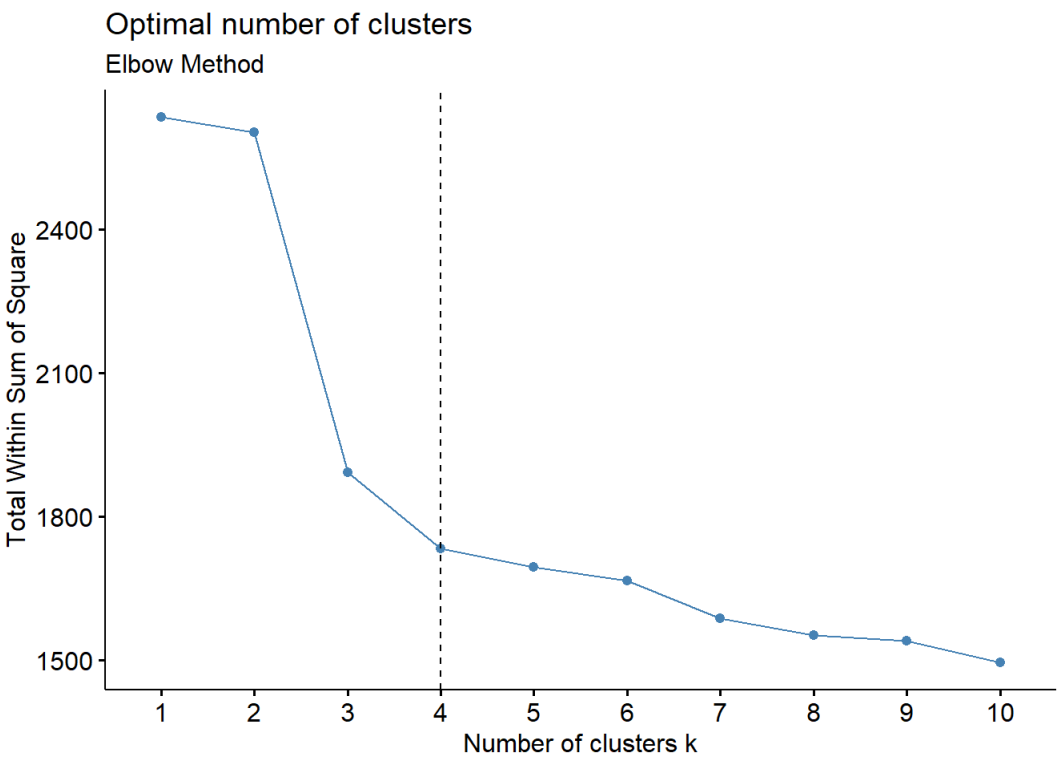
```
summary(nb2)
```

```
##                  Length Class  Mode
## All.index          364    -none- numeric
## All.CriticalValues  42    -none- numeric
## Best.nc             52    -none- numeric
## Best.partition     440    -none- numeric
```

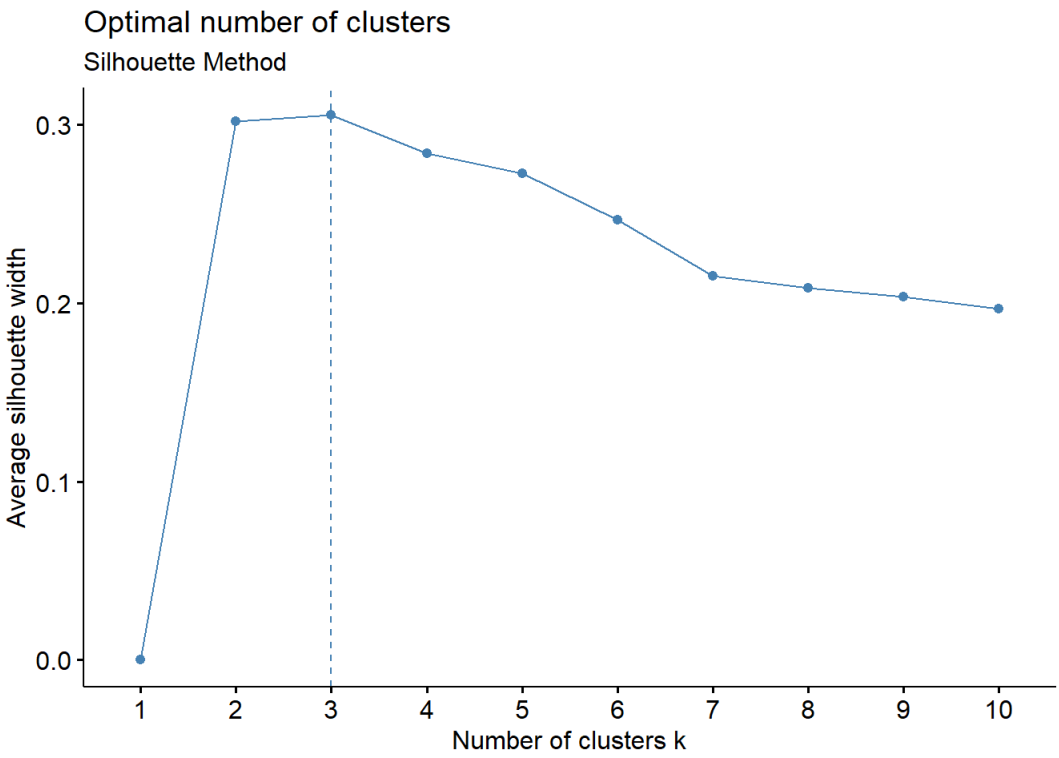
```
t2 <- table(nb2$Best.nc[1,])

par(mfrow= c(1,1))
barplot(t2,xlab="Number of Clusters",ylab="Number of Criteria",
        main="Number of Clusters chosen by 26 Criteria")
```




```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, hcut, method="wss", hc_metric="manhattan", hc_method="average")+geom_vline(xintercept = 4, linetype=2)+labs(subtitle = "Elbow Method")
```

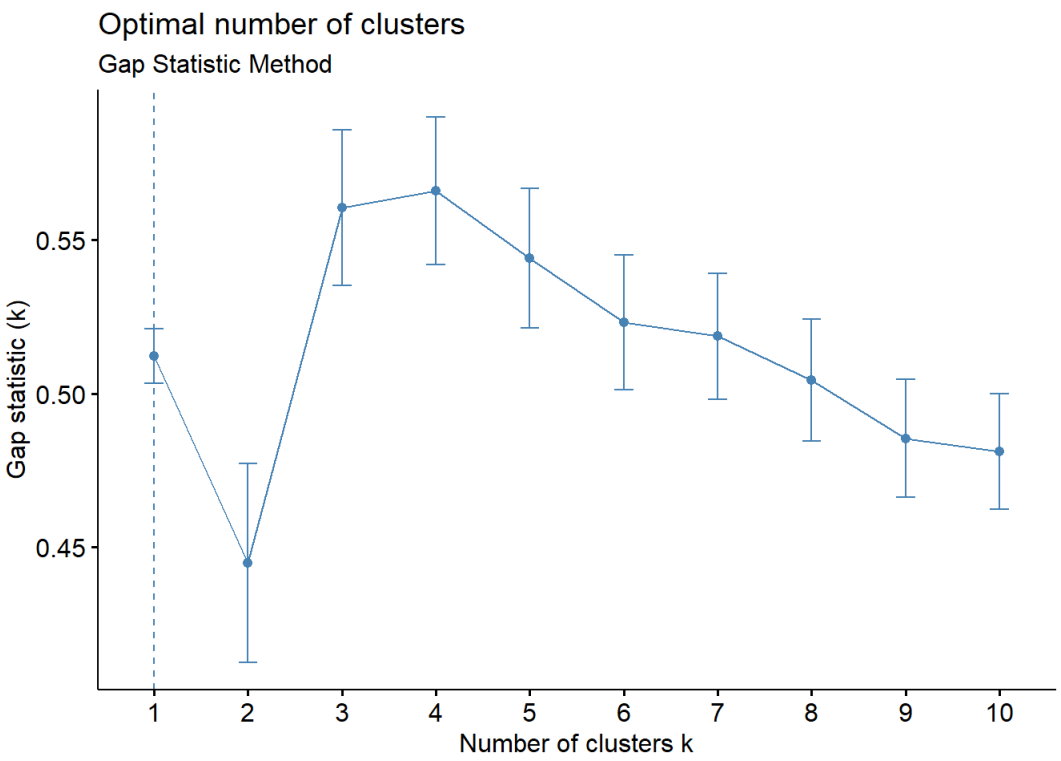



```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, hcut, method="silhouette", hc_metric="manhattan", hc_method="average")+labs(subtitle = "Silhouette Method")
```

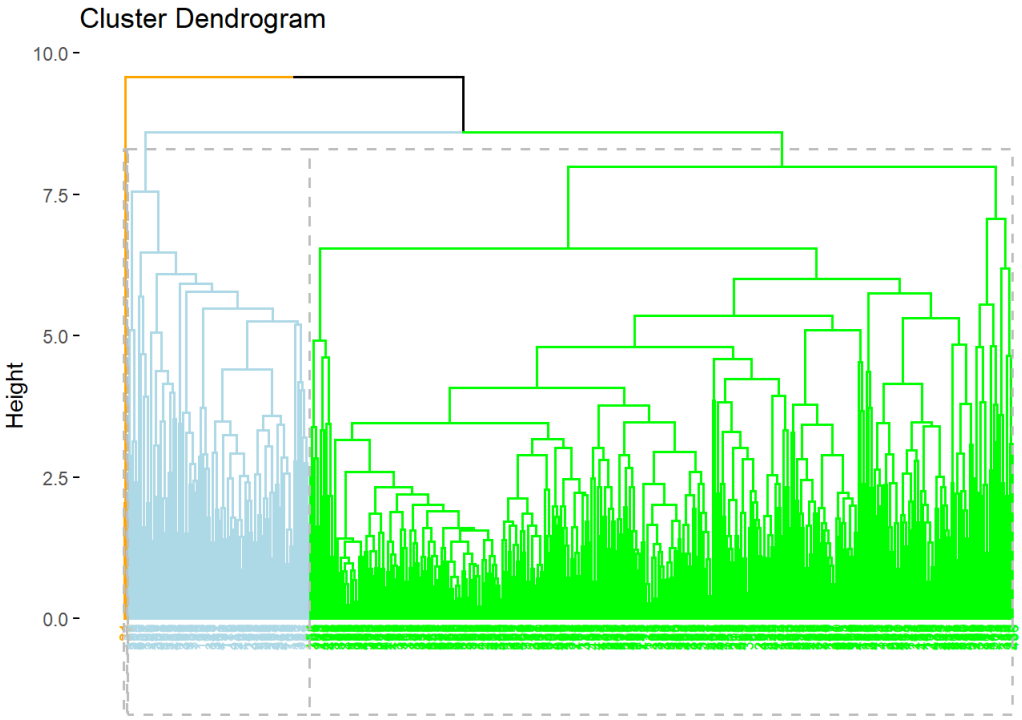



```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, hcut, method="gap_stat", nboot=500, hc_metric="manhattan", hc_method="average")+labs(subtitle = "Gap Statistic Method")
```

```
## 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
```




```
#according with this method, three clusters is the most suggested choice
avg.man.hc<-hclust(d=D_man, method="average")

par(mfrow=c(1,1))
fviz_dend(avg.man.hc, k=3, cex=0.5,k_colors=c("orange","lightblue","green"),rect=TRUE)
```



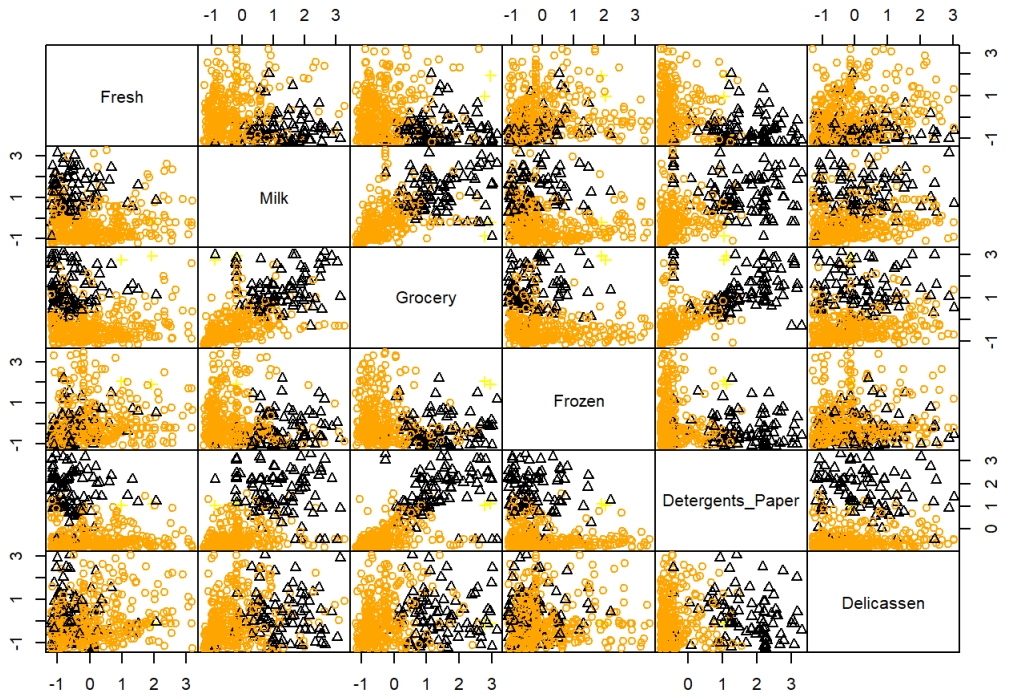

```
avg.man.coph<-cophenetic(avg.man.hc)
cor(D_euc, avg.man.coph)
```

```
## [1] 0.7183053
```

```
grp.man<-cutree(avg.man.hc, k=3)
table(grp.man)
```

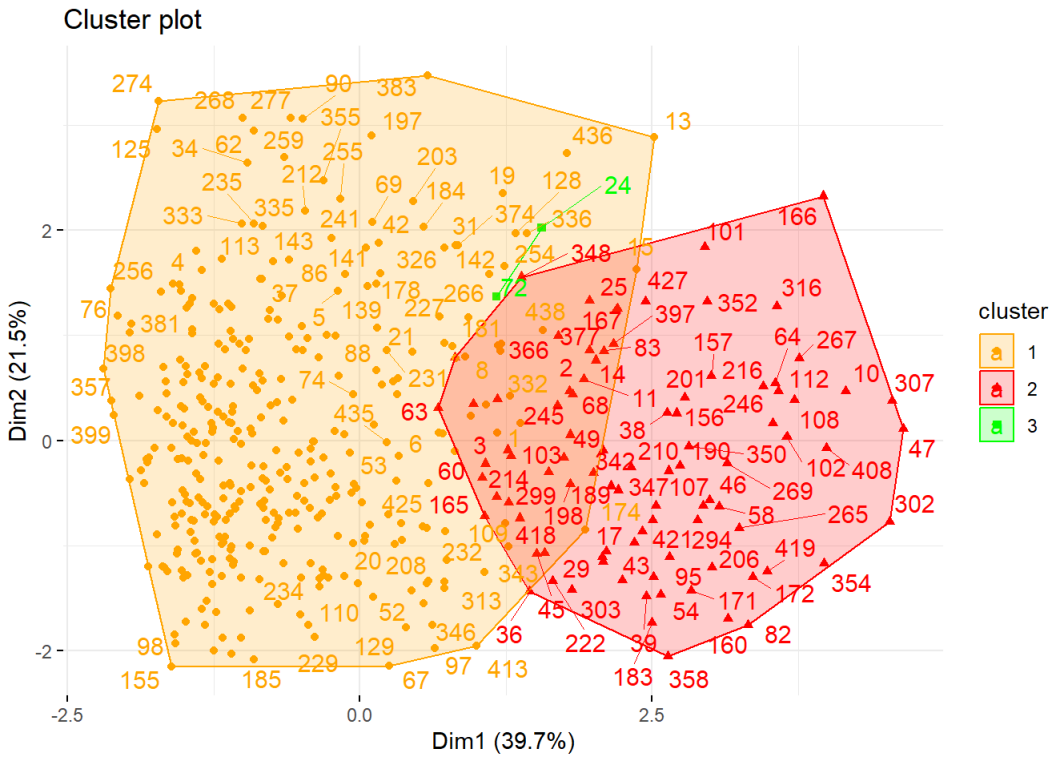
```
## grp.man
##   1   2   3
## 348  90   2
```

```
par(mfrow=c(1,1))
pairs(scale_df, gap=0, pch=grp.man, col=c("orange","black","yellow")[grp.man])
```



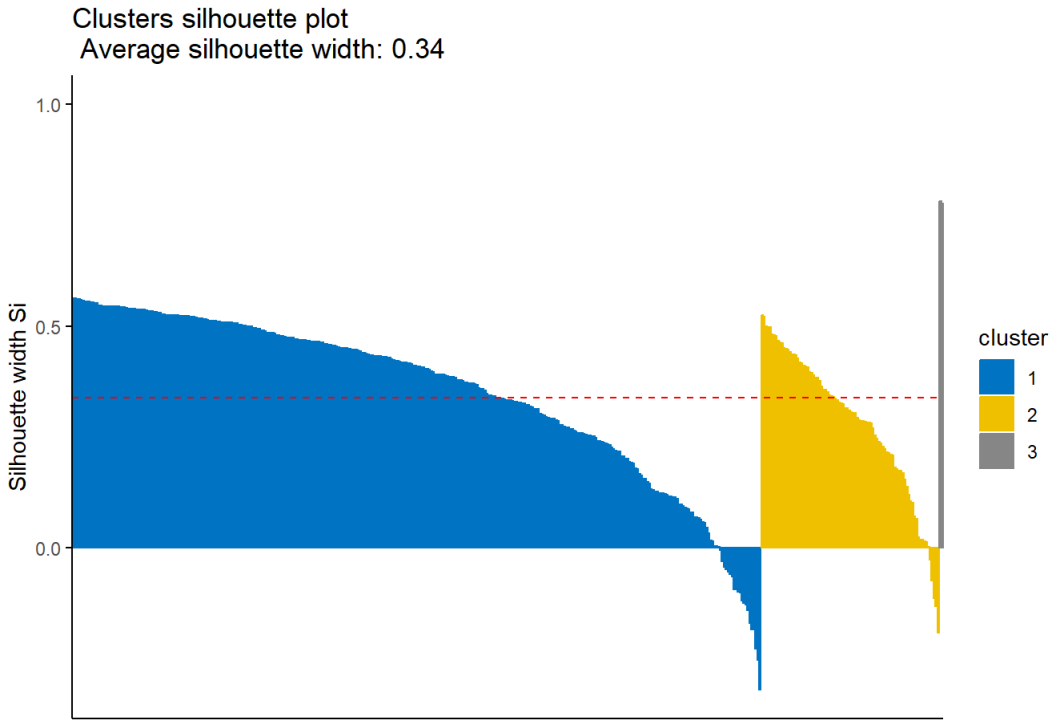

```
par(mfrow=c(1,1))
fviz_cluster(list(data=scale_df, cluster=grp.man), palette=c("orange","red","green"),
             ellipse.type="convex",repel=TRUE, show.clust.cent=FALSE, ggtheme= theme_minimal())
```

```
## Warning: ggrepel: 282 unlabeled data points
## (too many overlaps). Consider increasing
## max.overlaps
```




```
#Internal Validation
man.res<-eclust(scale_df, "hclust", k=3, hc_metric="manhattan", hc_method="average", graph=FALSE)
fviz_silhouette(man.res, palette="jco", ggtheme=theme_classic())
```

```
##   cluster size ave.sil.width
## 1       1  348          0.35
## 2       2   90          0.28
## 3       3    2          0.78
```




```
silinfo.man<-man.res$silinfo
neg_sil.man<-which(silinfo.man$widths[,"sil_width"]<0)

silinfo.man$avg.width
```

```
## [1] 0.3378243
```

```
silinfo.man$clus.avg.widths
```

```
## [1] 0.3502156 0.2801183 0.7785123
```

```
silinfo.man$widths[neg_sil.man,,drop=FALSE]
```

```
##     cluster neighbor    sil_width
## 254       1        2 -0.006184733
## 19        1        2 -0.030315016
## 326       1        3 -0.043337981
## 385       1        3 -0.049569313
## 436       1        2 -0.055141400
## 128       1        2 -0.060283763
## 208       1        2 -0.065709762
## 413       1        2 -0.093332667
## 8         1        2 -0.094866270
## 34        1        3 -0.100296155
## 1         1        2 -0.101259219
## 280       1        2 -0.120302775
## 203       1        3 -0.125157444
## 109       1        2 -0.128412733
## 188       1        2 -0.140500126
## 336       1        2 -0.170526908
## 343       1        2 -0.184383889
## 174       1        2 -0.185420189
## 13        1        2 -0.228003723
## 184       1        3 -0.253391019
## 15        1        2 -0.319919230
## 182       2        3 -0.027464050
## 166       2        3 -0.074611791
## 25        2        3 -0.114617353
## 14        2        3 -0.132531262
## 348       2        3 -0.191590037
```

```
man.link<-cluster.stats(D_man, man.res$cluster)
man.link$dunn
```

```
## [1] 0.08516489
```

```
#External validation
#confusion matrix
table(data$Channel, man.res$cluster)
```

```
##
##     1   2   3
##   1 286  11   1
##   2  62  79   1
```

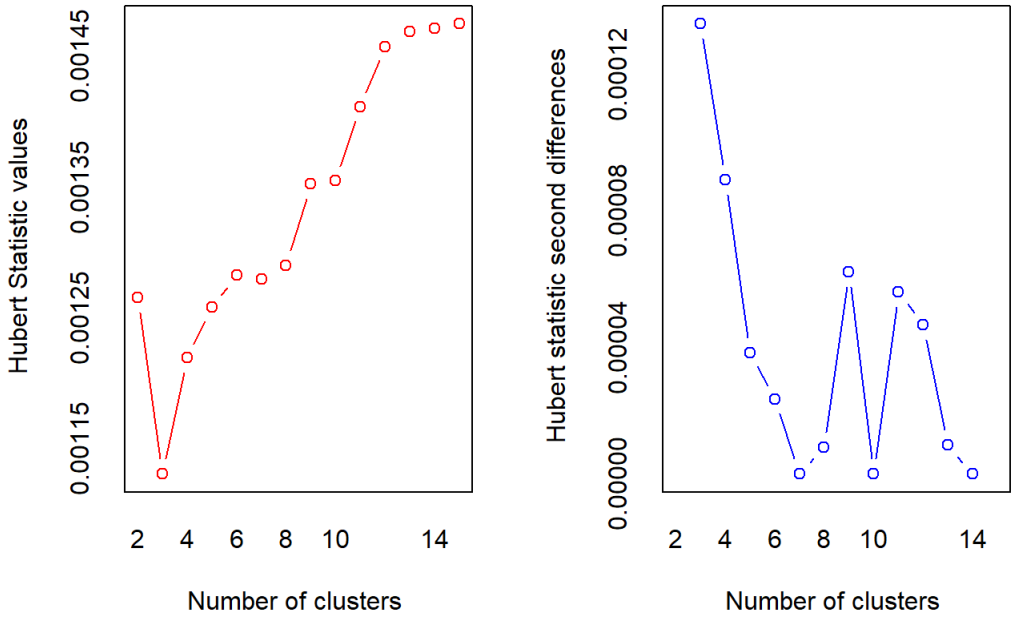
```
#rand index
tipes<-as.numeric(data$Channel)
clust_stats.man<-cluster.stats(d=D_man, tipes, man.res$cluster)
clust_stats.man$corrected.rand
```

```
## [1] 0.4150197
```

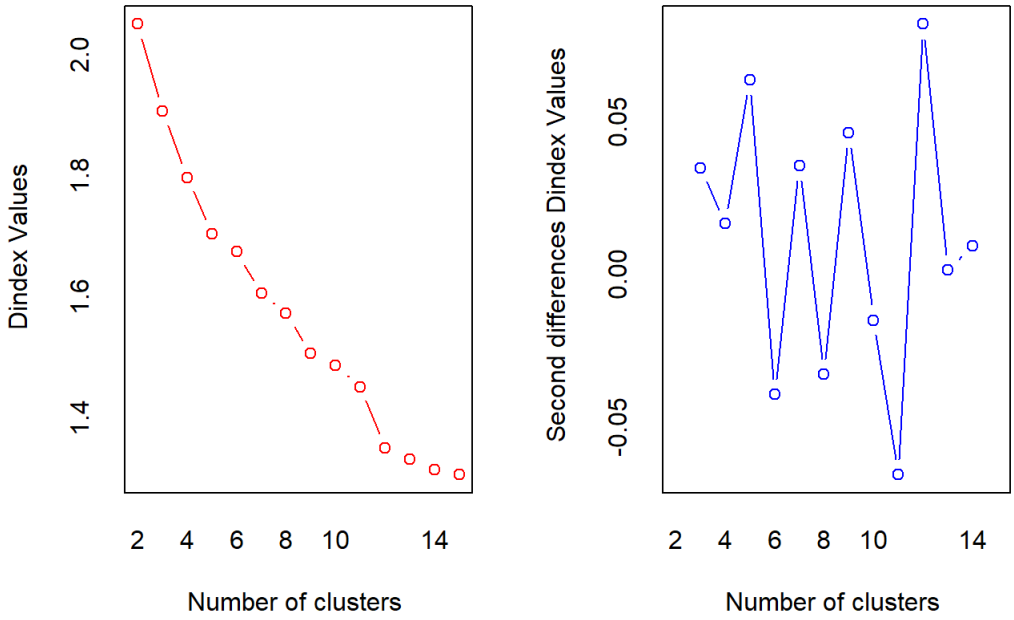
```
#The ARI provides an index that is close to 0 because it takes into account the chance of overlap.
#In addition, note that the ARI is a negative value indicating that the amount of overlap is less than expected.
##Meila's VI Index
clust_stats.man$vi
```

```
## [1] 0.8051784
```

```
#complete linkage method (Euclidean dist)-more robust
nb3<-NbClust(scale_df, distance="euclidean", method="complete")
```




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

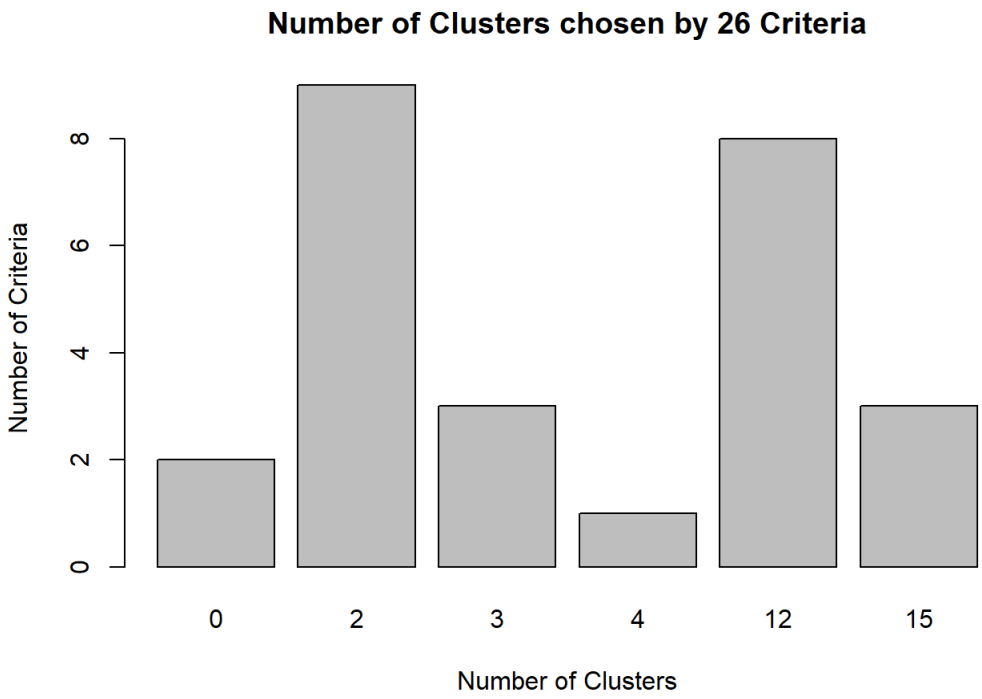



```
## *** : The D index is a graphical method of determining the number of clusters.
##               In the plot of D index, we seek a significant knee (the significant peak in Dindex
##               second differences plot) that corresponds to a significant increase of the value of
##               the measure.
##
## *******************************************************************
## * Among all indices:
## * 9 proposed 2 as the best number of clusters
## * 3 proposed 3 as the best number of clusters
## * 1 proposed 4 as the best number of clusters
## * 8 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
##
##
## *******************************************************************
```

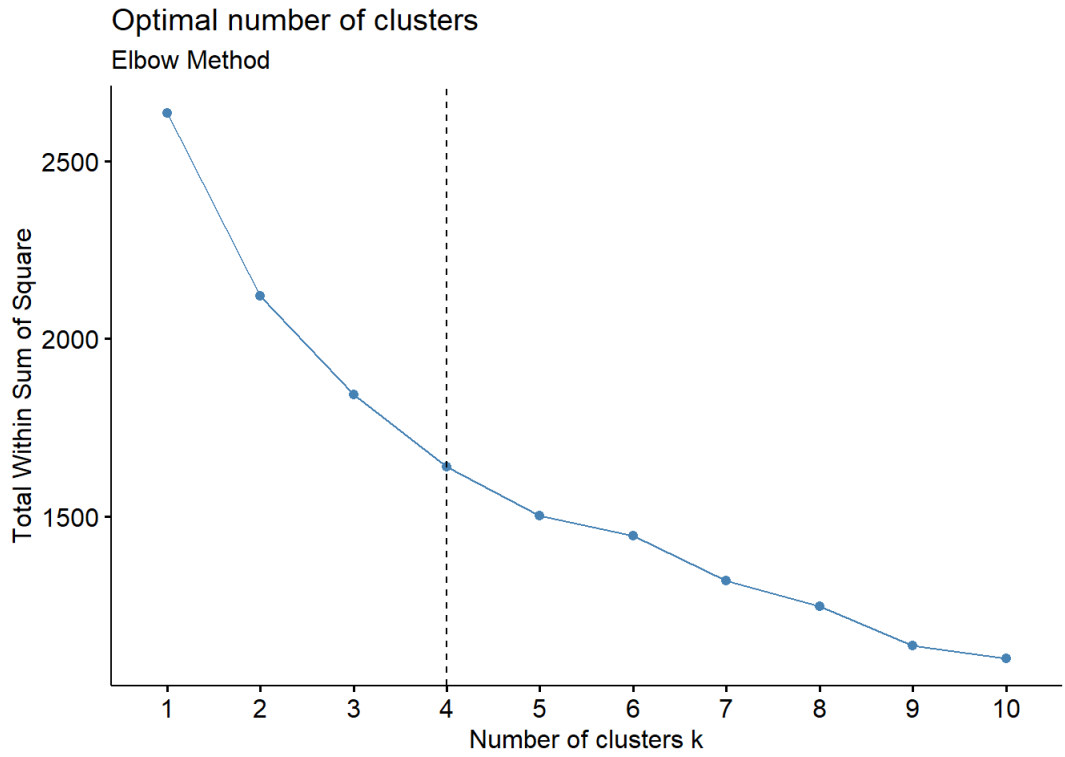
```
summary(nb3)
```

```
##                  Length Class  Mode
## All.index          364    -none- numeric
## All.CriticalValues  42    -none- numeric
## Best.nc             52    -none- numeric
## Best.partition     440    -none- numeric
```

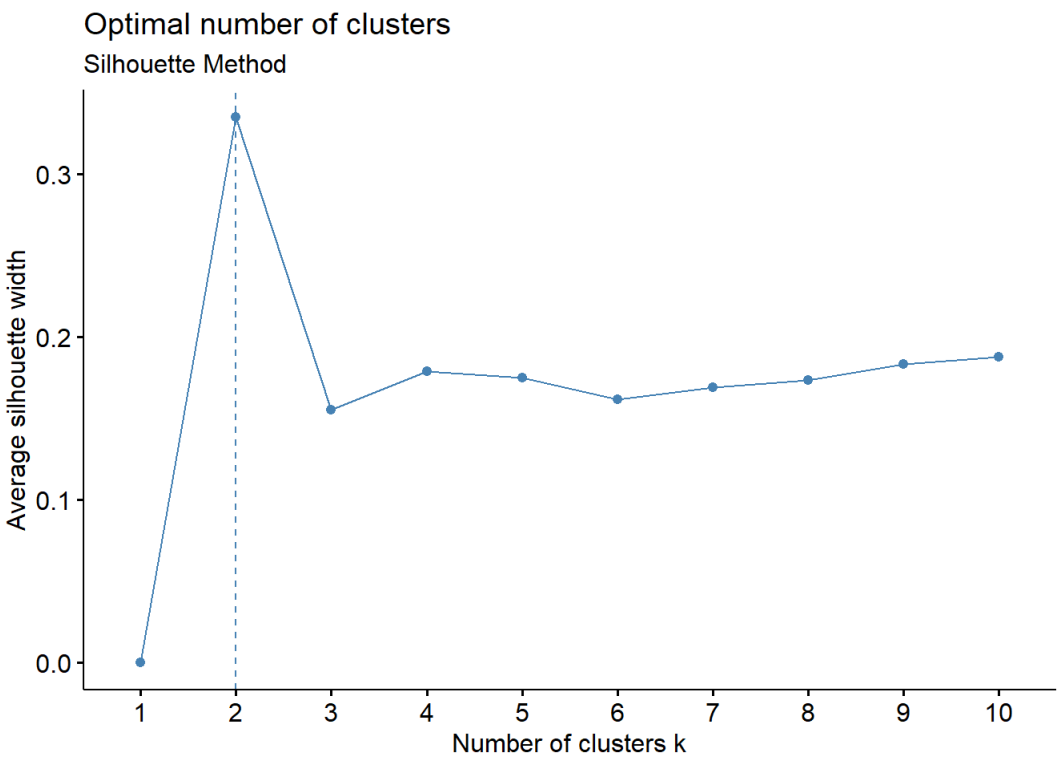
```
t3 <- table(nb3$Best.nc[1,])

par(mfrow= c(1,1))
barplot(t3,xlab="Number of Clusters",ylab="Number of Criteria",
        main="Number of Clusters chosen by 26 Criteria")
```




```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, hcut, method="wss", hc_metric="euclidean", hc_method="complete")+geom_vline(xintercept = 4, linetype=2)+labs(subtitle = "Elbow Method")
```

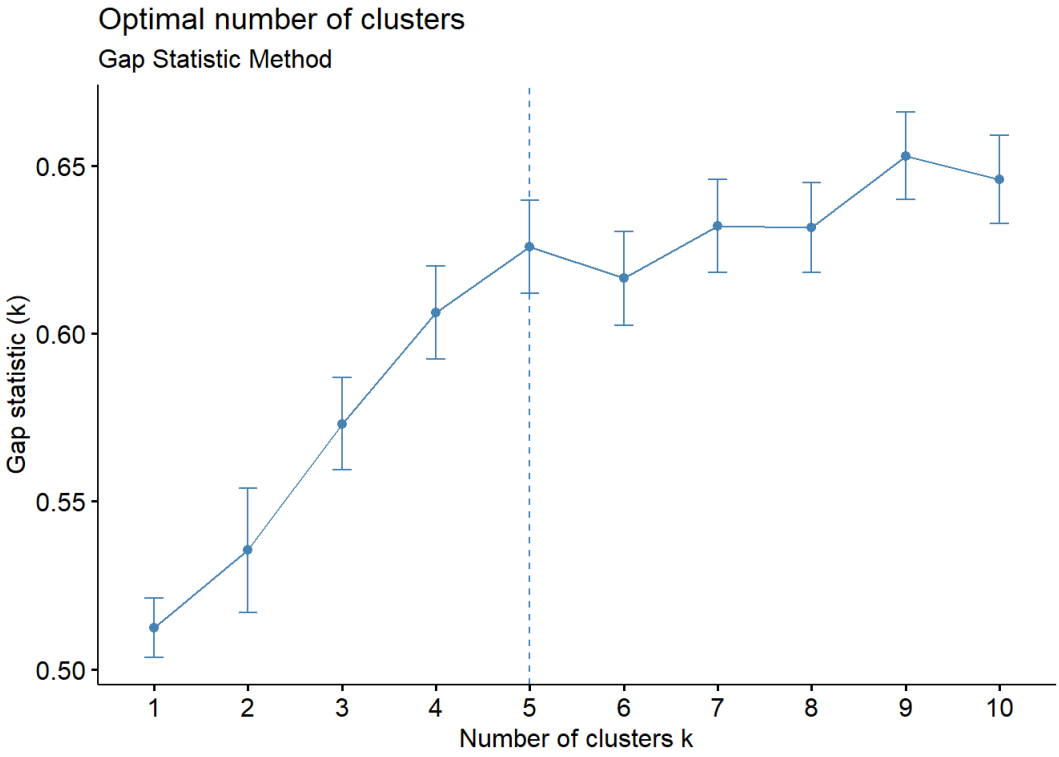



```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, hcut, method="silhouette", hc_metric="euclidean", hc_method="complete")+labs(subtitle = "Silhouette Method")
```

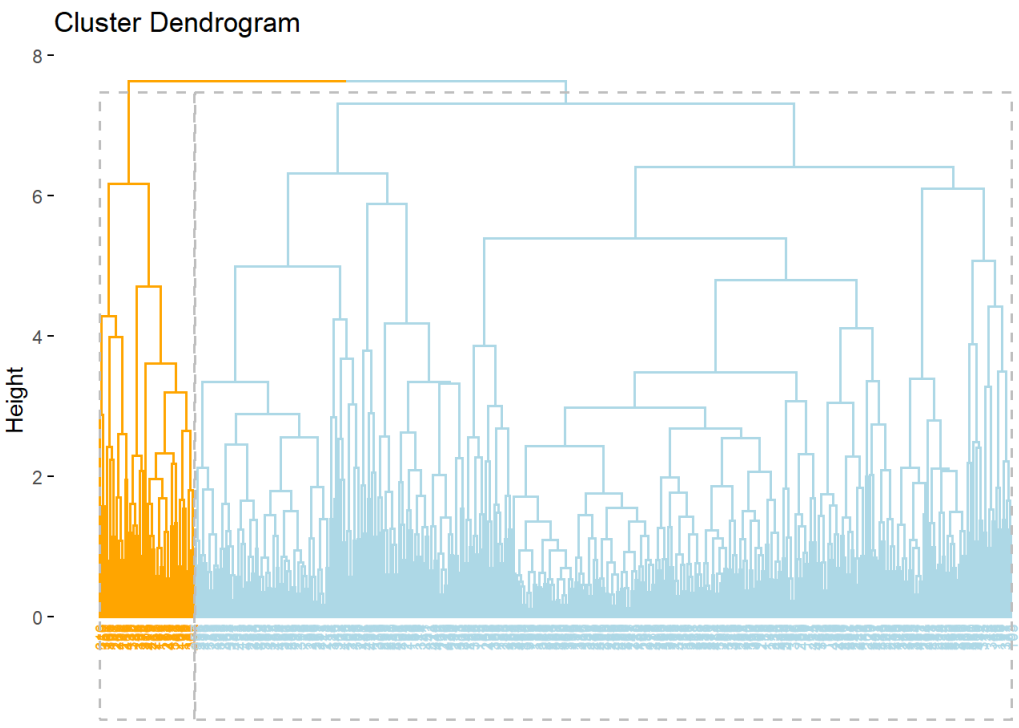



```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, hcut, method="gap_stat", nboot=500, hc_metric="euclidean", hc_method="complete")+labs(subtitle = "Gap Statistic Method")
```

```
## 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
```




```
#according with this method, 2 clusters is the most suggested choice
com.euc.hc<-hclust(d=D_euc, method="complete")

par(mfrow=c(1,1))
fviz_dend(com.euc.hc, k=2, cex=0.5,k_colors=c("orange","lightblue"),rect=TRUE)
```



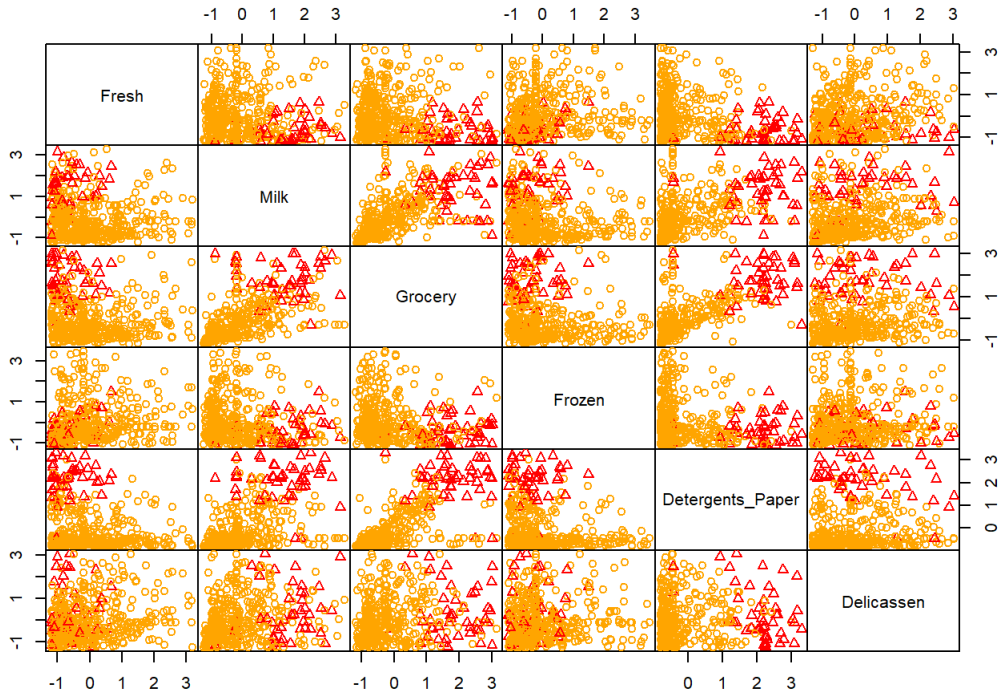

```
com.euc.coph<-cophenetic(com.euc.hc)
cor(D_euc, com.euc.coph)
```

```
## [1] 0.5841081
```

```
grp.c.euc<-cutree(com.euc.hc, k=2)
table(grp.c.euc)
```

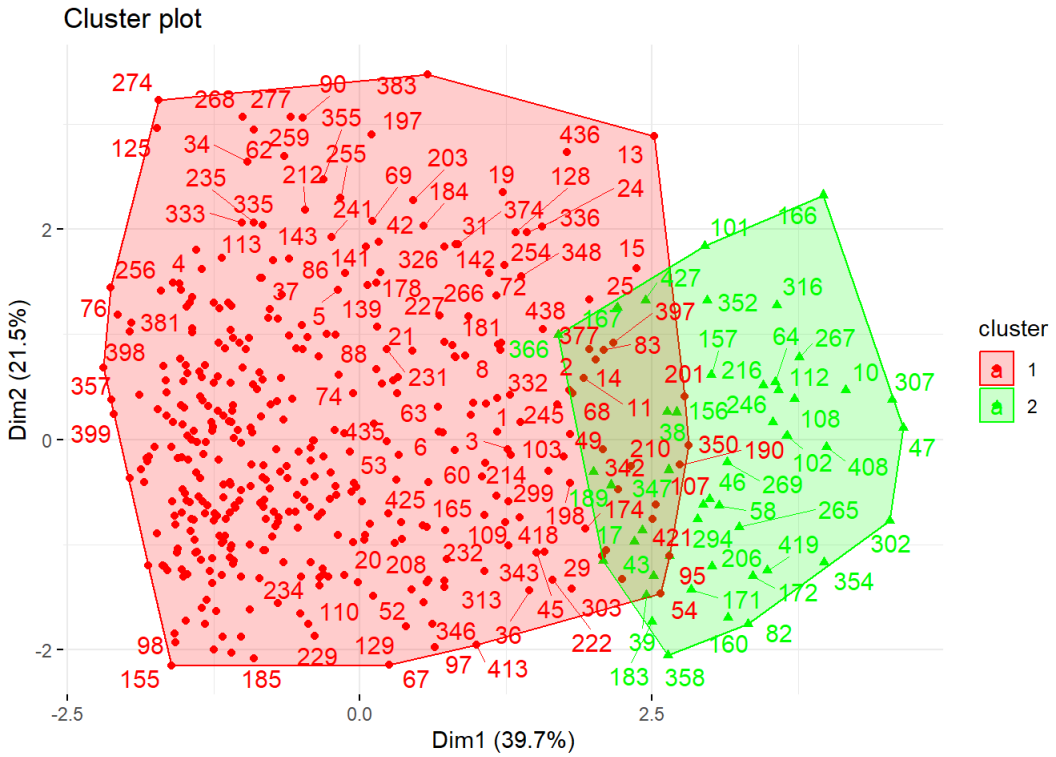
```
## grp.c.euc
##   1   2
## 394  46
```

```
par(mfrow=c(1,1))
pairs(scale_df, gap=0, pch=grp.c.euc, col=c("orange","red")[grp.c.euc])
```



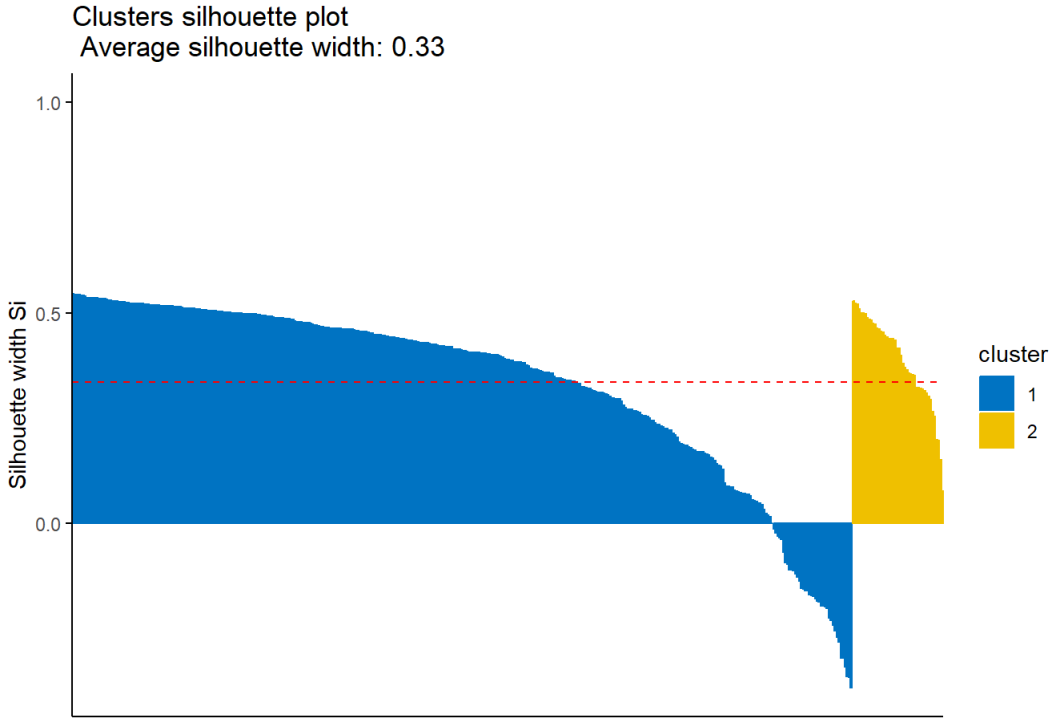

```
par(mfrow=c(1,1))
fviz_cluster(list(data=scale_df, cluster=grp.c.euc), palette=c("red","green"),
             ellipse.type="convex",repel=TRUE, show.clust.cent=FALSE, ggtheme= theme_minimal())
```

```
## Warning: ggrepel: 282 unlabeled data points
## (too many overlaps). Consider increasing
## max.overlaps
```




```
#Internal Validation
c.euc.res<-eclust(scale_df, "hclust", k=2, hc_metric="euclidean", hc_method="complete", graph=FALSE)
fviz_silhouette(c.euc.res, palette="jco", ggtheme=theme_classic())
```

```
##   cluster size ave.sil.width
## 1       1  394          0.33
## 2       2   46          0.39
```




```
silinfo.c.euc<-c.euc.res$silinfo
neg_sil.c.euc<-which(silinfo.c.euc$widths[,"sil_width"]<0)

silinfo.c.euc$avg.width
```

```
## [1] 0.3345638
```

```
silinfo.c.euc$clus.avg.widths
```

```
## [1] 0.3283461 0.3878197
```

```
silinfo.c.euc$widths[neg_sil.c.euc,,drop=FALSE]
```

```
##     cluster neighbor     sil_width
## 343       1        2 -0.0005691436
## 348       1        2 -0.0131817870
## 418       1        2 -0.0234801992
## 299       1        2 -0.0305737413
## 72        1        2 -0.0349046877
## 103       1        2 -0.0375495153
## 24        1        2 -0.0695747916
## 245       1        2 -0.0941168580
## 13        1        2 -0.0970155573
## 45        1        2 -0.1102633259
## 159       1        2 -0.1103971203
## 36        1        2 -0.1125305464
## 2         1        2 -0.1200045108
## 174       1        2 -0.1280302907
## 222       1        2 -0.1374388340
## 25        1        2 -0.1546754434
## 210       1        2 -0.1557503792
## 161       1        2 -0.1601464168
## 29        1        2 -0.1610523226
## 83        1        2 -0.1698499437
## 68        1        2 -0.1711687989
## 15        1        2 -0.1732654035
## 397       1        2 -0.1785862468
## 377       1        2 -0.1840997215
## 49        1        2 -0.1876735858
## 198       1        2 -0.1955714973
## 201       1        2 -0.1956990549
## 350       1        2 -0.1980684245
## 306       1        2 -0.2029047013
## 14        1        2 -0.2243090190
## 11        1        2 -0.2312670581
## 303       1        2 -0.2428806255
## 342       1        2 -0.2552369952
## 176       1        2 -0.2701620375
## 341       1        2 -0.2828795113
## 95        1        2 -0.3197487377
## 190       1        2 -0.3202164252
## 215       1        2 -0.3403727339
## 54        1        2 -0.3633124209
## 107       1        2 -0.3651517055
## 421       1        2 -0.3903060013
```

```
c.euc.link<-cluster.stats(D_euc, c.euc.res$cluster)
c.euc.link$dunn
```

```
## [1] 0.1100385
```

```
#External validation
#confusion matrix
table(data$Channel, c.euc.res$cluster)
```

```
##
##       1   2
##   1 294   4
##   2 100  42
```

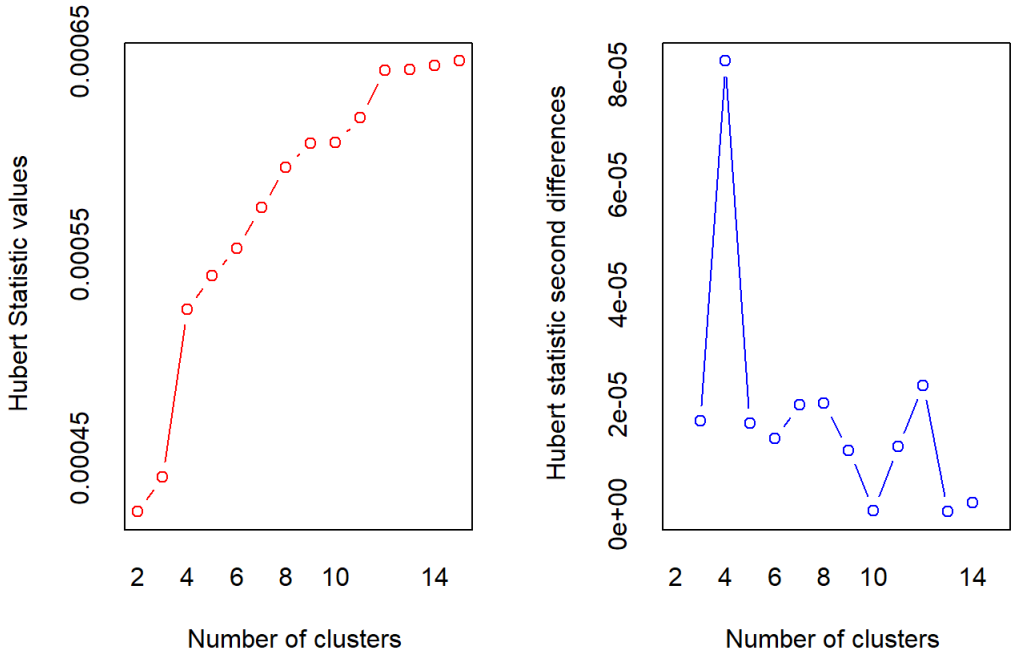
```
#rand index
tipes<-as.numeric(data$Channel)
clust_stats.c.euc<-cluster.stats(d=D_euc, tipes, c.euc.res$cluster)
clust_stats.c.euc$corrected.rand
```

```
## [1] 0.2157661
```

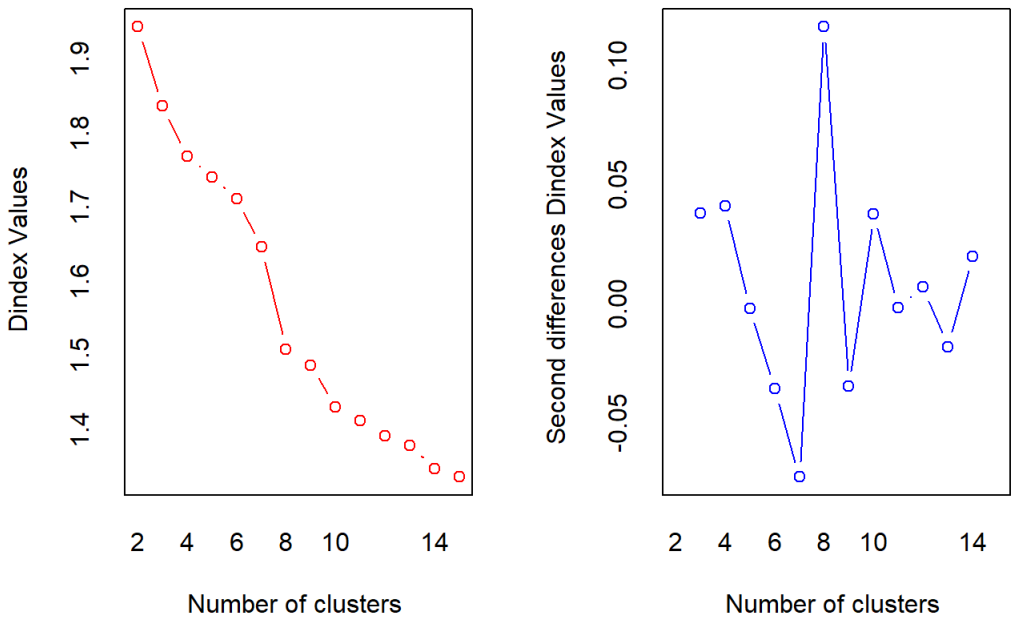
```
#The ARI provides an index that is close to 0 because it takes into account the chance of overlap.
#In addition, note that the ARI is a negative value indicating that the amount of overlap is less than expected.
##Meila's VI Index
clust_stats.c.euc$vi
```

```
## [1] 0.7823353
```

```
#complete linkage method (Manhattan dist)-more robust
nb4<-NbClust(scale_df, distance="manhattan", method="complete")
```




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

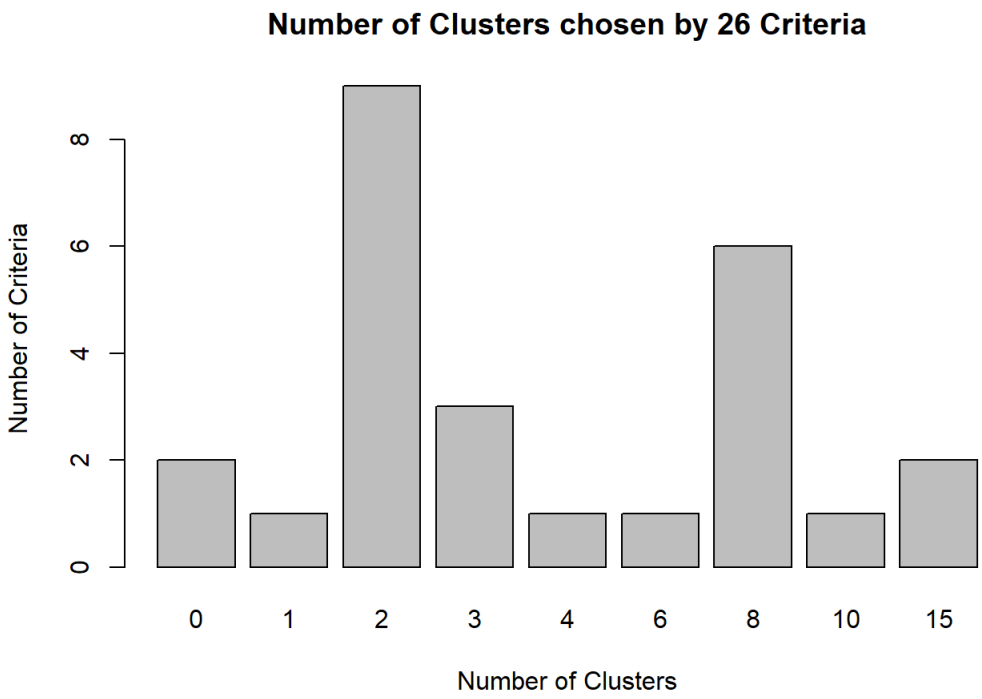


```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 9 proposed 2 as the best number of clusters
## * 3 proposed 3 as the best number of clusters
## * 1 proposed 4 as the best number of clusters
## * 1 proposed 6 as the best number of clusters
## * 6 proposed 8 as the best number of clusters
## * 1 proposed 10 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 is  2
##
##
## *******************************************************************
```

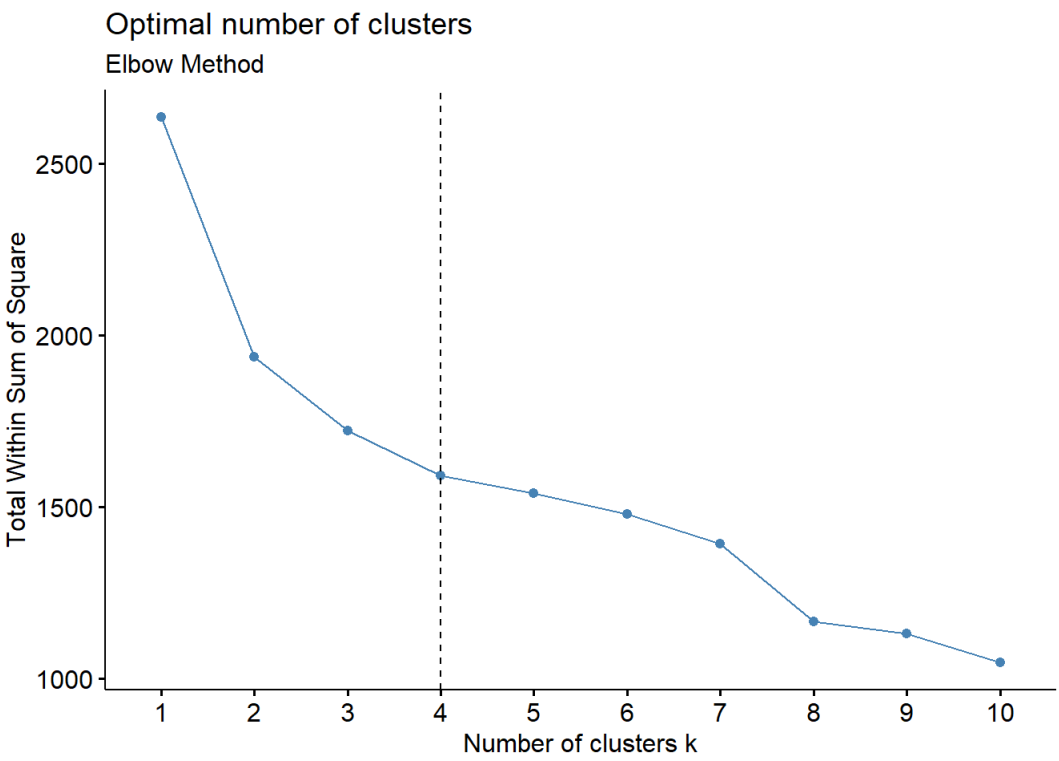
```
summary(nb4)
```

```
##                  Length Class  Mode
## All.index          364    -none- numeric
## All.CriticalValues  42    -none- numeric
## Best.nc             52    -none- numeric
## Best.partition     440    -none- numeric
```

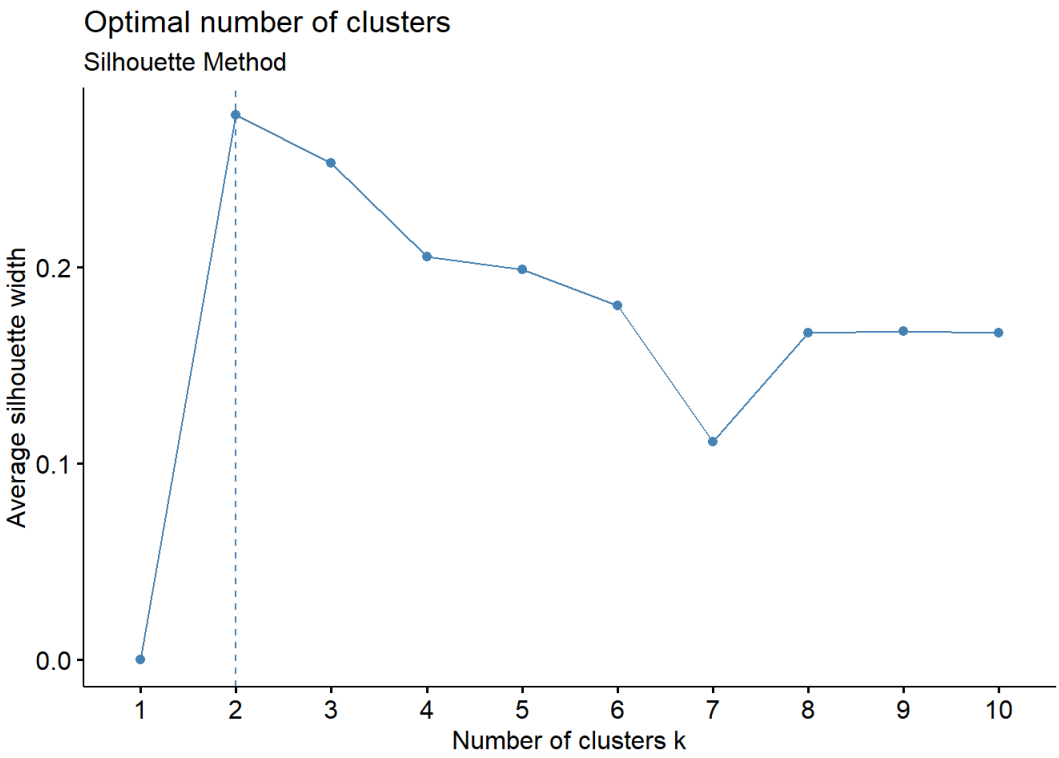
```
t4 <- table(nb4$Best.nc[1,])

par(mfrow= c(1,1))
barplot(t4,xlab="Number of Clusters",ylab="Number of Criteria",
        main="Number of Clusters chosen by 26 Criteria")
```




```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, hcut, method="wss", hc_metric="manhattan", hc_method="complete")+geom_vline(xintercept = 4, linetype=2)+labs(subtitle = "Elbow Method")
```

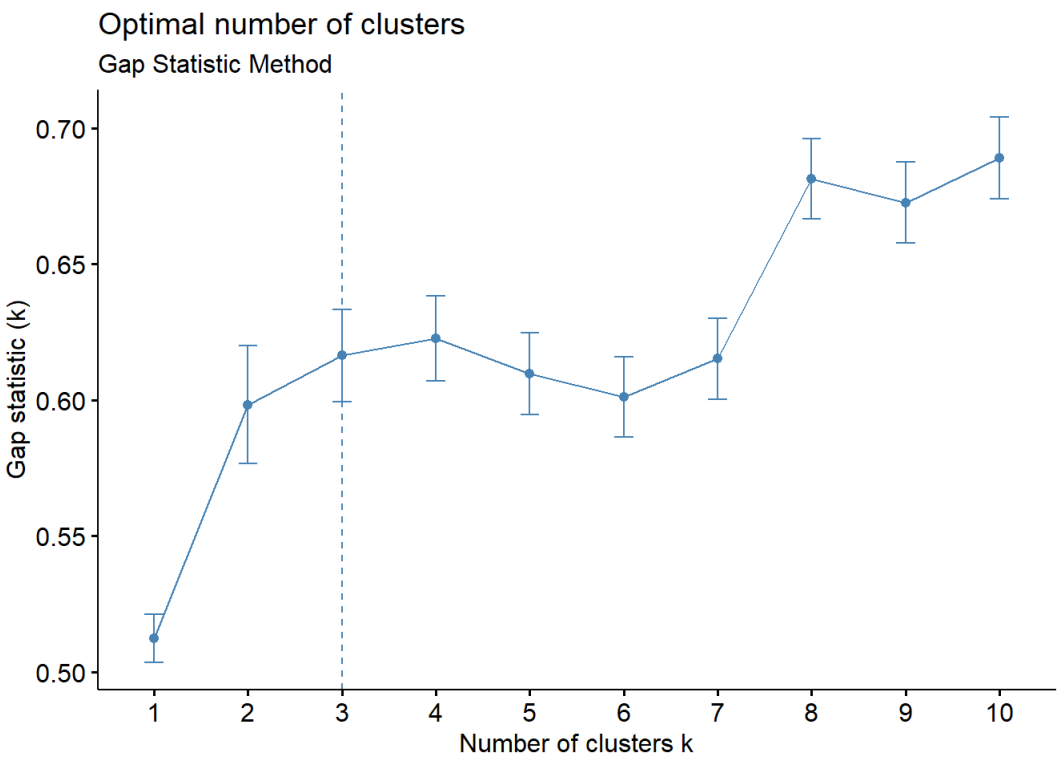



```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, hcut, method="silhouette", hc_metric="manhatta", hc_method="complete")+labs(subtitle = "Silhouette Method")
```

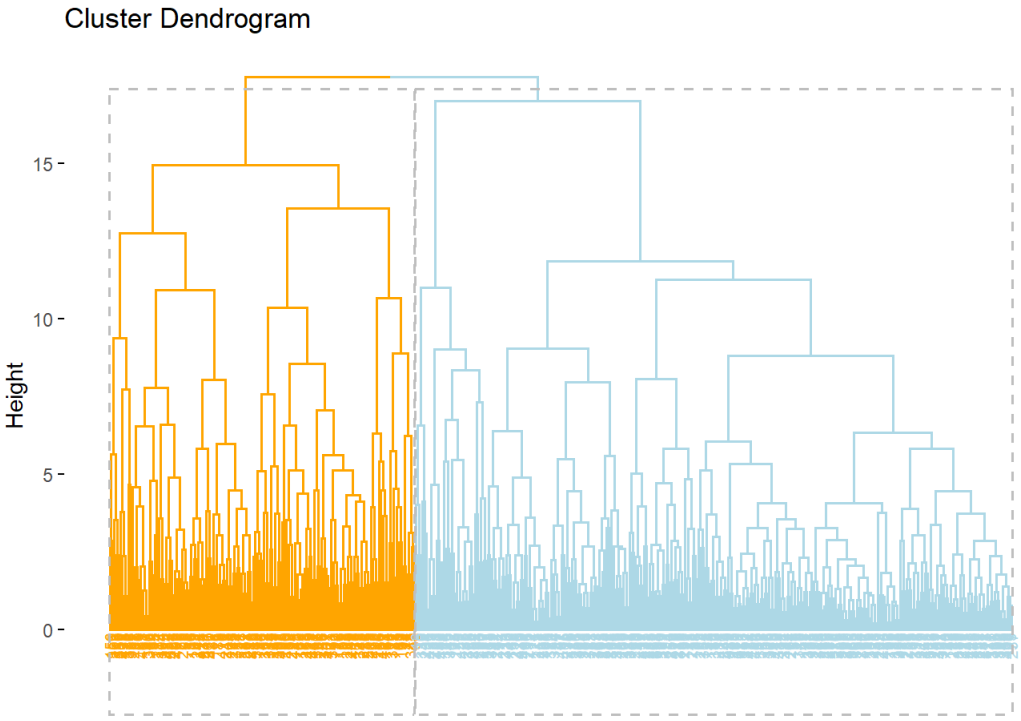



```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, hcut, method="gap_stat", nboot=500, hc_metric="manhattan", hc_method="complete")+labs(subtitle = "Gap Statistic Method")
```

```
## 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
```




```
#according with this method, 2 clusters is the most suggested choice
com.man.hc<-hclust(d=D_man, method="complete")

par(mfrow=c(1,1))
fviz_dend(com.man.hc, k=2, cex=0.5,k_colors=c("orange","lightblue"),rect=TRUE)
```



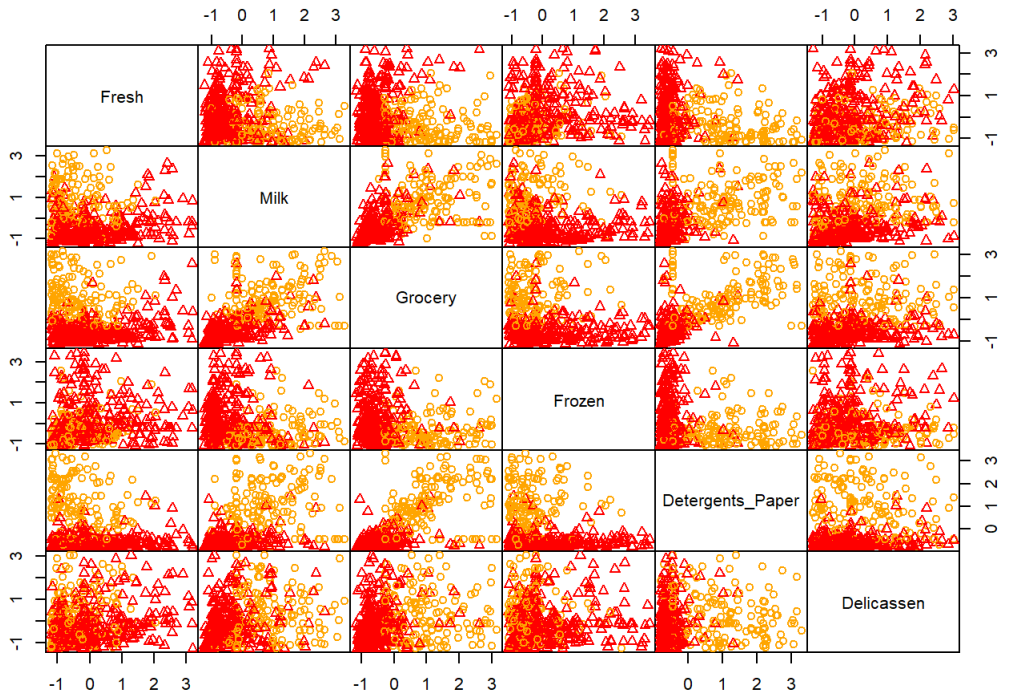

```
com.man.coph<-cophenetic(com.man.hc)
cor(D_man, com.man.coph)
```

```
## [1] 0.6171262
```

```
grp.c.man1<-cutree(com.man.hc, k=2)
table(grp.c.man1)
```

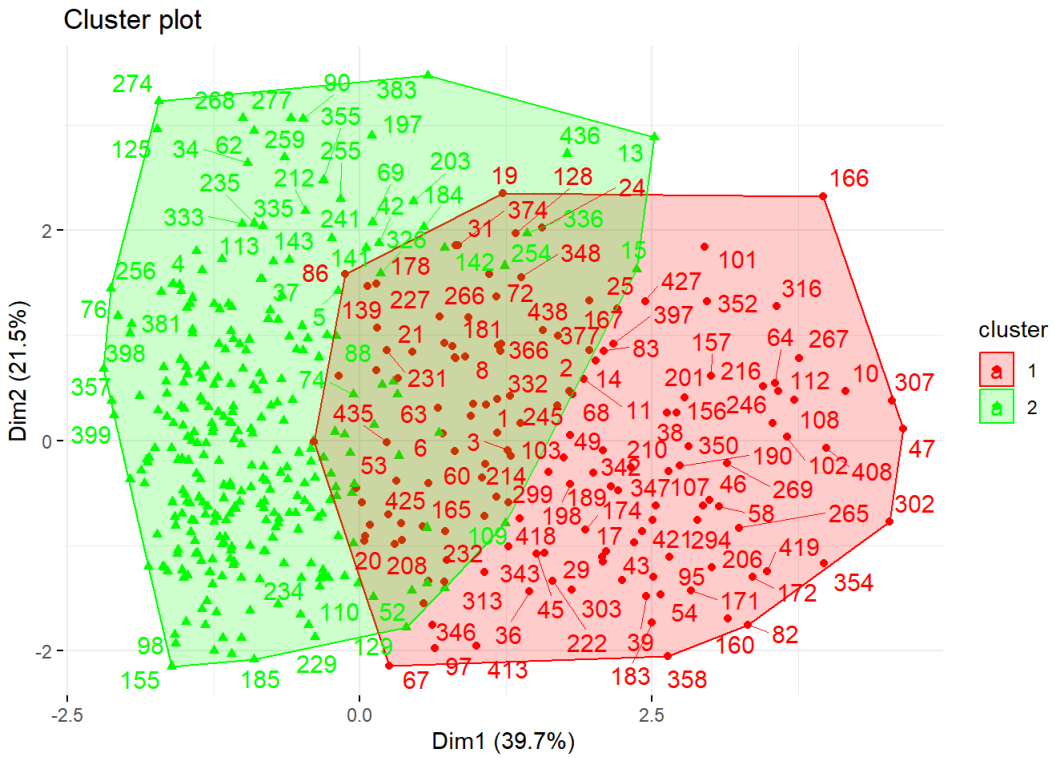
```
## grp.c.man1
##   1   2
## 149 291
```

```
par(mfrow=c(1,1))
pairs(scale_df, gap=0, pch=grp.c.man1, col=c("orange","red")[grp.c.man1])
```



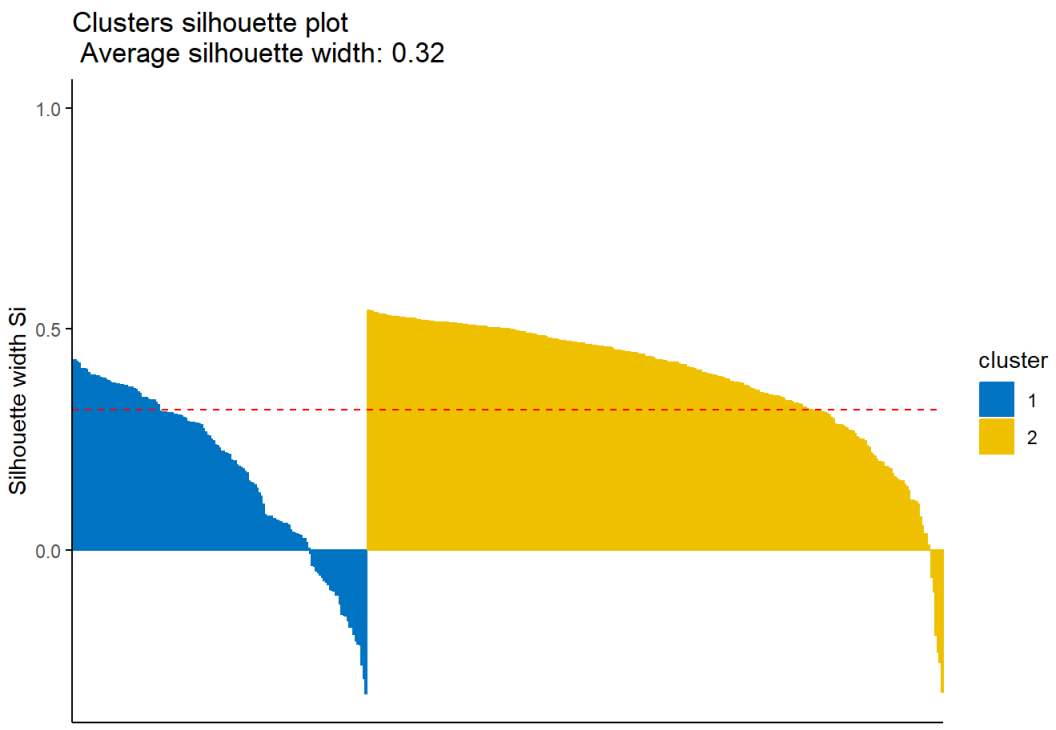

```
par(mfrow=c(1,1))
fviz_cluster(list(data=scale_df, cluster=grp.c.man1), palette=c("red","green"),
             ellipse.type="convex",repel=TRUE, show.clust.cent=FALSE, ggtheme= theme_minimal())
```

```
## Warning: ggrepel: 282 unlabeled data points
## (too many overlaps). Consider increasing
## max.overlaps
```




```
#Internal Validation
c.man.res<-eclust(scale_df, "hclust", k=2, hc_metric="manhattan", hc_method="complete", graph=FALSE)
fviz_silhouette(c.man.res, palette="jco", ggtheme=theme_classic())
```

```
##   cluster size ave.sil.width
## 1       1  149         0.18
## 2       2  291         0.39
```




```
silinfo.c.man<-c.man.res$silinfo
neg_sil.c.man<-which(silinfo.c.man$widths[,"sil_width"]<0)

silinfo.c.man$avg.width
```

```
## [1] 0.3176102
```

```
silinfo.c.man$clus.avg.widths
```

```
## [1] 0.1795695 0.3882908
```

```
silinfo.c.man$widths[neg_sil.c.man,,drop=FALSE]
```

```
##     cluster neighbor     sil_width
## 6         1        2 -0.0088772745
## 438       1        2 -0.0357764347
## 304       1        2 -0.0374413023
## 67        1        2 -0.0482444758
## 298       1        2 -0.0523013846
## 209       1        2 -0.0575088363
## 104       1        2 -0.0626539684
## 20        1        2 -0.0699349777
## 202       1        2 -0.0741987530
## 232       1        2 -0.0795841723
## 332       1        2 -0.0889746337
## 53        1        2 -0.0911125987
## 7         1        2 -0.0937800099
## 435       1        2 -0.1018357704
## 78        1        2 -0.1023490645
## 414       1        2 -0.1216558317
## 85        1        2 -0.1456152964
## 21        1        2 -0.1473109563
## 409       1        2 -0.1502650327
## 231       1        2 -0.1600684358
## 244       1        2 -0.1744568858
## 324       1        2 -0.1756180652
## 178       1        2 -0.1909196091
## 9         1        2 -0.2052516169
## 56        1        2 -0.2135872592
## 139       1        2 -0.2156448423
## 86        1        2 -0.2597102110
## 371       1        2 -0.2909452235
## 424       1        2 -0.3256227192
## 383       2        1 -0.0008914533
## 254       2        1 -0.0616848329
## 436       2        1 -0.0952620146
## 109       2        1 -0.1935108200
## 336       2        1 -0.2309306460
## 13        2        1 -0.2557553117
## 15        2        1 -0.3224851930
```

```
c.man.link<-cluster.stats(D_man, c.man.res$cluster)
c.man.link$dunn
```

```
## [1] 0.06518729
```

```
#External validation
#confusion matrix
table(data$Channel, c.man.res$cluster)
```

```
##
##      1   2
##   1  35 263
##   2 114  28
```

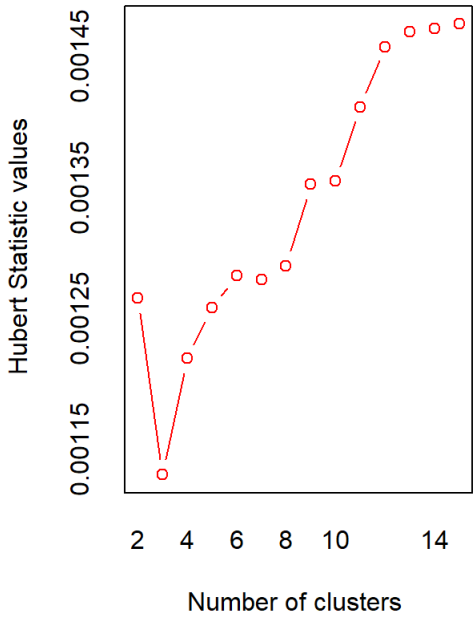
```
#rand index
tipes<-as.numeric(data$Channel)
clust_stats.c.man<-cluster.stats(d=D_man, tipes, c.man.res$cluster)
clust_stats.c.man$corrected.rand
```

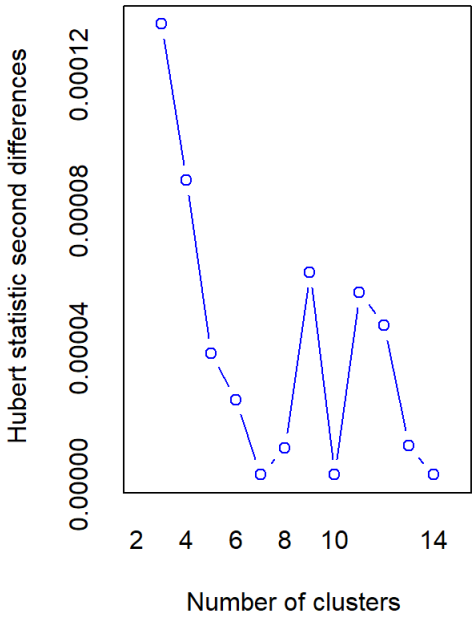
```
## [1] 0.5018664
```

```
#The ARI provides an index that is close to 0 because it takes into account the chance of overlap.
#In addition, note that the ARI is a negative value indicating that the amount of overlap is less than expected.
##Meila's VI Index
clust_stats.c.man$vi
```

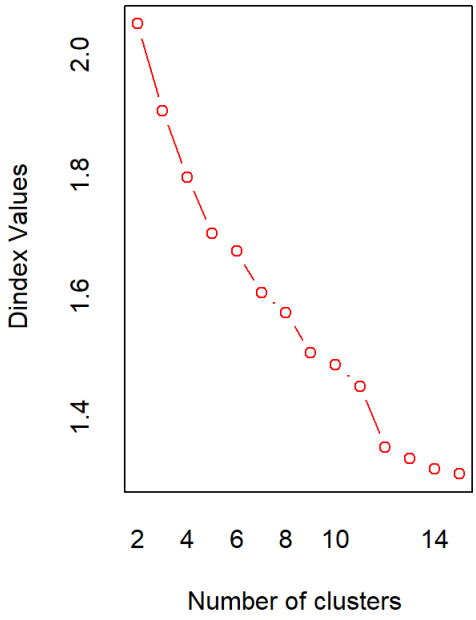
```
## [1] 0.7993234
```

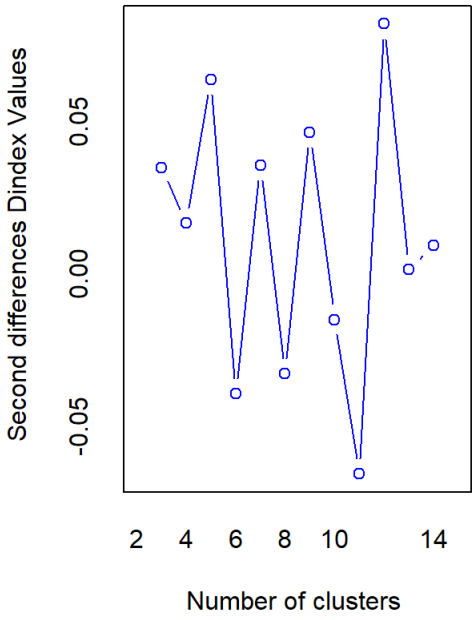
```
#complete linkage method (Minkowski dist)-more robust
nb5<-NbClust(scale_df, distance="minkowski", method="complete")
```






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



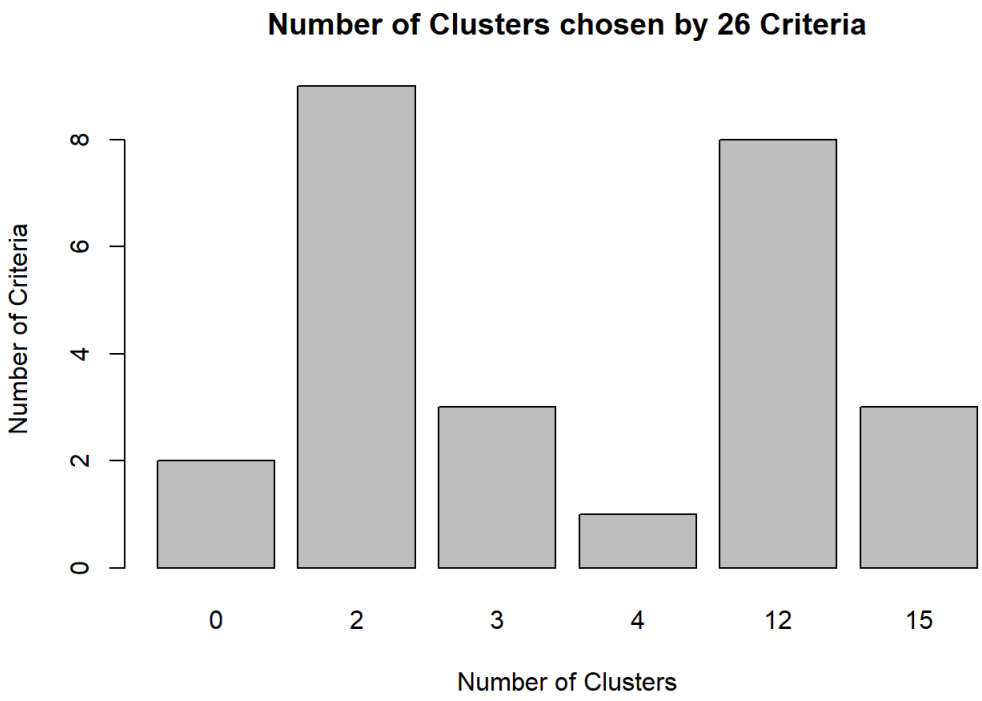


```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 9 proposed 2 as the best number of clusters
## * 3 proposed 3 as the best number of clusters
## * 1 proposed 4 as the best number of clusters
## * 8 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
##
##
## *******************************************************************
```

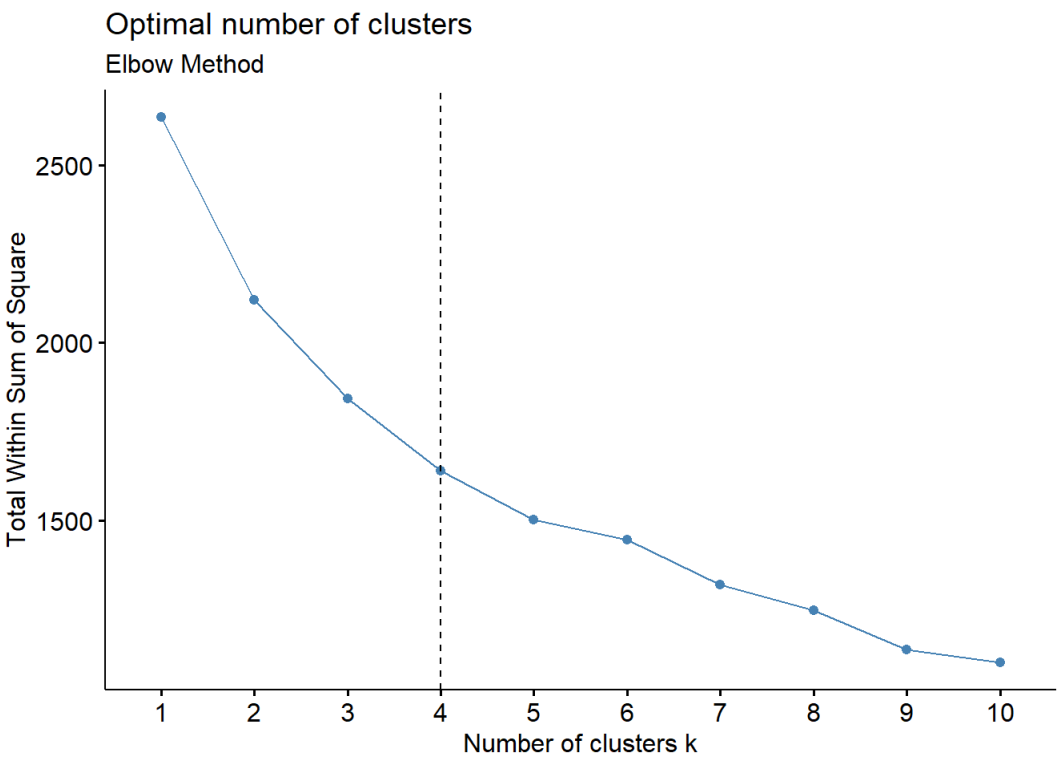
```
summary(nb5)
```

```
##                  Length Class  Mode
## All.index          364    -none- numeric
## All.CriticalValues  42    -none- numeric
## Best.nc             52    -none- numeric
## Best.partition     440    -none- numeric
```

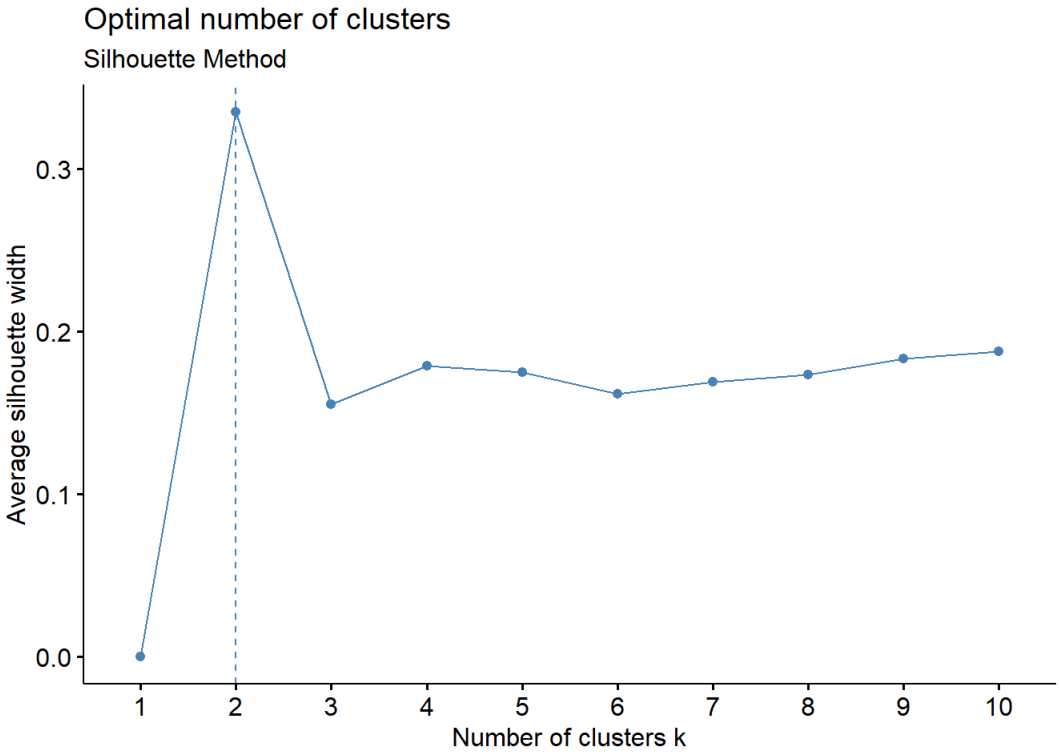
```
t5 <- table(nb5$Best.nc[1,])

par(mfrow= c(1,1))
barplot(t5,xlab="Number of Clusters",ylab="Number of Criteria",
        main="Number of Clusters chosen by 26 Criteria")
```




```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, hcut, method="wss", hc_metric="minkowski", hc_method="complete")+geom_vline(xintercept = 4, linetype=2)+labs(subtitle = "Elbow Method")
```

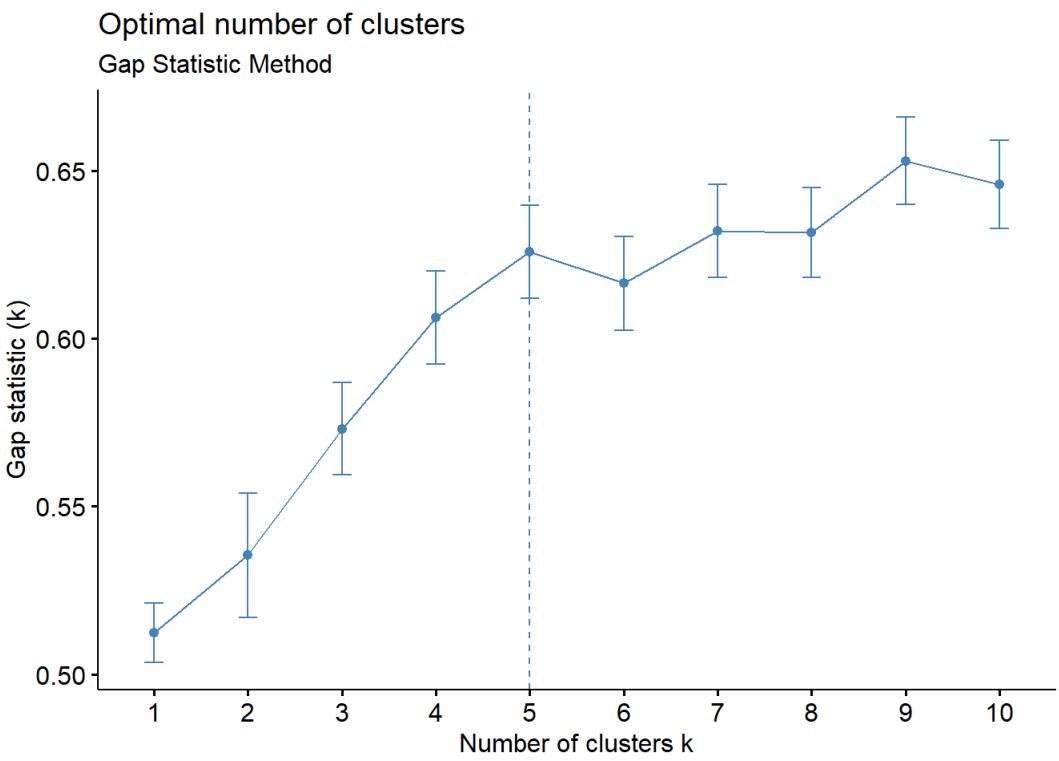



```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, hcut, method="silhouette", hc_metric="minkowski", hc_method="complete")+labs(subtitle = "Silhouette Method")
```

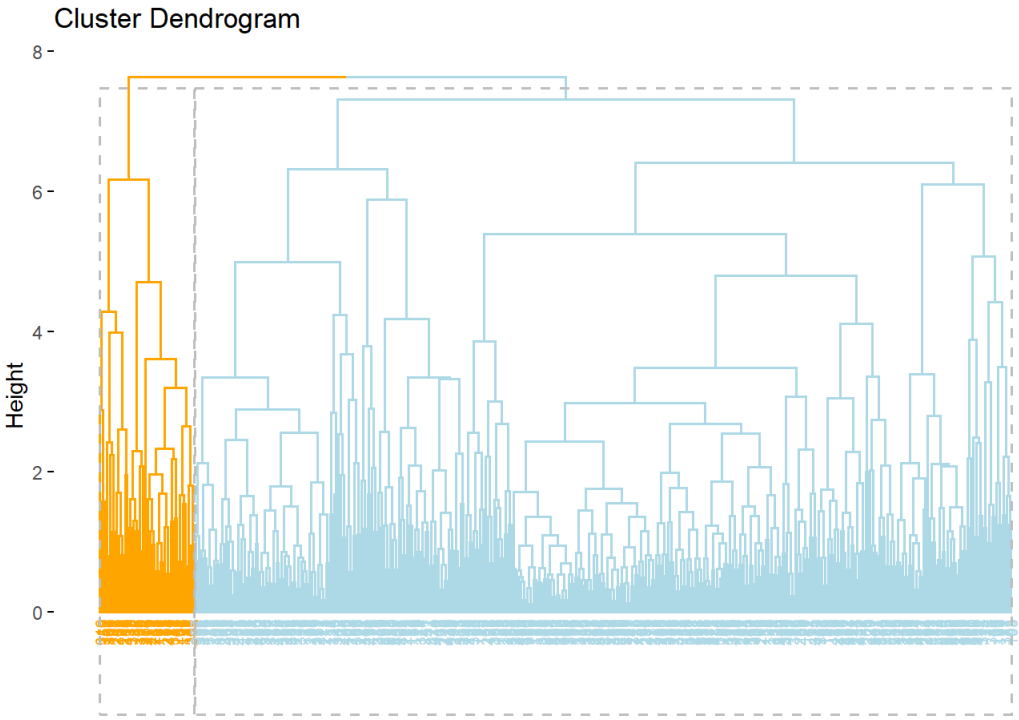



```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, hcut, method="gap_stat", nboot=500, hc_metric="minkowski", hc_method="complete")+labs(subtitle = "Gap Statistic Method")
```

```
## 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
```




```
#according with this method, 2 clusters is the most suggested choice
com.min.hc<-hclust(d=D_min, method="complete")

par(mfrow=c(1,1))
fviz_dend(com.min.hc, k=2, cex=0.5,k_colors=c("orange","lightblue"),rect=TRUE)
```



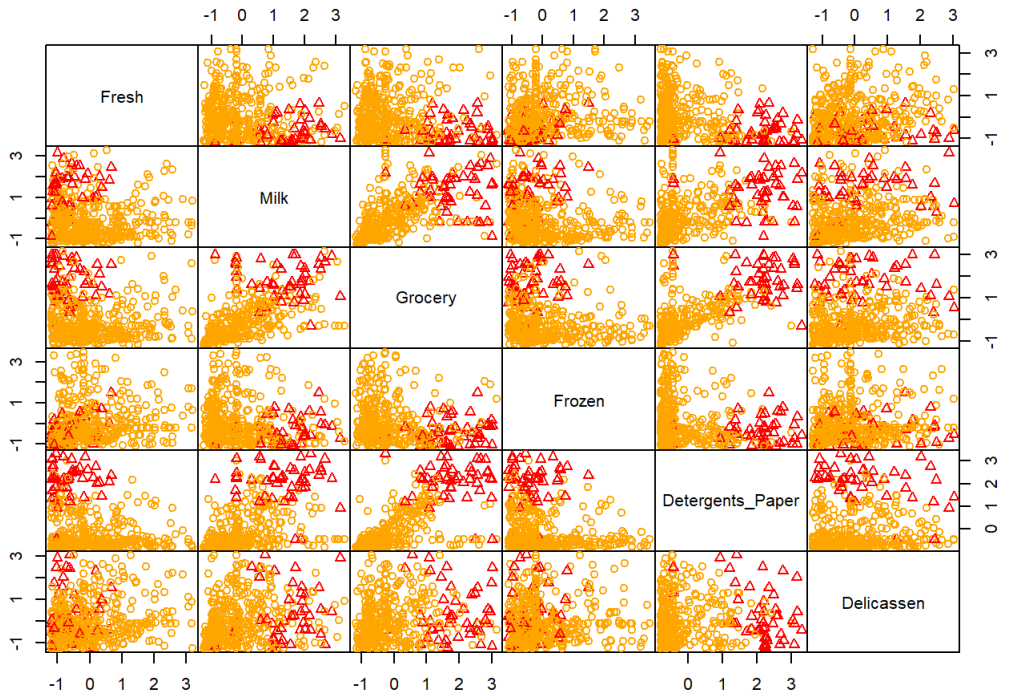

```
com.min.coph<-cophenetic(com.min.hc)
cor(D_min, com.min.coph)
```

```
## [1] 0.5841081
```

```
grp.c.min<-cutree(com.min.hc, k=2)
table(grp.c.min)
```

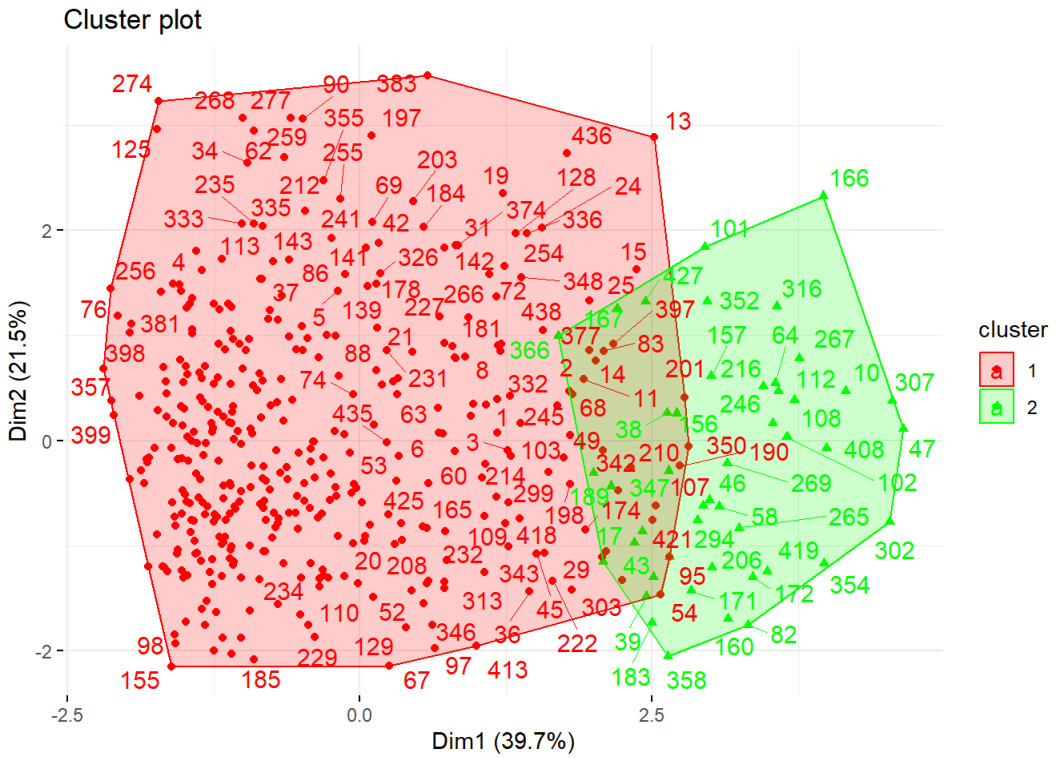
```
## grp.c.min
##   1   2
## 394  46
```

```
par(mfrow=c(1,1))
pairs(scale_df, gap=0, pch=grp.c.min, col=c("orange","red")[grp.c.min])
```



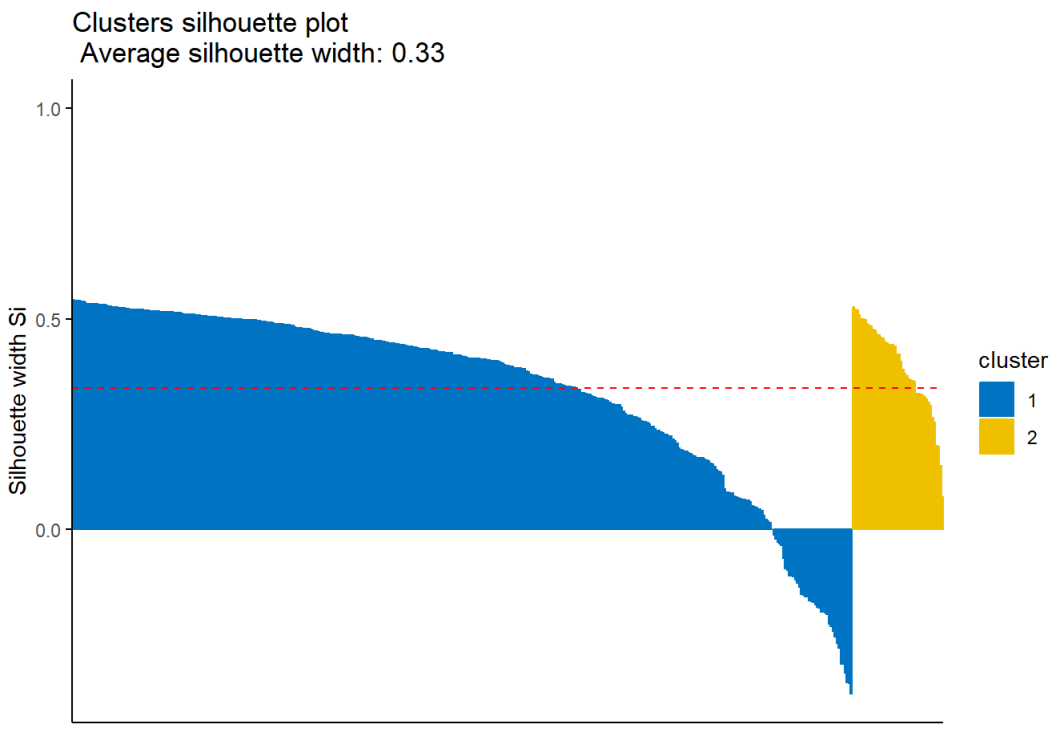

```
par(mfrow=c(1,1))
fviz_cluster(list(data=scale_df, cluster=grp.c.min), palette=c("red","green"),
             ellipse.type="convex",repel=TRUE, show.clust.cent=FALSE, ggtheme= theme_minimal())
```

```
## Warning: ggrepel: 282 unlabeled data points
## (too many overlaps). Consider increasing
## max.overlaps
```




```
#Internal Validation
c.min.res<-eclust(scale_df, "hclust", k=2, hc_metric="minkowski", hc_method="complete", graph=FALSE)
fviz_silhouette(c.min.res, palette="jco", ggtheme=theme_classic())
```

```
##   cluster size ave.sil.width
## 1       1  394          0.33
## 2       2   46          0.39
```




```
silinfo.c.min<-c.min.res$silinfo
neg_sil.c.min<-which(silinfo.c.min$widths[,"sil_width"]<0)

silinfo.c.min$avg.width
```

```
## [1] 0.3345638
```

```
silinfo.c.min$clus.avg.widths
```

```
## [1] 0.3283461 0.3878197
```

```
silinfo.c.min$widths[neg_sil.c.min,,drop=FALSE]
```

```
##     cluster neighbor     sil_width
## 343       1        2 -0.0005691436
## 348       1        2 -0.0131817870
## 418       1        2 -0.0234801992
## 299       1        2 -0.0305737413
## 72        1        2 -0.0349046877
## 103       1        2 -0.0375495153
## 24        1        2 -0.0695747916
## 245       1        2 -0.0941168580
## 13        1        2 -0.0970155573
## 45        1        2 -0.1102633259
## 159       1        2 -0.1103971203
## 36        1        2 -0.1125305464
## 2         1        2 -0.1200045108
## 174       1        2 -0.1280302907
## 222       1        2 -0.1374388340
## 25        1        2 -0.1546754434
## 210       1        2 -0.1557503792
## 161       1        2 -0.1601464168
## 29        1        2 -0.1610523226
## 83        1        2 -0.1698499437
## 68        1        2 -0.1711687989
## 15        1        2 -0.1732654035
## 397       1        2 -0.1785862468
## 377       1        2 -0.1840997215
## 49        1        2 -0.1876735858
## 198       1        2 -0.1955714973
## 201       1        2 -0.1956990549
## 350       1        2 -0.1980684245
## 306       1        2 -0.2029047013
## 14        1        2 -0.2243090190
## 11        1        2 -0.2312670581
## 303       1        2 -0.2428806255
## 342       1        2 -0.2552369952
## 176       1        2 -0.2701620375
## 341       1        2 -0.2828795113
## 95        1        2 -0.3197487377
## 190       1        2 -0.3202164252
## 215       1        2 -0.3403727339
## 54        1        2 -0.3633124209
## 107       1        2 -0.3651517055
## 421       1        2 -0.3903060013
```

```
c.min.link<-cluster.stats(D_min, c.min.res$cluster)
c.min.link$dunn
```

```
## [1] 0.1100385
```

```
#External validation
#confusion matrix
table(data$Channel, c.min.res$cluster)
```

```
##
##      1   2
##   1 294   4
##   2 100  42
```

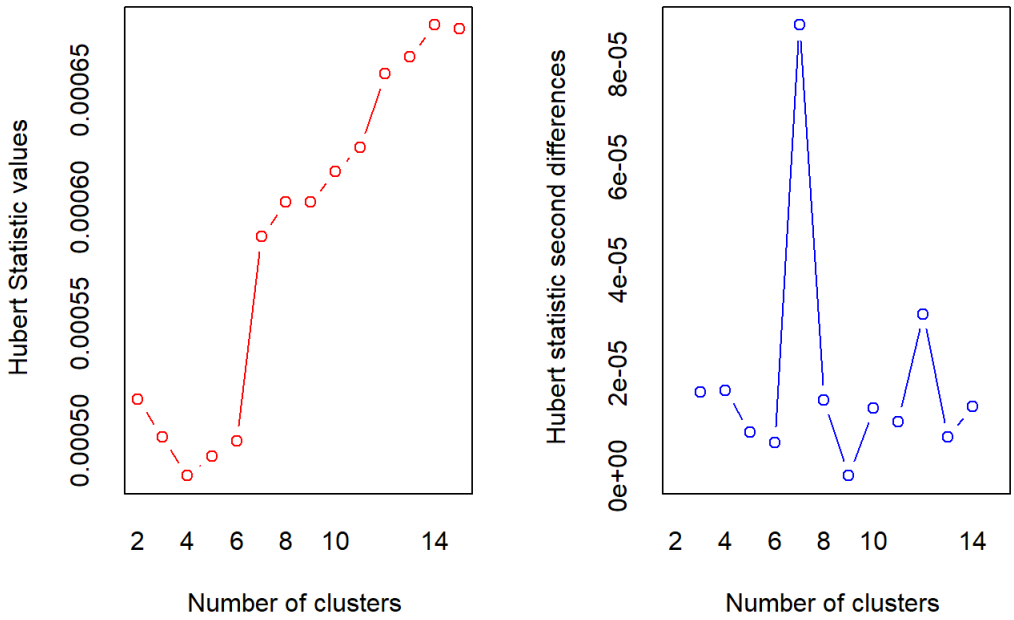
```
#rand index
tipes<-as.numeric(data$Channel)
clust_stats.c.min<-cluster.stats(d=D_min, tipes, c.min.res$cluster)
clust_stats.c.min$corrected.rand
```

```
## [1] 0.2157661
```

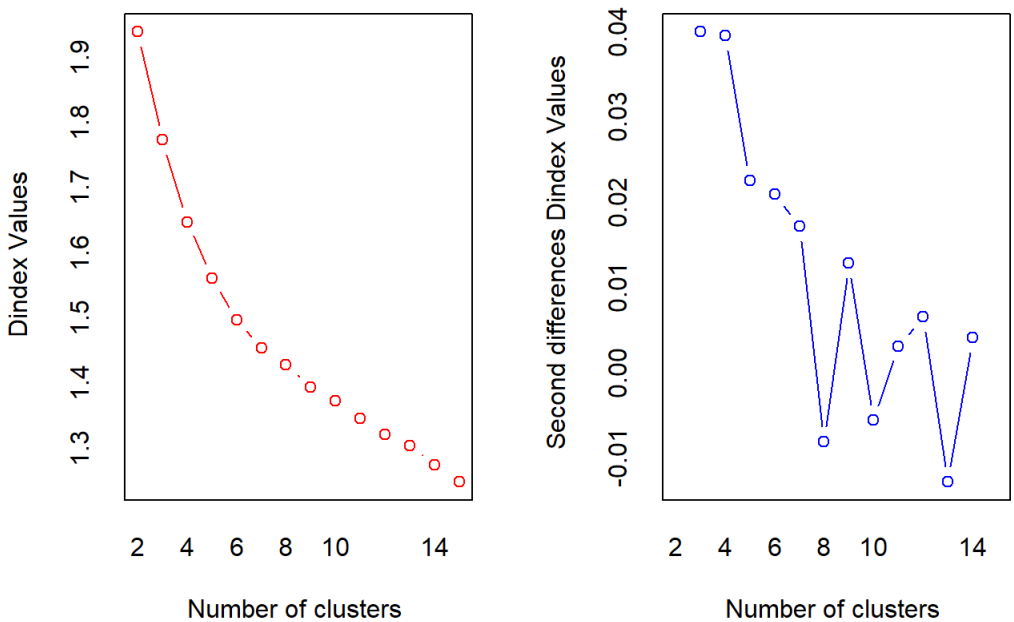
```
#The ARI provides an index that is close to 0 because it takes into account the chance of overlap.
#In addition, note that the ARI is a negative value indicating that the amount of overlap is less than expected.
##Meila's VI Index
clust_stats.c.min$vi
```

```
## [1] 0.7823353
```

```
#Ward's method (Manhattan dist)
nb6<-NbClust(scale_df, distance="manhattan", method="ward.D2")
```




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

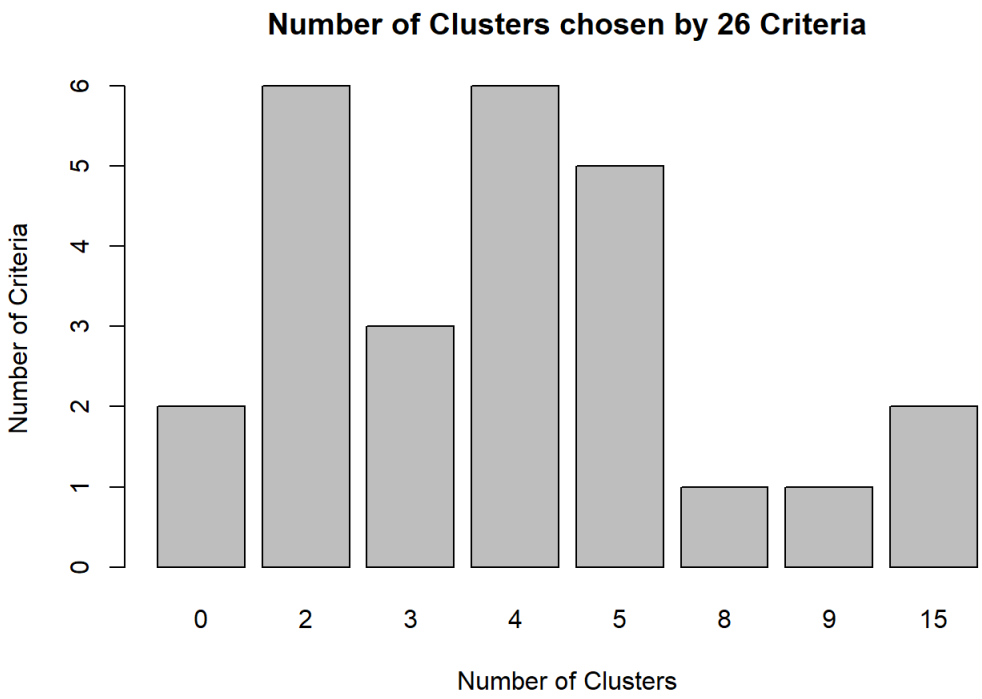


```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 6 proposed 2 as the best number of clusters
## * 3 proposed 3 as the best number of clusters
## * 6 proposed 4 as the best number of clusters
## * 5 proposed 5 as the best number of clusters
## * 1 proposed 8 as the best number of clusters
## * 1 proposed 9 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 is  2
##
##
## *******************************************************************
```

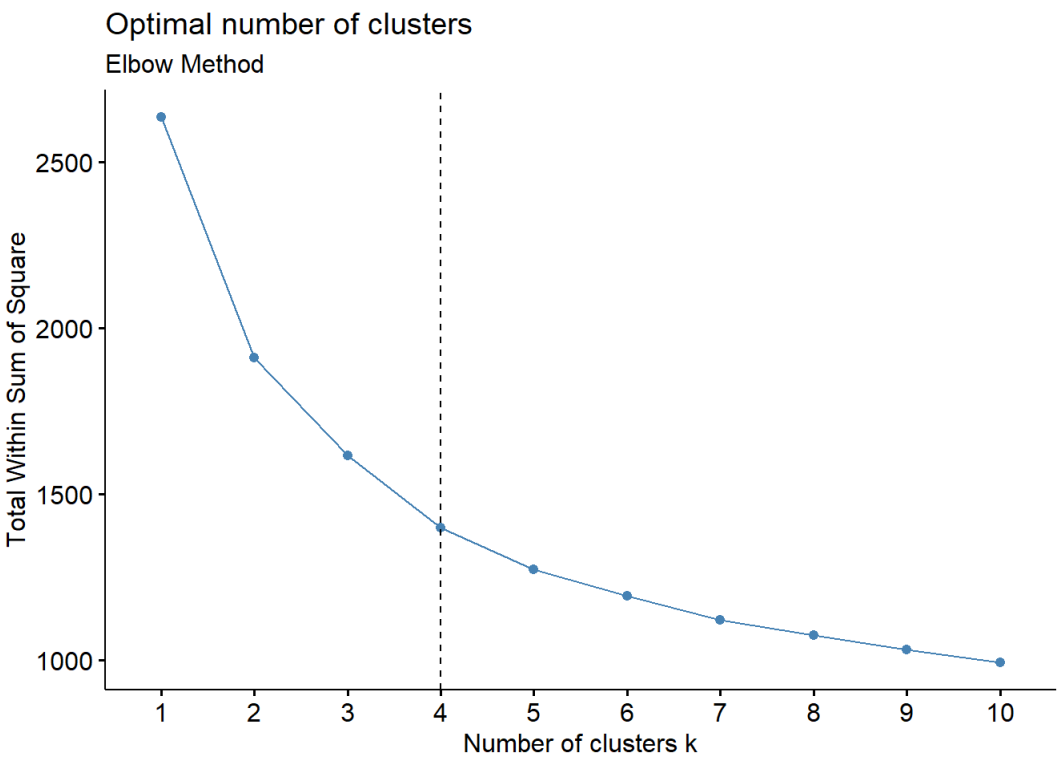
```
summary(nb6)
```

```
##                  Length Class  Mode
## All.index          364    -none- numeric
## All.CriticalValues  42    -none- numeric
## Best.nc             52    -none- numeric
## Best.partition     440    -none- numeric
```

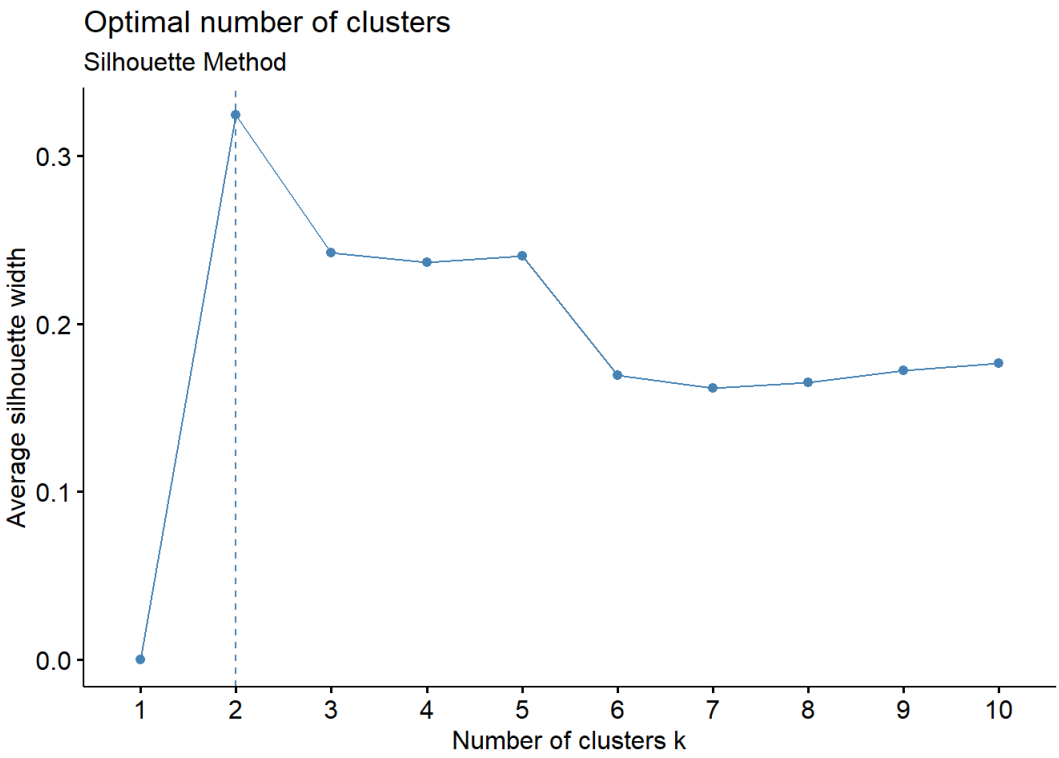
```
t6 <- table(nb6$Best.nc[1,])

par(mfrow= c(1,1))
barplot(t6,xlab="Number of Clusters",ylab="Number of Criteria",
        main="Number of Clusters chosen by 26 Criteria")
```




```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, hcut, method="wss", hc_metric="manhattan", hc_method="ward.D2")+geom_vline(xintercept = 4, linetype=2)+labs(subtitle = "Elbow Method")
```

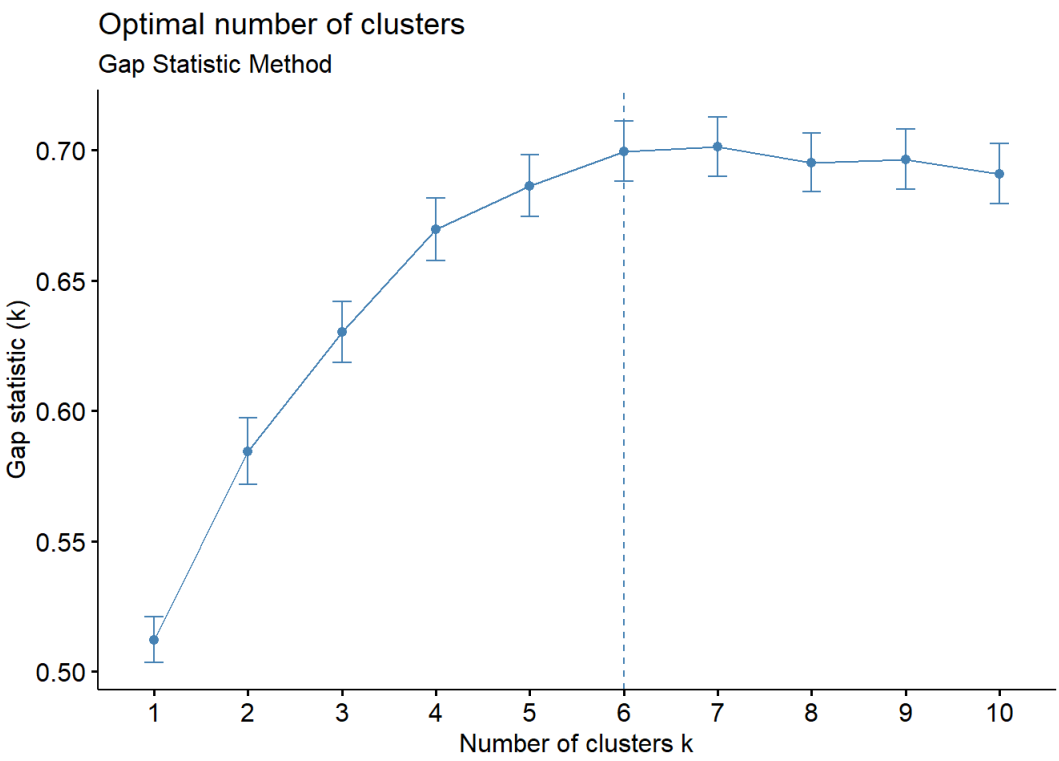



```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, hcut, method="silhouette", hc_metric="manhattan", hc_method="ward.D2")+labs(subtitle = "Silhouette Method")
```

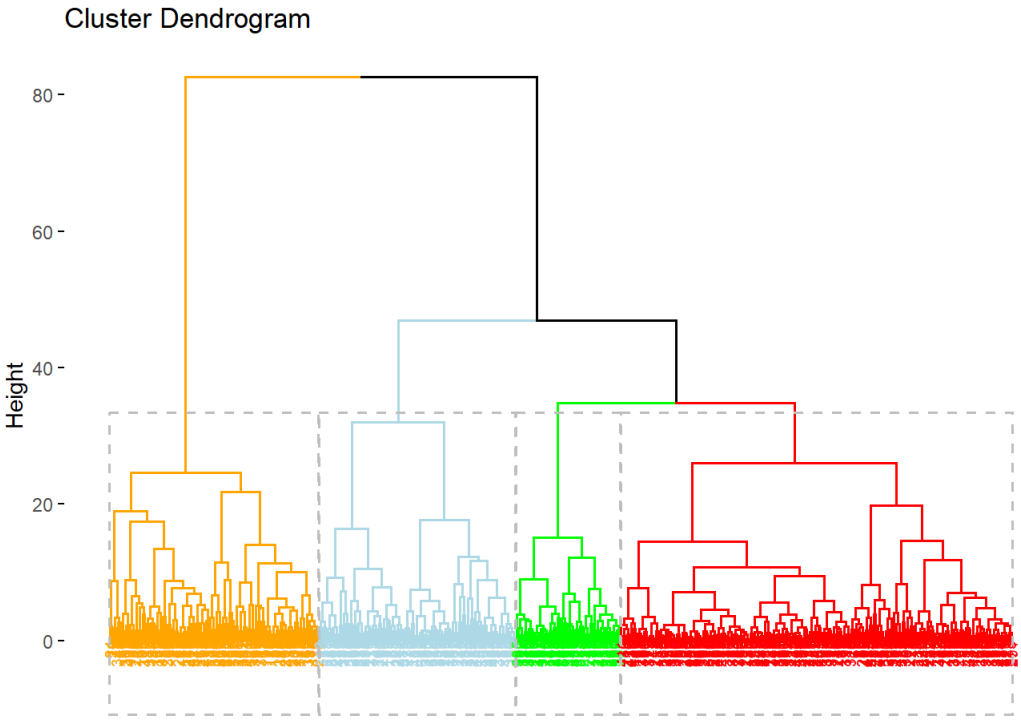



```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, hcut, method="gap_stat", nboot=500, hc_metric="manhattan", hc_method="ward.D2")+labs(subtitle = "Gap Statistic Method")
```

```
## 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
```




```
#according with this method, 4 clusters is the most suggested choice
ward.man.hc<-hclust(d=D_man, method="ward.D2")

par(mfrow=c(1,1))
fviz_dend(ward.man.hc, k=4, cex=0.5,k_colors=c("orange","lightblue","green","red"),rect=TRUE)
```



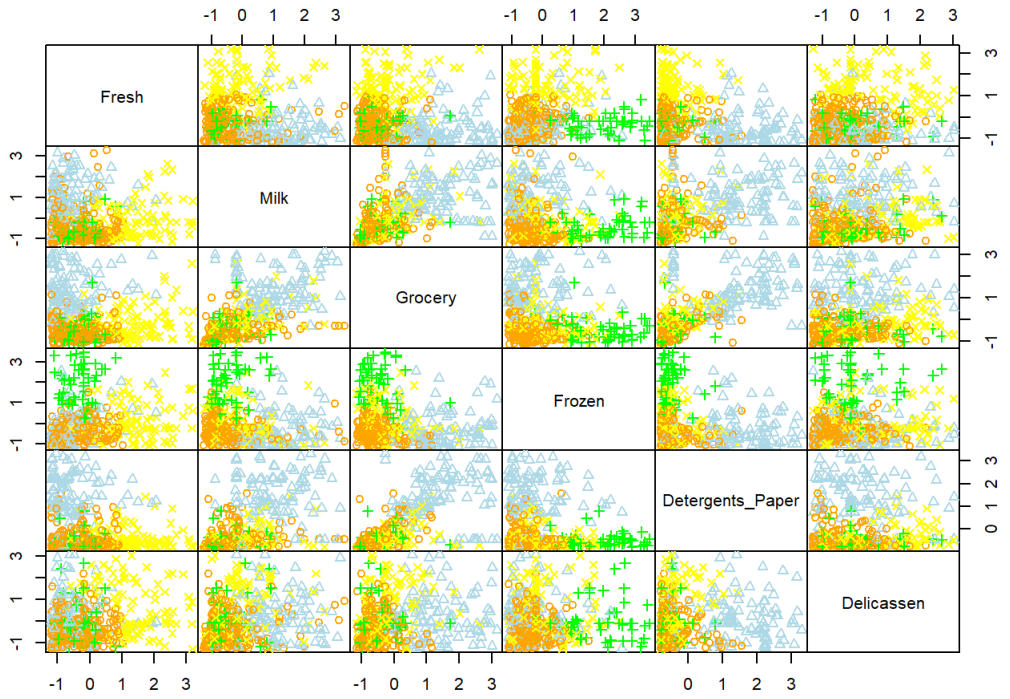

```
ward.man.coph<-cophenetic(ward.man.hc)
cor(D_man, ward.man.coph)
```

```
## [1] 0.6486654
```

```
grp.w<-cutree(ward.man.hc, k=4)
table(grp.w)
```

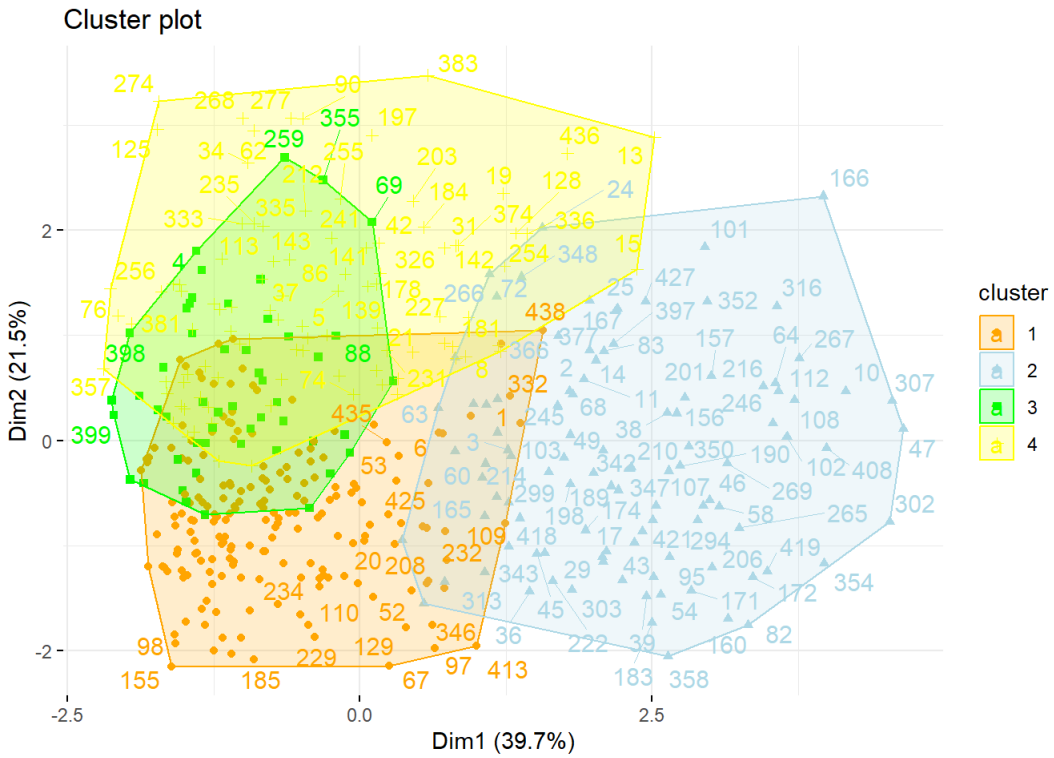
```
## grp.w
##   1   2   3   4
## 191 102  51  96
```

```
par(mfrow=c(1,1))
pairs(scale_df, gap=0, pch=grp.w, col=c("orange","lightblue","green","yellow")[grp.w])
```



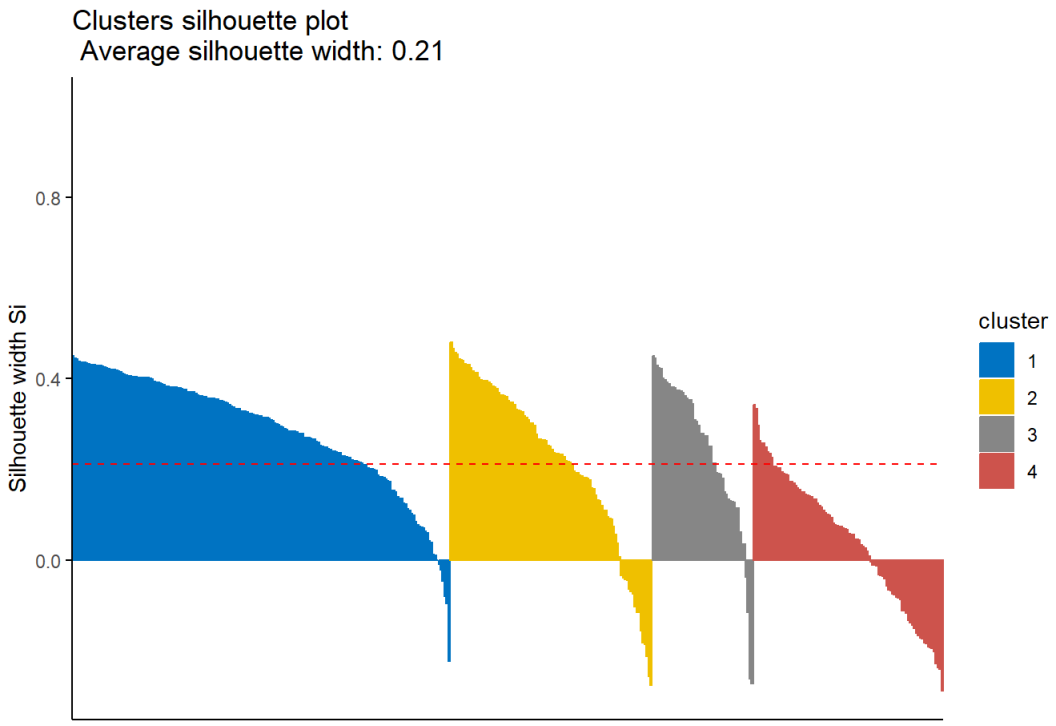

```
par(mfrow=c(1,1))
fviz_cluster(list(data=scale_df, cluster=grp.w), palette=c("orange","lightblue","green","yellow"),
             ellipse.type="convex",repel=TRUE, show.clust.cent=FALSE, ggtheme= theme_minimal())
```

```
## Warning: ggrepel: 282 unlabeled data points
## (too many overlaps). Consider increasing
## max.overlaps
```




```
#Internal Validation
w.res<-eclust(scale_df, "hclust", k=4, hc_metric="manhattan", hc_method="ward.D2", graph=FALSE)
fviz_silhouette(w.res, palette="jco", ggtheme=theme_classic())
```

```
##   cluster size ave.sil.width
## 1       1  191         0.29
## 2       2  102         0.22
## 3       3   51         0.24
## 4       4   96         0.04
```




```
silinfo.w<-w.res$silinfo
neg_sil.w<-which(silinfo.w$widths[,"sil_width"]<0)

silinfo.w$avg.width
```

```
## [1] 0.211195
```

```
silinfo.w$clus.avg.widths
```

```
## [1] 0.28581309 0.21779421 0.24170608
## [4] 0.03951562
```

```
silinfo.w$widths[neg_sil.w,,drop=FALSE]
```

```
##     cluster neighbor     sil_width
## 91        1        3 -0.0008463006
## 438       1        4 -0.0085819538
## 1         1        2 -0.0208482455
## 27        1        3 -0.0458298968
## 114       1        3 -0.0804528128
## 202       1        3 -0.0957642480
## 314       1        3 -0.2224867105
## 24        2        4 -0.0344369364
## 75        2        1 -0.0383208326
## 60        2        1 -0.0422415264
## 29        2        1 -0.0443267486
## 431       2        1 -0.0644033249
## 124       2        1 -0.0696930904
## 266       2        1 -0.0751797176
## 188       2        3 -0.1032228891
## 72        2        3 -0.1149952880
## 320       2        1 -0.1153038972
## 313       2        1 -0.1562149066
## 63        2        3 -0.1826621707
## 137       2        3 -0.1852537685
## 173       2        1 -0.2116617901
## 304       2        1 -0.2566099964
## 209       2        1 -0.2757869936
## 322       3        1 -0.0379402301
## 410       3        1 -0.1152690898
## 315       3        1 -0.2605455240
## 126       3        1 -0.2721541245
## 13        4        2 -0.0036833217
## 231       4        1 -0.0102211566
## 403       4        1 -0.0102777233
## 285       4        1 -0.0117170905
## 233       4        1 -0.0322329156
## 104       4        1 -0.0333953432
## 297       4        3 -0.0357729257
## 381       4        3 -0.0407041765
## 34        4        3 -0.0567775238
## 357       4        1 -0.0650790437
## 388       4        1 -0.0664026031
## 8         4        2 -0.0750019245
## 244       4        1 -0.0756158758
## 235       4        3 -0.0815270503
## 84        4        1 -0.0823160503
## 94        4        1 -0.0863267414
## 361       4        1 -0.1111425681
## 369       4        1 -0.1111426659
## 414       4        1 -0.1160491505
## 15        4        2 -0.1332492563
## 372       4        1 -0.1369899925
## 263       4        1 -0.1430422353
## 371       4        1 -0.1495908894
## 153       4        1 -0.1609935889
## 158       4        3 -0.1666962964
## 433       4        1 -0.1717189327
## 405       4        3 -0.1726786796
## 212       4        3 -0.1817783272
## 238       4        3 -0.1836401824
## 255       4        3 -0.1916403285
## 243       4        3 -0.1928993852
## 76        4        3 -0.1940632103
## 33        4        1 -0.2015624601
## 127       4        1 -0.2274207301
## 323       4        3 -0.2372027975
## 163       4        1 -0.2405323921
## 119       4        3 -0.2874696927
```

```
w.link<-cluster.stats(D_man, w.res$cluster)
w.link$dunn
```

```
## [1] 0.07270178
```

```
#External validation
#confusion matrix
table(data$Channel, w.res$cluster)
```

```
##
##      1   2   3   4
##   1 160  18  47  73
##   2  31  84   4  23
```

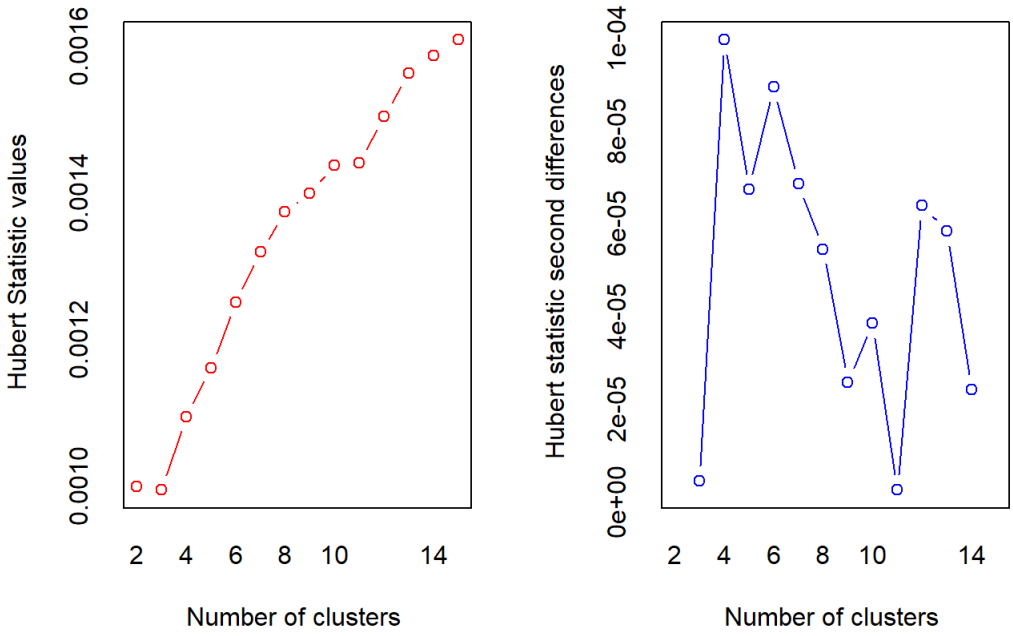
```
#rand index
tipes<-as.numeric(data$Channel)
clust_stats.w<-cluster.stats(d=D_euc, tipes, w.res$cluster)
clust_stats.w$corrected.rand
```

```
## [1] 0.1746772
```

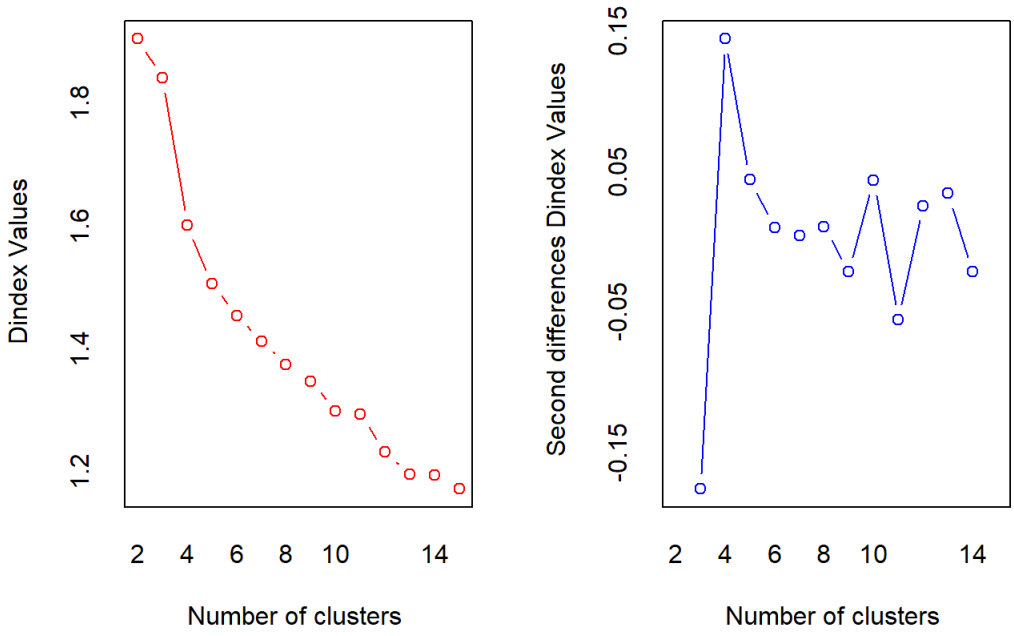
```
#The ARI provides an index that is close to 0 because it takes into account the chance of overlap.
#In addition, note that the ARI is a negative value indicating that the amount of overlap is less than expected.
##Meila's VI Index
clust_stats.w$vi
```

```
## [1] 1.559218
```

```
#K-means method (Euclidean dist)
nb7<-NbClust(scale_df, distance="euclidean", method="kmeans")
```




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

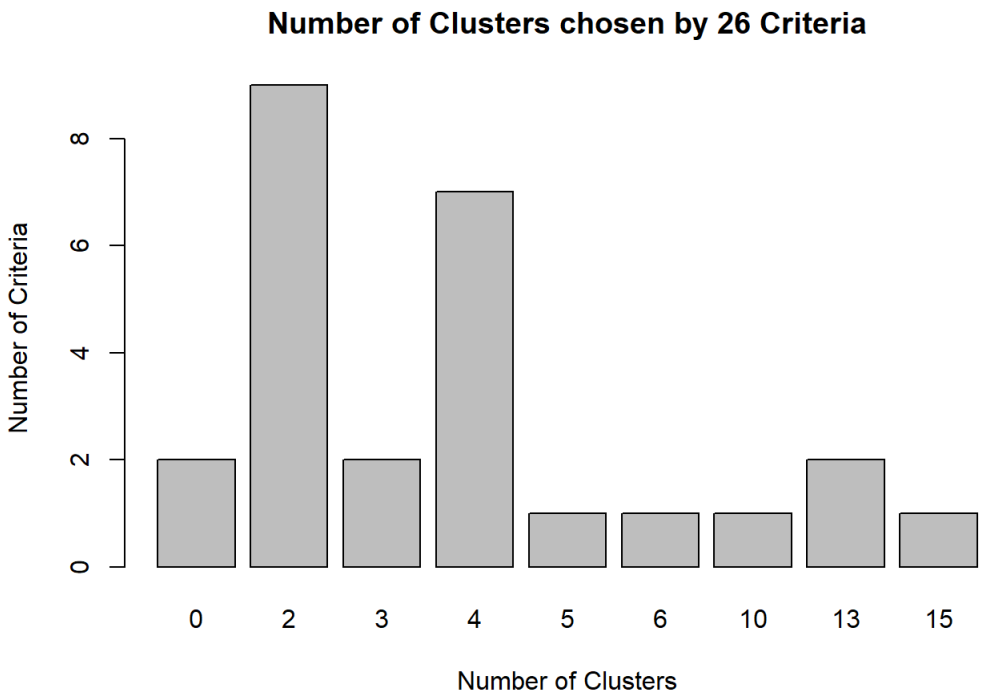



```
## *** : The D index is a graphical method of determining the number of clusters.
##               In the plot of D index, we seek a significant knee (the significant peak in Dindex
##               second differences plot) that corresponds to a significant increase of the value of
##               the measure.
##
## *******************************************************************
## * Among all indices:
## * 9 proposed 2 as the best number of clusters
## * 2 proposed 3 as the best number of clusters
## * 7 proposed 4 as the best number of clusters
## * 1 proposed 5 as the best number of clusters
## * 1 proposed 6 as the best number of clusters
## * 1 proposed 10 as the best number of clusters
## * 2 proposed 13 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 is  2
##
##
## *******************************************************************
```

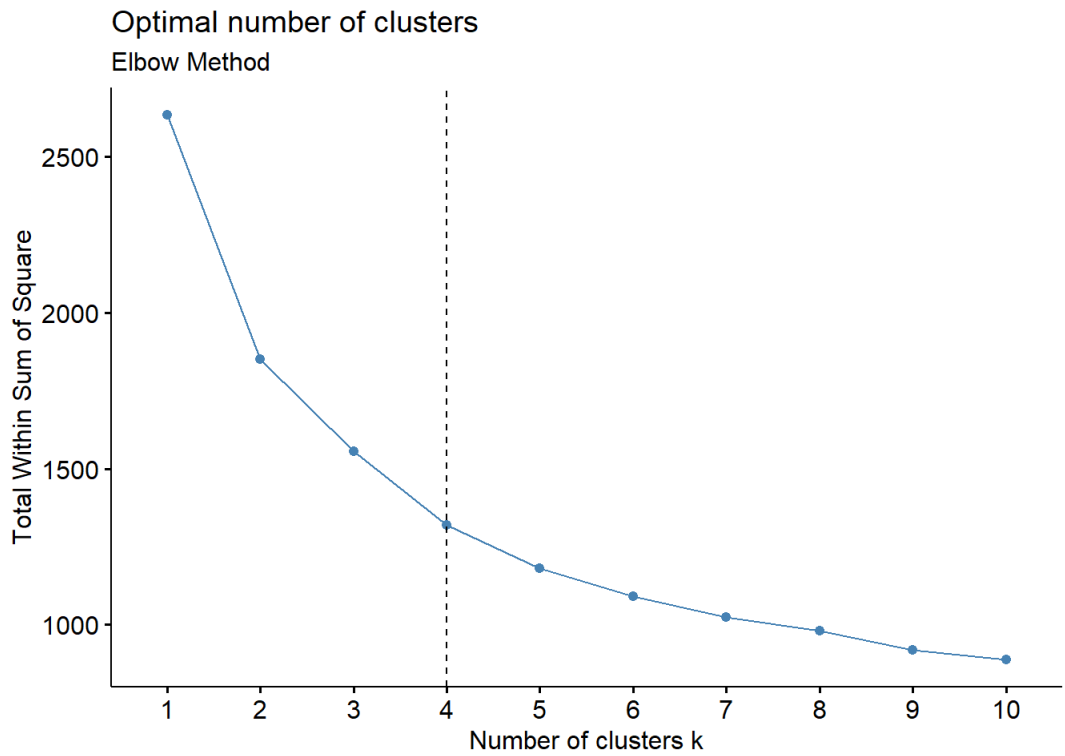
```
summary(nb7)
```

```
##                  Length Class  Mode
## All.index          364    -none- numeric
## All.CriticalValues  42    -none- numeric
## Best.nc             52    -none- numeric
## Best.partition     440    -none- numeric
```

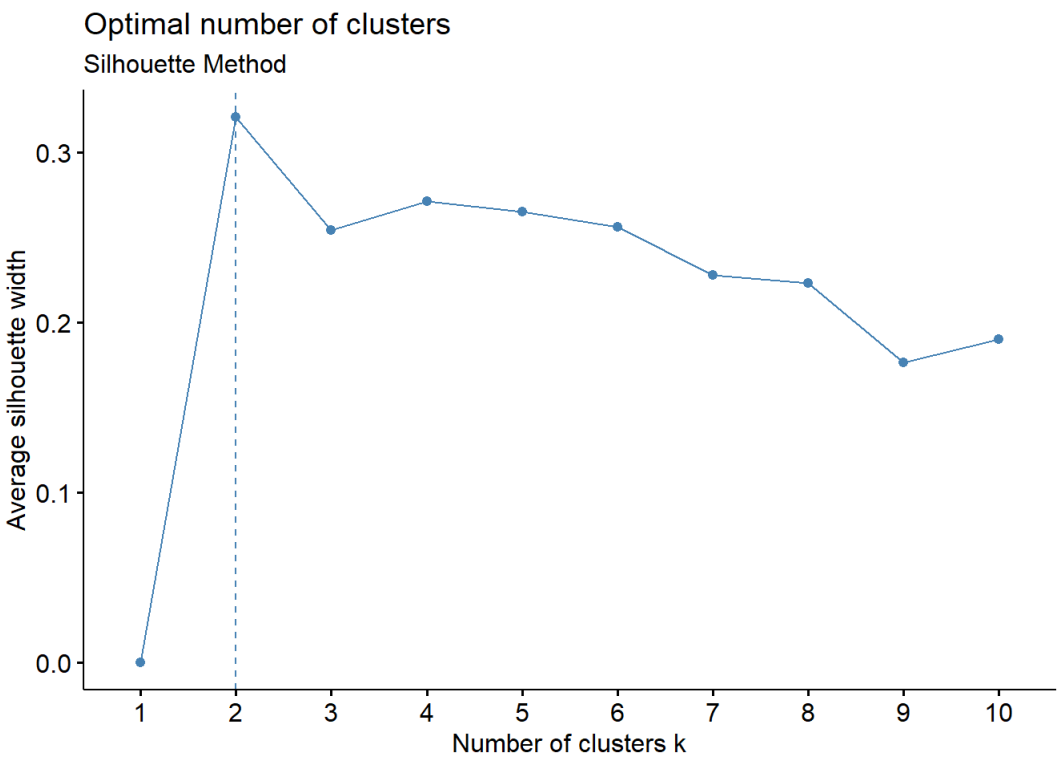
```
t7 <- table(nb7$Best.nc[1,])

par(mfrow= c(1,1))
barplot(t7,xlab="Number of Clusters",ylab="Number of Criteria",
        main="Number of Clusters chosen by 26 Criteria")
```




```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, kmeans, method="wss")+geom_vline(xintercept = 4, linetype=2)+labs(subtitle = "Elbow Method")
```




```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, kmeans, method="silhouette")+labs(subtitle = "Silhouette Method")
```




```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, kmeans, method="gap_stat", nboot=500)+labs(subtitle = "Gap Statistic Method")
```

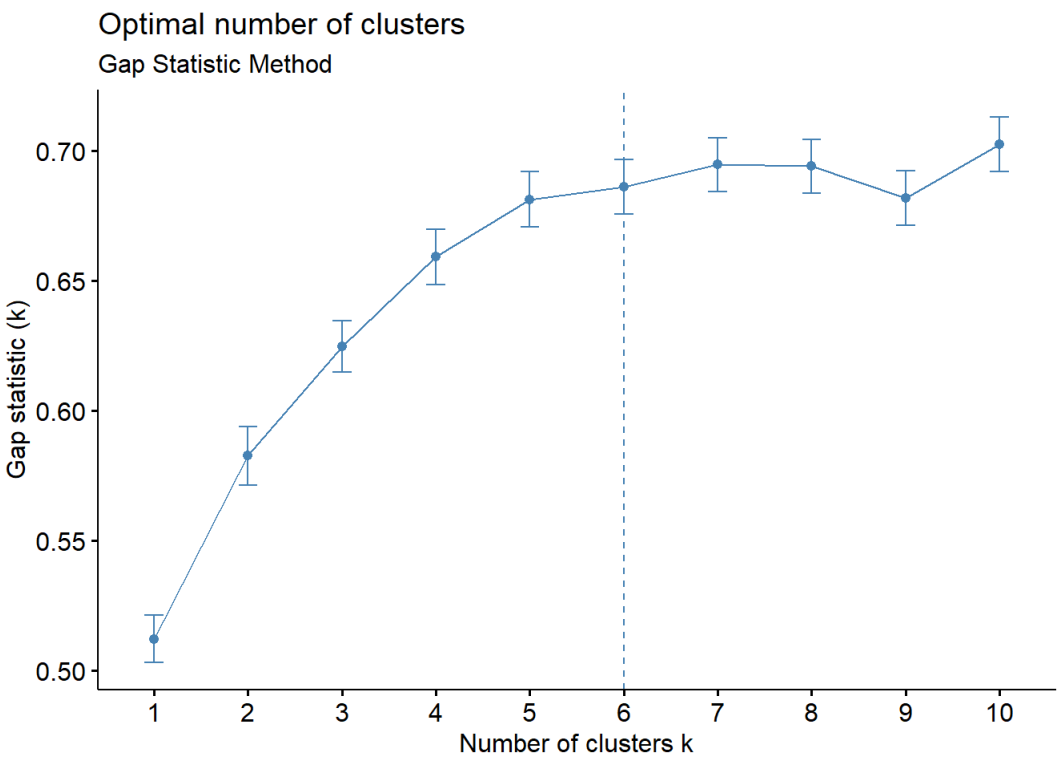
```
## Clustering k = 1,2,..., K.max (= 10): .. done
## Bootstrapping, b = 1,2,..., B (= 500)  [one "." per sample]:
## ................................................. 50
## ................................................. 100
## ................................................. 150
## ..................
```

```
## Warning: non converge in 10 iterazioni
```

```
## ............................. 200
## ..............................
```

```
## Warning: non converge in 10 iterazioni
```

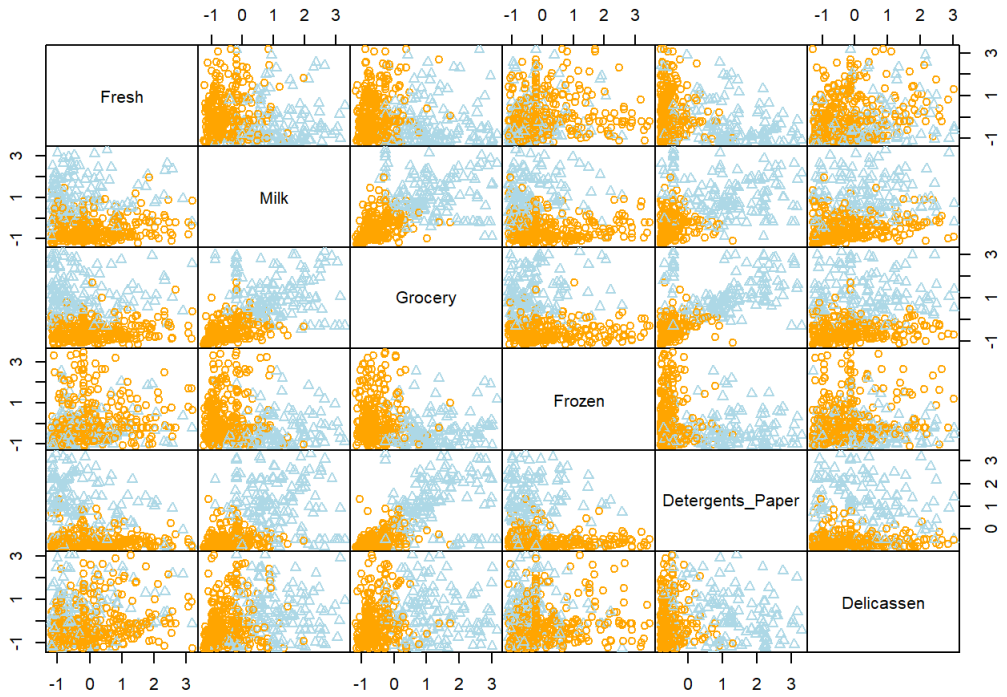
```
## ................. 250
## ................................................. 300
## ................................................. 350
## ................................................. 400
## ................................................. 450
## ................................................. 500
```




```
#according with this method, 2 clusters is the most suggested choice

set.seed(123)
km.euc<-kmeans(scale_df, 2, iter.max=100, nstart=50)
print(km.euc)
```

```
## K-means clustering with 2 clusters of sizes 299, 141
##
## Cluster means:
##        Fresh      Milk    Grocery     Frozen
## 1  0.1098610 -0.488112 -0.5361535  0.1455237
## 2 -0.2329676  1.035074  1.1369497 -0.3085928
##   Detergents_Paper Delicassen
## 1       -0.5107819 -0.1402449
## 2        1.0831475  0.2973988
##
## Clustering vector:
##   [1] 2 2 2 1 1 2 1 2 1 2 2 1 2 2 2 1 2 1 2 1
##  [21] 1 1 1 2 2 2 1 1 2 1 2 1 1 1 1 2 1 2 2 1
##  [41] 1 1 2 2 2 2 2 1 2 1 1 1 1 2 1 1 1 2 1 2
##  [61] 2 1 2 2 1 1 1 2 1 1 1 2 1 1 2 1 1 2 1 1
##  [81] 1 2 2 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1
## [101] 2 2 2 1 1 1 2 2 2 1 1 2 1 1 1 1 1 1 1 1
## [121] 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 2 2 1 1
## [141] 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 2
## [161] 2 1 1 1 2 2 2 1 1 1 2 2 2 2 1 2 1 1 1 1
## [181] 2 2 2 2 1 1 1 2 2 2 1 1 1 2 1 1 1 2 1 1
## [201] 2 2 2 1 1 2 1 2 1 2 1 1 1 2 2 2 1 1 2 1
## [221] 1 2 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1
## [241] 1 1 1 1 2 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1
## [261] 1 1 1 1 2 2 2 1 2 1 1 1 2 1 1 1 1 1 1 2
## [281] 1 2 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1
## [301] 2 2 2 2 2 2 2 1 1 2 1 1 2 1 1 2 1 1 1 2
## [321] 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1
## [341] 2 2 2 1 1 2 2 2 1 2 1 2 1 2 1 1 1 2 1 1
## [361] 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 2 1 1 1
## [381] 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
## [401] 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 2 2 2 2 1
## [421] 2 2 1 1 2 1 2 1 1 1 2 1 1 1 1 2 1 2 1 1
##
## Within cluster sum of squares by cluster:
## [1] 1054.3612  796.3182
##  (between_SS / total_SS =  29.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"
## [3] "totss"        "withinss"
## [5] "tot.withinss" "betweenss"
## [7] "size"         "iter"
## [9] "ifault"
```

```
#Visualize results
km.cl<-km.euc$cluster

par(mfrow=c(1,1))
pairs(scale_df, gap=0, pch=km.cl, col=c("orange","lightblue")[km.cl])
```




```
#Profiling Cluster
aggregate(data[3:8], by=list(cluster=km.euc$cluster), mean)
```

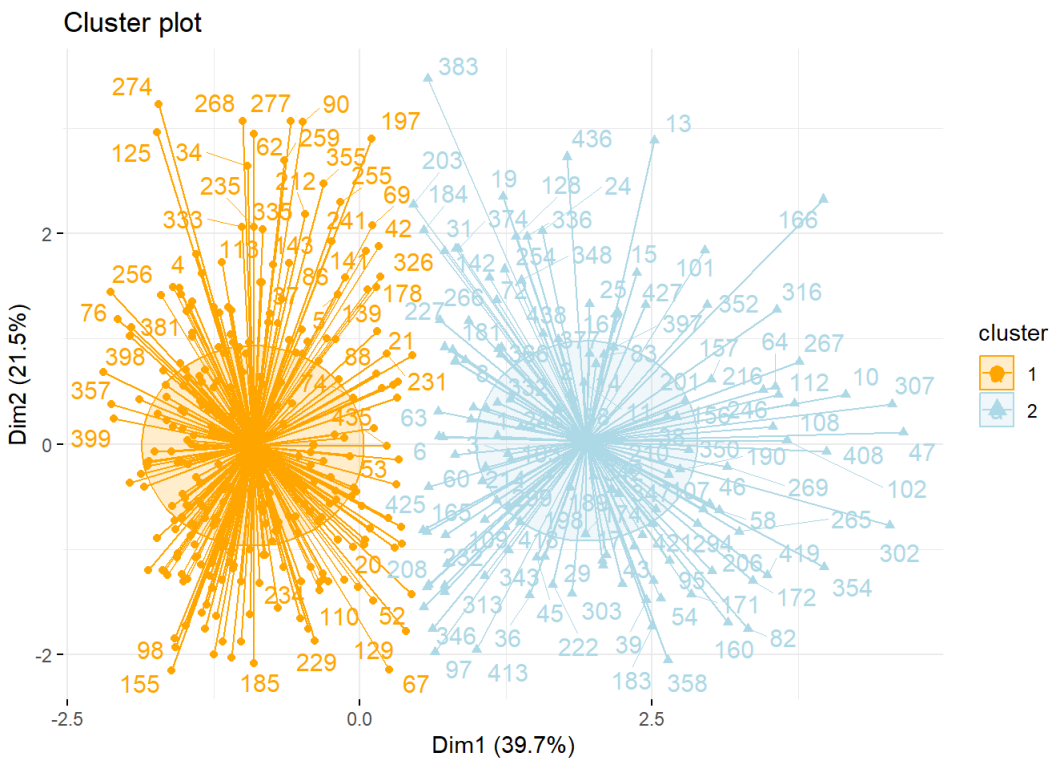
```
##   cluster     Fresh     Milk   Grocery
## 1       1 10961.465 2621.087  3355.963
## 2       2  8064.376 7803.546 12289.837
##     Frozen Detergents_Paper Delicassen
## 1 2157.997         652.6839   966.4314
## 2 1386.837        4251.7553  1334.2376
```

```
#explain
#In cluster 1 it seems that units have higher value of all the variables except for Annual Spending
# for Fresh and Frozen Product, this variables doesn't seem good to clusterize data.

#Adding the classification to the original dataset:
data.km<-cbind(data, cluster=km.euc$cluster)
head(data.km)
```

```
##   Channel Region Fresh Milk Grocery Frozen
## 1       2      3 12669 9656    7561    214
## 2       2      3  7057 9810    9568   1762
## 3       2      3  6353 8808    7684   2405
## 4       1      3 13265 1196    4221   6404
## 5       2      3 22615 5410    7198   3915
## 6       2      3  9413 8259    5126    666
##   Detergents_Paper Delicassen cluster
## 1             2674     1338.0       2
## 2             3293     1776.0       2
## 3             3516      965.5       2
## 4              507     1788.0       1
## 5             1777      965.5       1
## 6             1795     1451.0       2
```

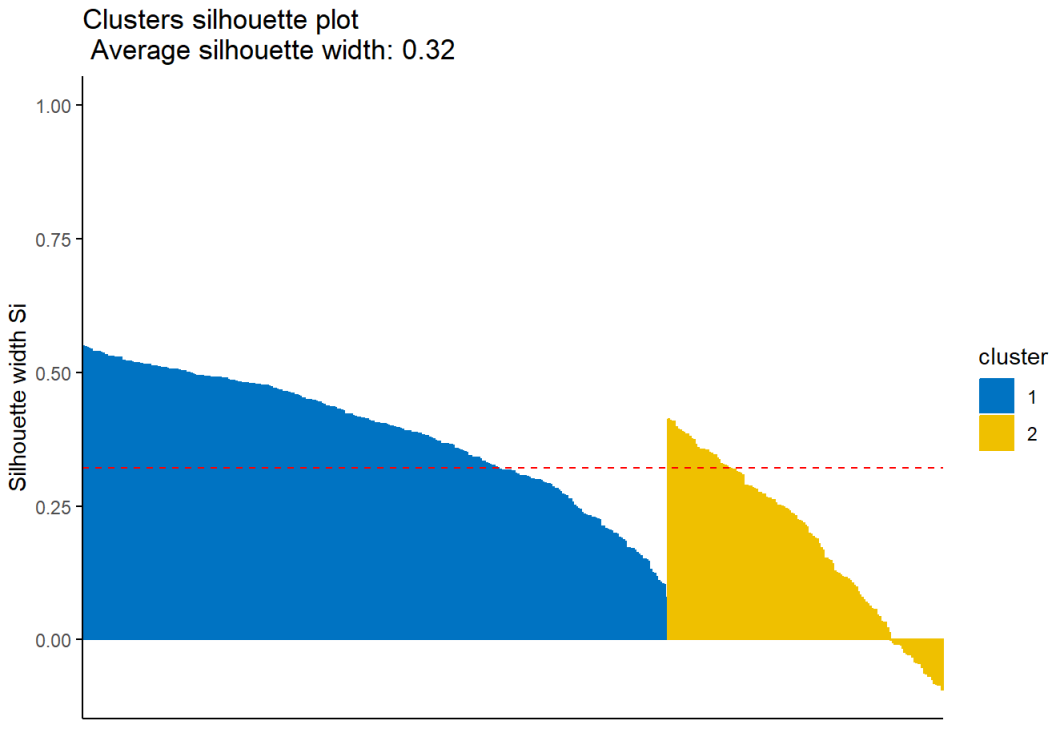
```
par(mfrow=c(1,1))
fviz_cluster(km.euc, data=scale_df, palette=c("orange","lightblue"),ellipse.type="euclid", star.plot=TRUE,
             repel = TRUE, ggtheme=theme_minimal())
```

```
## Warning: ggrepel: 282 unlabeled data points
## (too many overlaps). Consider increasing
## max.overlaps
```




```
#INternal Validation
km.euc.ec<-eclust(scale_df, "kmeans", k=2, hc_metric="euclidean", graph=FALSE)
fviz_silhouette(km.euc.ec, palette="jco", ggtheme=theme_classic())
```

```
##   cluster size ave.sil.width
## 1       1  299          0.39
## 2       2  141          0.18
```




```
silinfo.km.euc<-km.euc.ec$silinfo
silinfo.km.euc$avg.width
```

```
## [1] 0.320847
```

```
silinfo.km.euc$clus.avg.widths
```

```
## [1] 0.3864921 0.1816423
```

```
neg_sil.km.euc<-which(silinfo.km.euc$widths[,"sil_width"]<0)
silinfo.km.euc$widths[neg_sil.km.euc,,drop=FALSE]
```

```
##     cluster neighbor     sil_width
## 188       2        1 -0.0004688525
## 374       2        1 -0.0056451747
## 137       2        1 -0.0079474045
## 31        2        1 -0.0086474081
## 142       2        1 -0.0087814720
## 282       2        1 -0.0092260454
## 208       2        1 -0.0159607639
## 285       2        1 -0.0230623183
## 184       2        1 -0.0263769074
## 232       2        1 -0.0275468724
## 44        2        1 -0.0284734105
## 383       2        1 -0.0321639652
## 301       2        1 -0.0413561812
## 63        2        1 -0.0433440850
## 227       2        1 -0.0441721899
## 203       2        1 -0.0443391842
## 6         2        1 -0.0520313973
## 305       2        1 -0.0623330505
## 173       2        1 -0.0641971302
## 97        2        1 -0.0679313972
## 425       2        1 -0.0680157559
## 346       2        1 -0.0747209452
## 138       2        1 -0.0821046651
## 61        2        1 -0.0833909417
## 304       2        1 -0.0844457955
## 26        2        1 -0.0855312554
## 273       2        1 -0.0935915849
```

```
k.link<-cluster.stats(D_euc, km.euc.ec$cluster)
k.link$dunn
```

```
## [1] 0.06485737
```

```
#External validation for k=2

table(data$Channel, km.euc.ec$cluster)
```

```
##
##      1   2
##   1 267  31
##   2  32 110
```

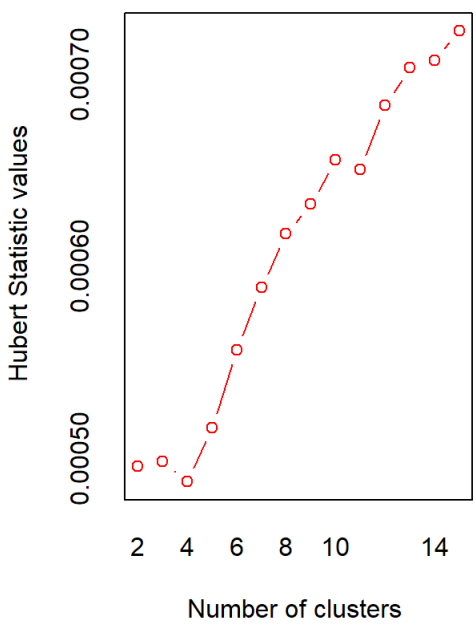
```
#In this case there' s no perfect agreement between the external information and the cluster structure

tipes<-as.numeric(data$Channel)
clust_stats.km.euc<-cluster.stats(d=D_euc, tipes, km.euc$cluster)
clust_stats.km.euc$corrected.rand
```

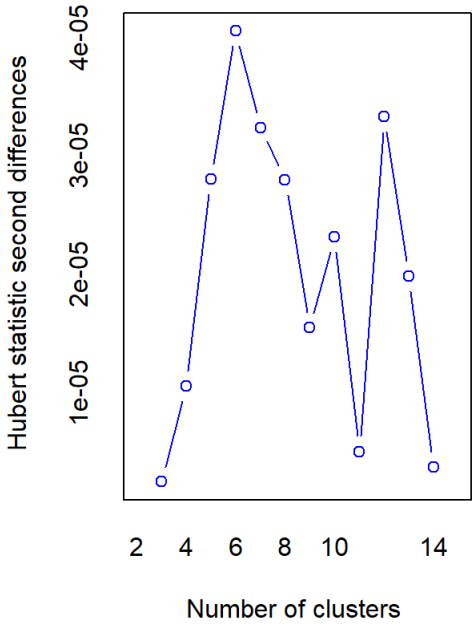
```
## [1] 0.5003106
```

```
#According to the correct Rand index there is a good agreement between the seed types and the cluster
#solution

clust_stats.km.euc$vi
```

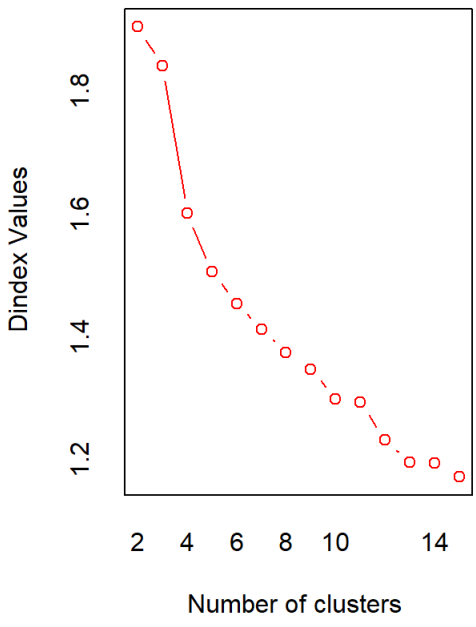
```
## [1] 0.7983143
```

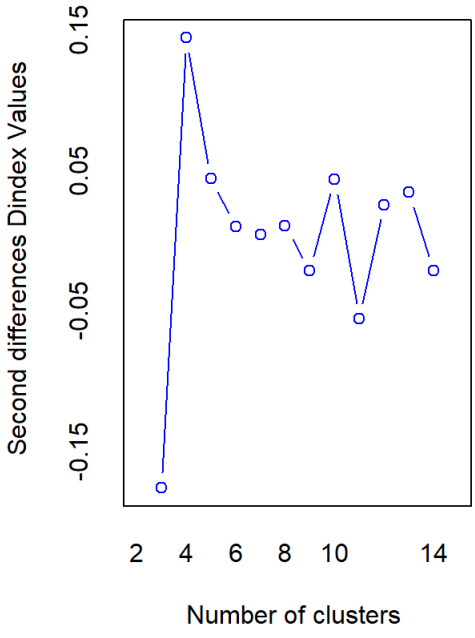
```
#K-means method (Manhattan dist)
nb8<-NbClust(scale_df, distance="manhattan", method="kmeans")
```






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



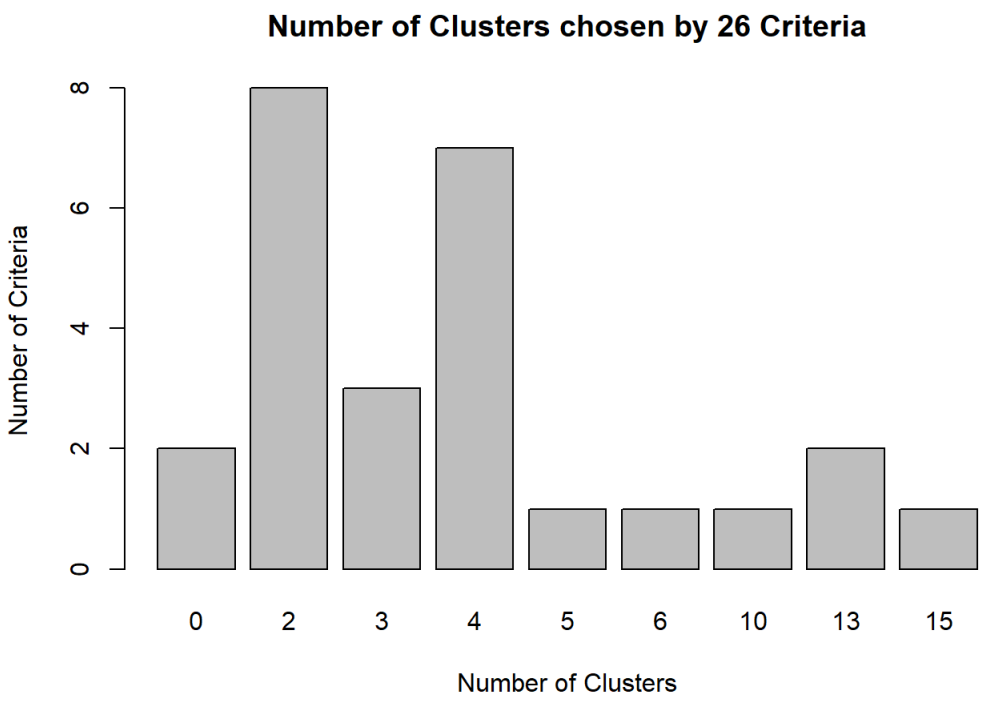


```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 8 proposed 2 as the best number of clusters
## * 3 proposed 3 as the best number of clusters
## * 7 proposed 4 as the best number of clusters
## * 1 proposed 5 as the best number of clusters
## * 1 proposed 6 as the best number of clusters
## * 1 proposed 10 as the best number of clusters
## * 2 proposed 13 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 is  2
##
##
## *******************************************************************
```

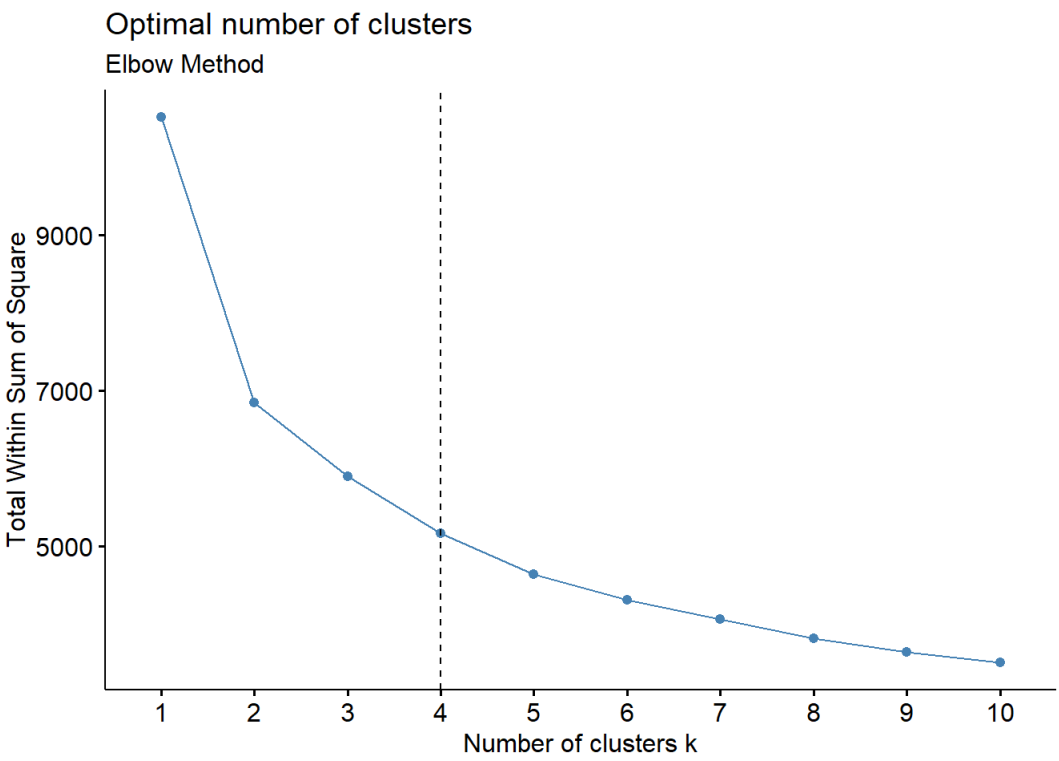
```
summary(nb8)
```

```
##                  Length Class  Mode
## All.index          364    -none- numeric
## All.CriticalValues  42    -none- numeric
## Best.nc             52    -none- numeric
## Best.partition     440    -none- numeric
```

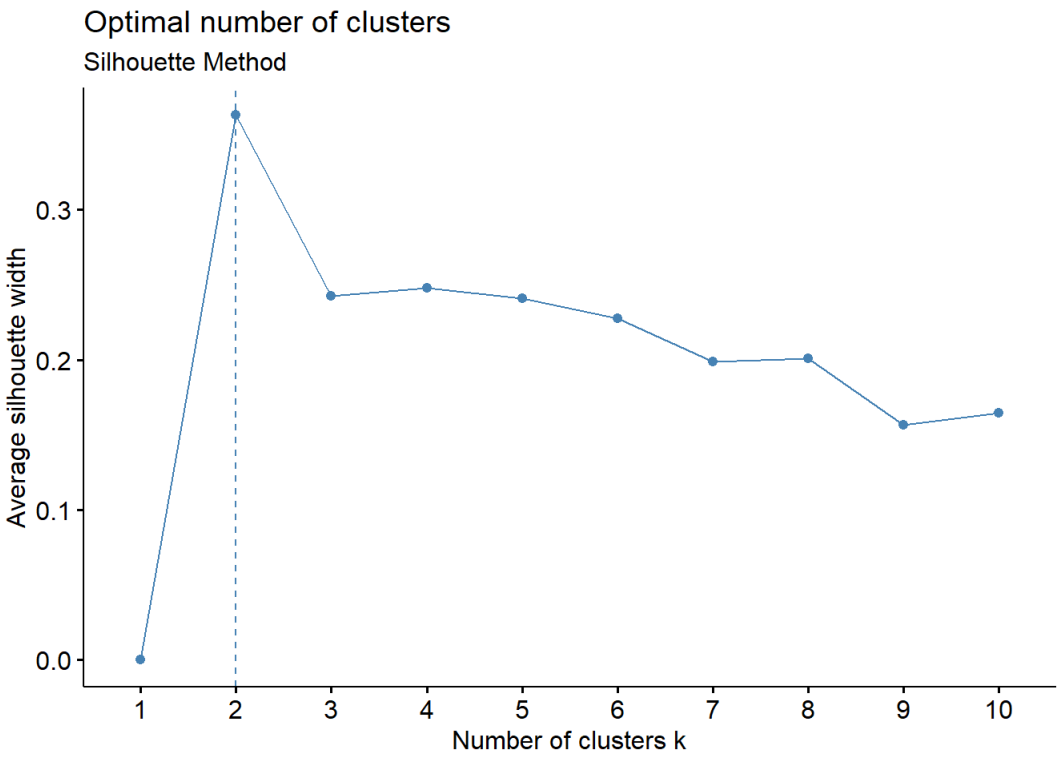
```
t8 <- table(nb8$Best.nc[1,])

par(mfrow= c(1,1))
barplot(t8,xlab="Number of Clusters",ylab="Number of Criteria",
        main="Number of Clusters chosen by 26 Criteria")
```




```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, kmeans, method="wss", diss= dist(scale_df, method="manhattan"))+geom_vline(xintercept = 4, linetype=2)+labs(subtitle = "Elbow Method")
```




```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, kmeans, method="silhouette", diss= dist(scale_df, method="manhattan"))+labs(subtitle = "Silhouette Method")
```




```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, kmeans, method="gap_stat",diss= dist(scale_df, method="manhattan"), nboot=500)+labs(subtitle = "Gap Statistic Method")
```

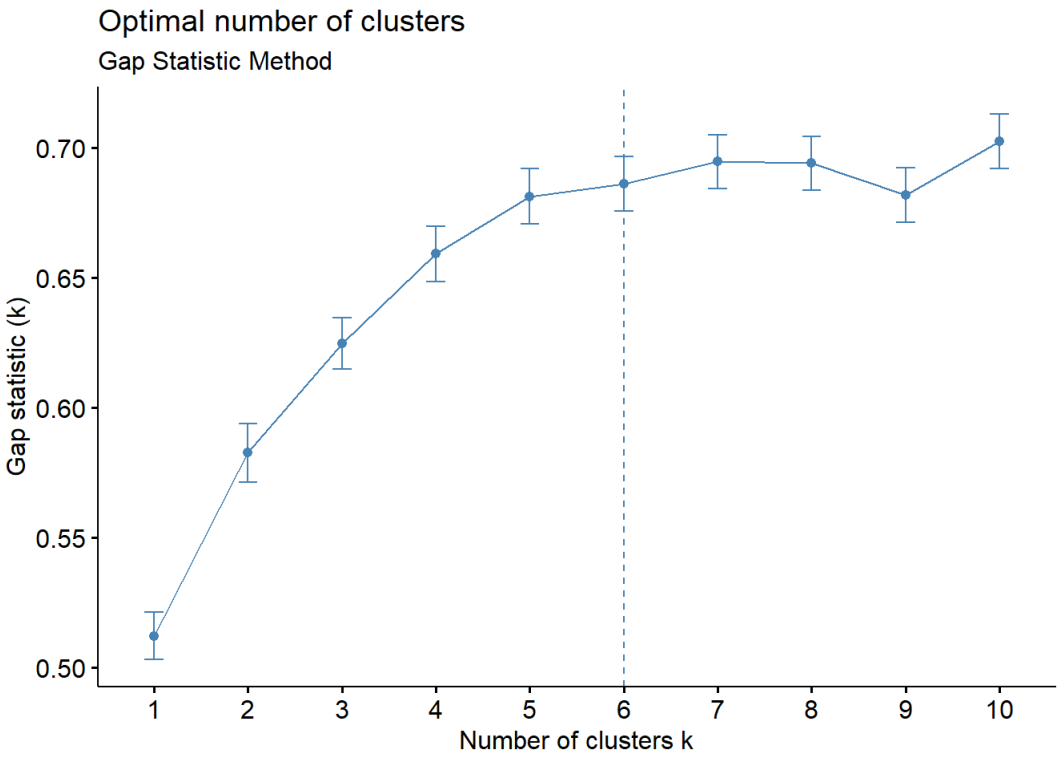
```
## Clustering k = 1,2,..., K.max (= 10): .. done
## Bootstrapping, b = 1,2,..., B (= 500)  [one "." per sample]:
## ..... 50
## ..... 100
## ..... 150
## .....
```

```
## Warning: non converge in 10 iterazioni
```

```
## ..... 200
## .....
```

```
## Warning: non converge in 10 iterazioni
```

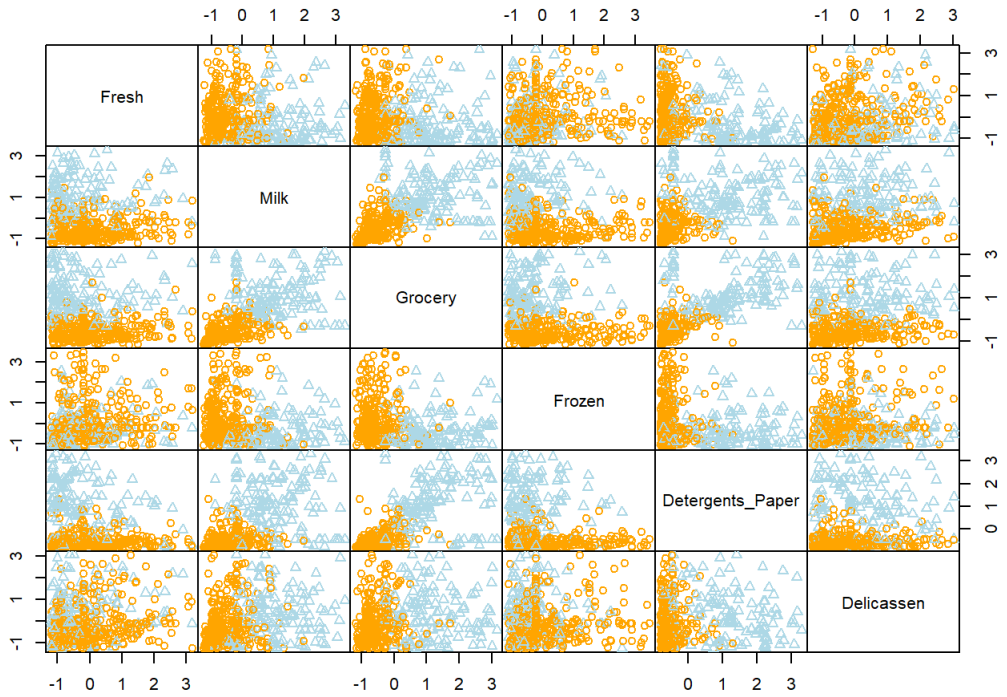
```
## ..... 250
## ..... 300
## ..... 350
## ..... 400
## ..... 450
## ..... 500
```




```
#according with this method, 2 clusters is the most suggested choice

set.seed(123)
km.man<-kmeans(scale_df, 2, iter.max = 100, nstart=50)
print(km.man)
```

```
## K-means clustering with 2 clusters of sizes 299, 141
##
## Cluster means:
##         Fresh       Milk    Grocery     Frozen
## 1  0.1098610 -0.488112 -0.5361535  0.1455237
## 2 -0.2329676  1.035074  1.1369497 -0.3085928
##   Detergents_Paper Delicassen
## 1       -0.5107819 -0.1402449
## 2        1.0831475  0.2973988
##
## Clustering vector:
##   [1] 2 2 2 1 1 2 1 2 1 2 2 1 2 2 2 1 2 1 2 1
##  [21] 1 1 1 2 2 2 1 1 2 1 2 1 1 1 1 2 1 2 2 1
##  [41] 1 1 2 2 2 2 2 1 2 1 1 1 1 2 1 1 1 2 1 2
##  [61] 2 1 2 2 1 1 1 2 1 1 1 2 1 1 2 1 1 2 1 1
##  [81] 1 2 2 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1
## [101] 2 2 2 1 1 1 2 2 2 1 1 2 1 1 1 1 1 1 1 1
## [121] 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 2 2 1 1
## [141] 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 2
## [161] 2 1 1 1 2 2 2 1 1 1 2 2 2 2 1 2 1 1 1 1
## [181] 2 2 2 2 1 1 1 2 2 2 1 1 1 2 1 1 1 2 1 1
## [201] 2 2 2 1 1 2 1 2 1 2 1 1 1 2 2 2 1 1 2 1
## [221] 1 2 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1
## [241] 1 1 1 1 2 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1
## [261] 1 1 1 1 2 2 2 1 2 1 1 1 2 1 1 1 1 1 1 2
## [281] 1 2 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1
## [301] 2 2 2 2 2 2 2 1 1 2 1 1 2 1 1 2 1 1 1 2
## [321] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1
## [341] 2 2 2 1 1 2 2 2 1 2 1 2 1 2 1 1 1 2 1 1
## [361] 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 2 1 1 1
## [381] 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
## [401] 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 2 2 2 2 1
## [421] 2 2 1 1 2 1 2 1 1 1 2 1 1 1 1 2 1 2 1 1
##
## Within cluster sum of squares by cluster:
## [1] 1054.3612  796.3182
##  (between_SS / total_SS =  29.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"
## [3] "totss"        "withinss"
## [5] "tot.withinss" "betweenss"
## [7] "size"         "iter"
## [9] "ifault"
```

```
#Visualize results
km.man.cl<-km.man$cluster

par(mfrow=c(1,1))
pairs(scale_df, gap=0, pch=km.man.cl, col=c("orange","lightblue")[km.man.cl])
```




```
#Profiling Cluster
aggregate(data[3:8], by=list(cluster=km.man$cluster), mean)
```

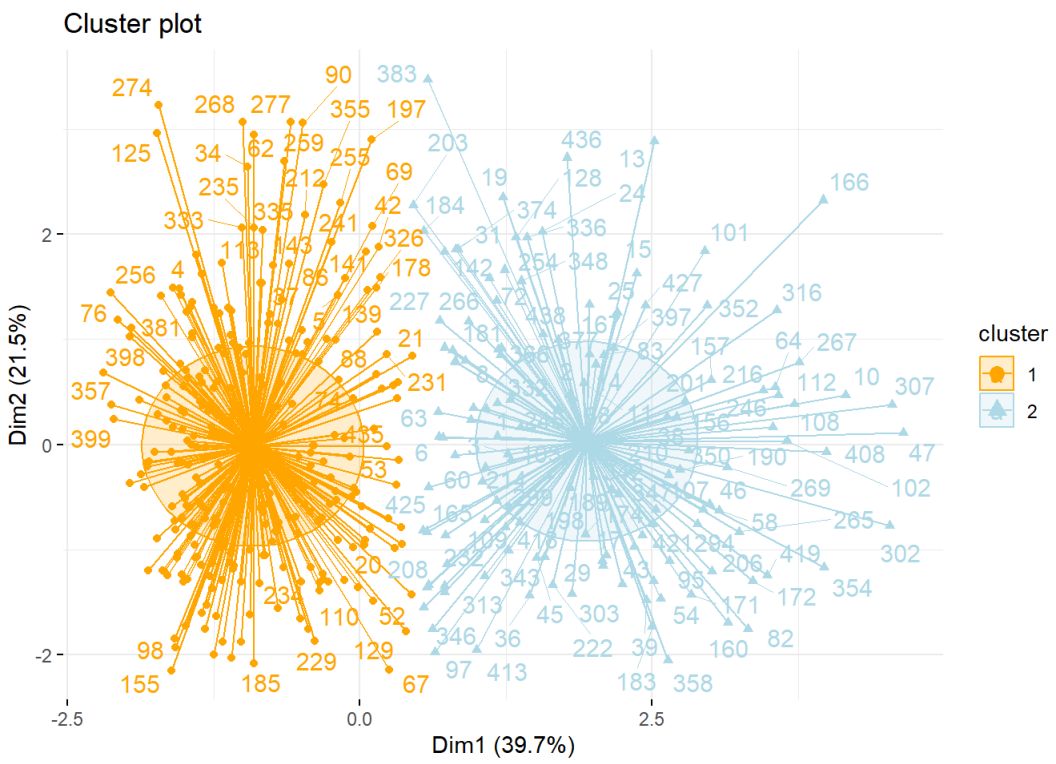
```
##   cluster     Fresh     Milk   Grocery
## 1       1 10961.465 2621.087  3355.963
## 2       2  8064.376 7803.546 12289.837
##     Frozen Detergents_Paper Delicassen
## 1 2157.997         652.6839   966.4314
## 2 1386.837        4251.7553  1334.2376
```

```
#explain
#In cluster 1 it seems that units have higher value of all the variables except for Annual Spending
# for Fresh and Frozen Product, this variables doesn't seem good to clusterize data.

#Adding the classification to the original dataset:
data.km.man<-cbind(data, cluster=km.man$cluster)
head(data.km.man)
```

```
##   Channel Region Fresh Milk Grocery Frozen
## 1       2      3 12669 9656    7561    214
## 2       2      3  7057 9810    9568   1762
## 3       2      3  6353 8808    7684   2405
## 4       1      3 13265 1196    4221   6404
## 5       2      3 22615 5410    7198   3915
## 6       2      3  9413 8259    5126    666
##   Detergents_Paper Delicassen cluster
## 1             2674     1338.0       2
## 2             3293     1776.0       2
## 3             3516      965.5       2
## 4              507     1788.0       1
## 5             1777      965.5       1
## 6             1795     1451.0       2
```

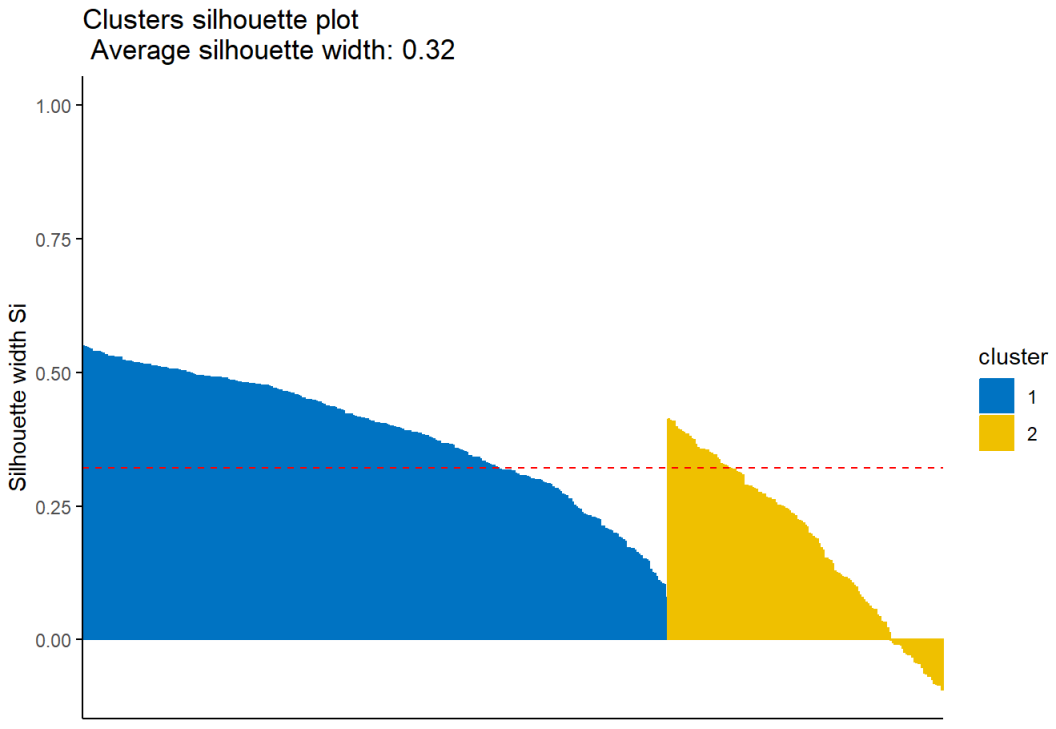
```
par(mfrow=c(1,1))
fviz_cluster(km.man, data=scale_df, palette=c("orange","lightblue"),ellipse.type="euclid", star.plot=TRUE,
             repel = TRUE, ggtheme=theme_minimal())
```

```
## Warning: ggrepel: 282 unlabeled data points
## (too many overlaps). Consider increasing
## max.overlaps
```




```
#INternal Validation
km.man.ec<-eclust(scale_df, "kmeans", k=2, hc_metric="manhattan", graph=FALSE)
fviz_silhouette(km.man.ec, palette="jco", ggtheme=theme_classic())
```

```
##   cluster size ave.sil.width
## 1       1  299          0.39
## 2       2  141          0.18
```




```
silinfo.km.man<-km.man.ec$silinfo
silinfo.km.man$avg.width
```

```
## [1] 0.320847
```

```
silinfo.km.man$clus.avg.widths
```

```
## [1] 0.3864921 0.1816423
```

```
neg_sil.km.man<-which(silinfo.km.man$widths[,"sil_width"]<0)
silinfo.km.man$widths[neg_sil.km.man,,drop=FALSE]
```

```
##     cluster neighbor     sil_width
## 188       2        1 -0.0004688525
## 374       2        1 -0.0056451747
## 137       2        1 -0.0079474045
## 31        2        1 -0.0086474081
## 142       2        1 -0.0087814720
## 282       2        1 -0.0092260454
## 208       2        1 -0.0159607639
## 285       2        1 -0.0230623183
## 184       2        1 -0.0263769074
## 232       2        1 -0.0275468724
## 44        2        1 -0.0284734105
## 383       2        1 -0.0321639652
## 301       2        1 -0.0413561812
## 63        2        1 -0.0433440850
## 227       2        1 -0.0441721899
## 203       2        1 -0.0443391842
## 6         2        1 -0.0520313973
## 305       2        1 -0.0623330505
## 173       2        1 -0.0641971302
## 97        2        1 -0.0679313972
## 425       2        1 -0.0680157559
## 346       2        1 -0.0747209452
## 138       2        1 -0.0821046651
## 61        2        1 -0.0833909417
## 304       2        1 -0.0844457955
## 26        2        1 -0.0855312554
## 273       2        1 -0.0935915849
```

```
k.link2<-cluster.stats(D_man, km.man.ec$cluster)
k.link2$dunn
```

```
## [1] 0.05669353
```

```
#External validation for k=2

table(data$Channel, km.man.ec$cluster)
```

```
##
##      1   2
##   1 267  31
##   2  32 110
```

```
#In this case there' s no perfect agreement between the external information and the cluster structure

tipes<-as.numeric(data$Channel)
clust_stats.km.man<-cluster.stats(d=D_man, tipes, km.man$cluster)
clust_stats.km.man$corrected.rand
```

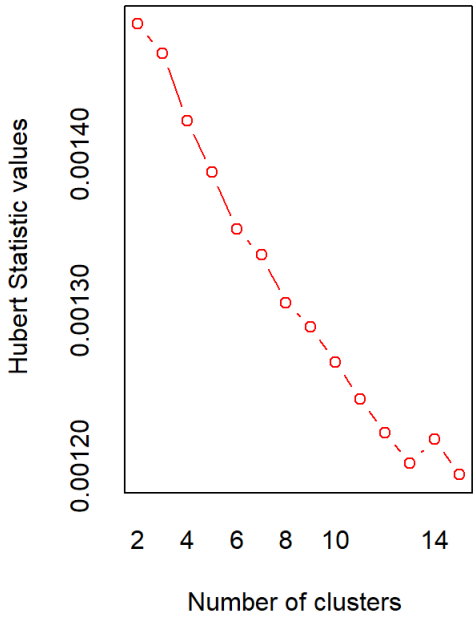
```
## [1] 0.5003106
```

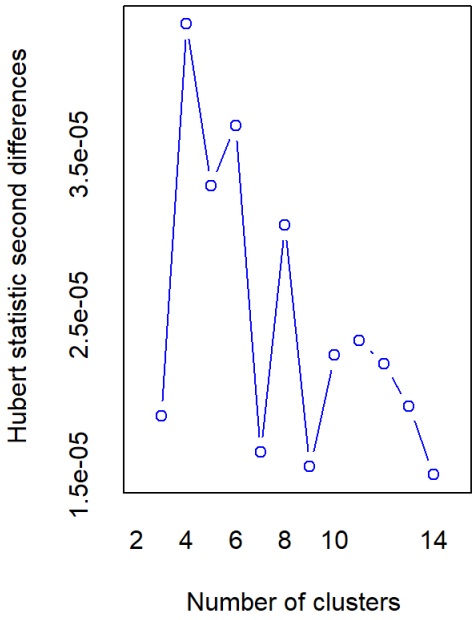
```
#According to the correct Rand index there is a good agreement between the seed types and the cluster
#solution
clust_stats.km.man$vi
```

```
## [1] 0.7983143
```

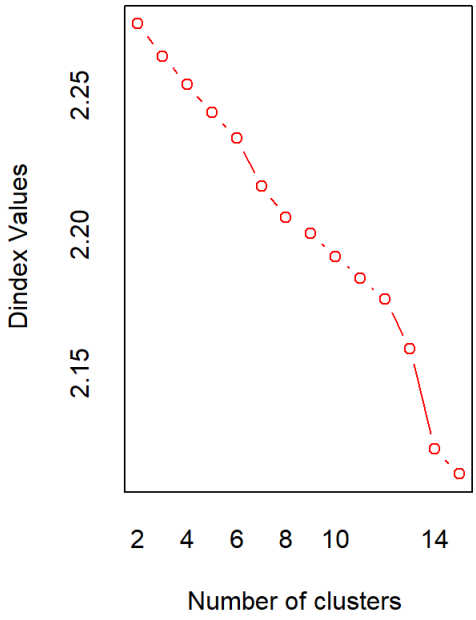
```
#PAM Method (Euclidean distance)
nb9<-NbClust(scale_df, distance="euclidean", method="centroid")
```

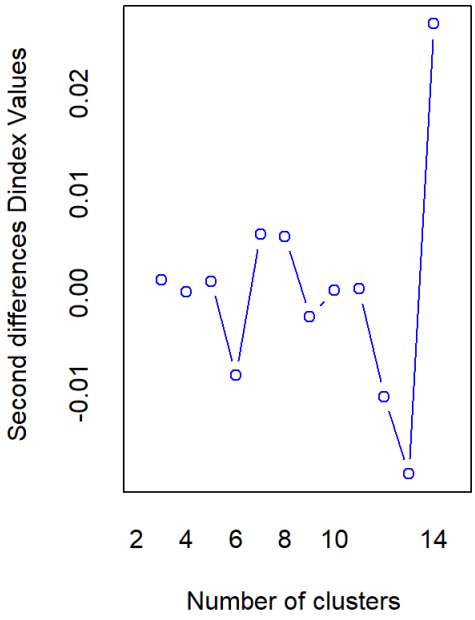
```
## Warning in pf(beale, pp, df2): Si è prodotto
## un NaN
```






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



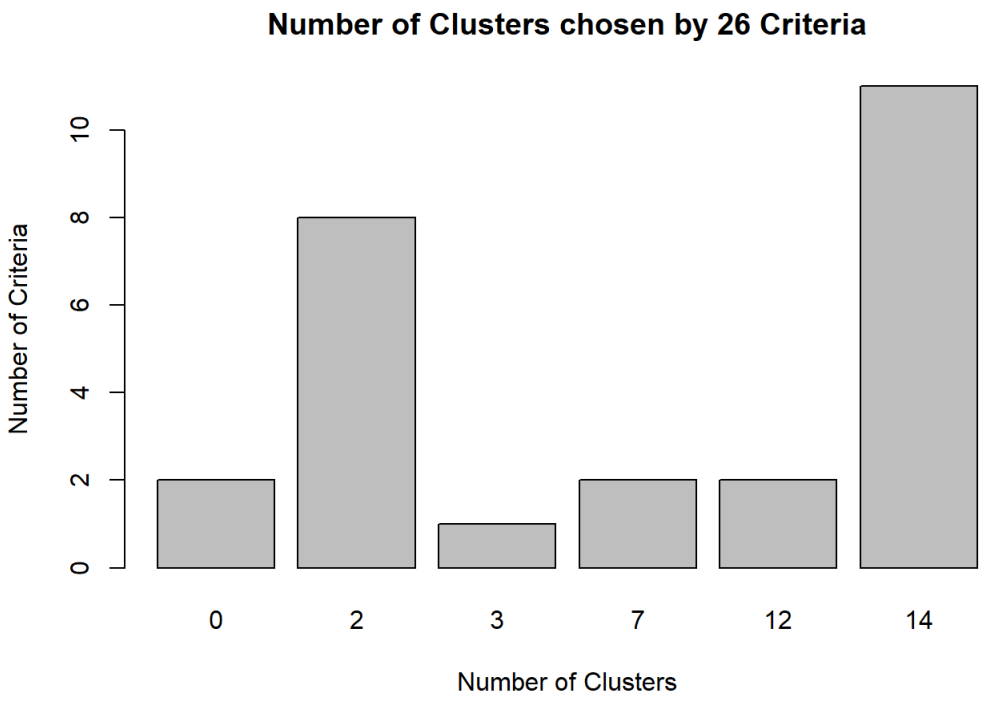


```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 8 proposed 2 as the best number of clusters
## * 1 proposed 3 as the best number of clusters
## * 2 proposed 7 as the best number of clusters
## * 2 proposed 12 as the best number of clusters
## * 11 proposed 14 as the best number of clusters
##
##                   ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  14
##
##
## *******************************************************************
```

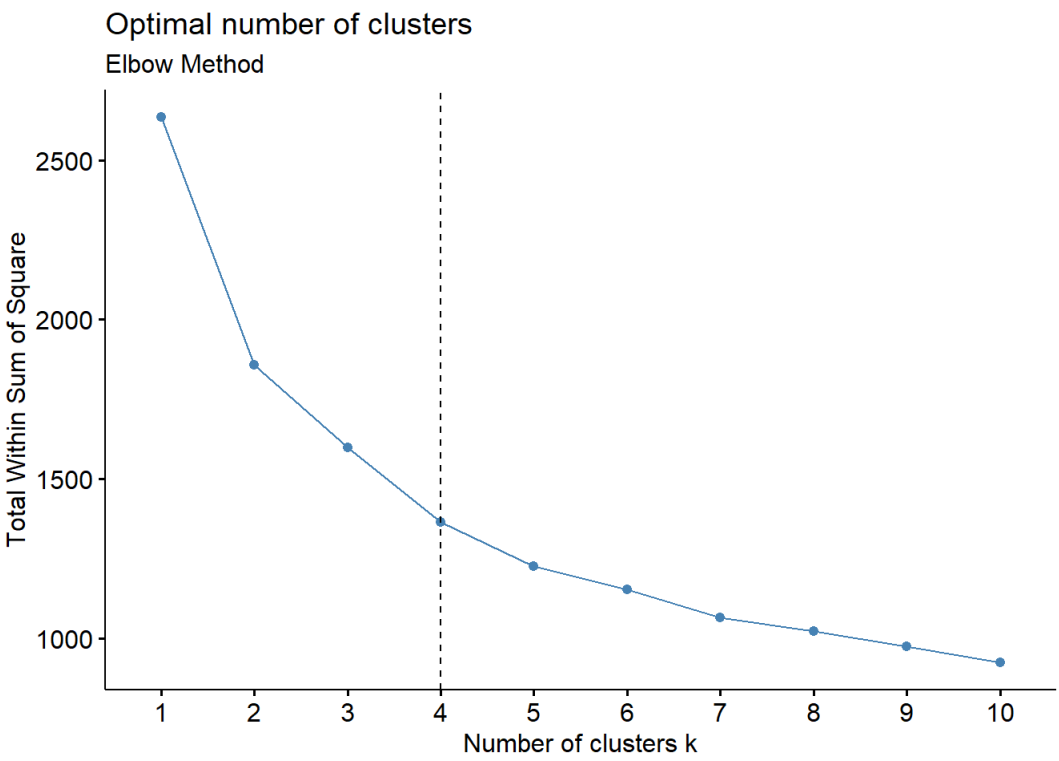
```
summary(nb9)
```

```
##                  Length Class  Mode
## All.index          364    -none- numeric
## All.CriticalValues  42    -none- numeric
## Best.nc             52    -none- numeric
## Best.partition     440    -none- numeric
```

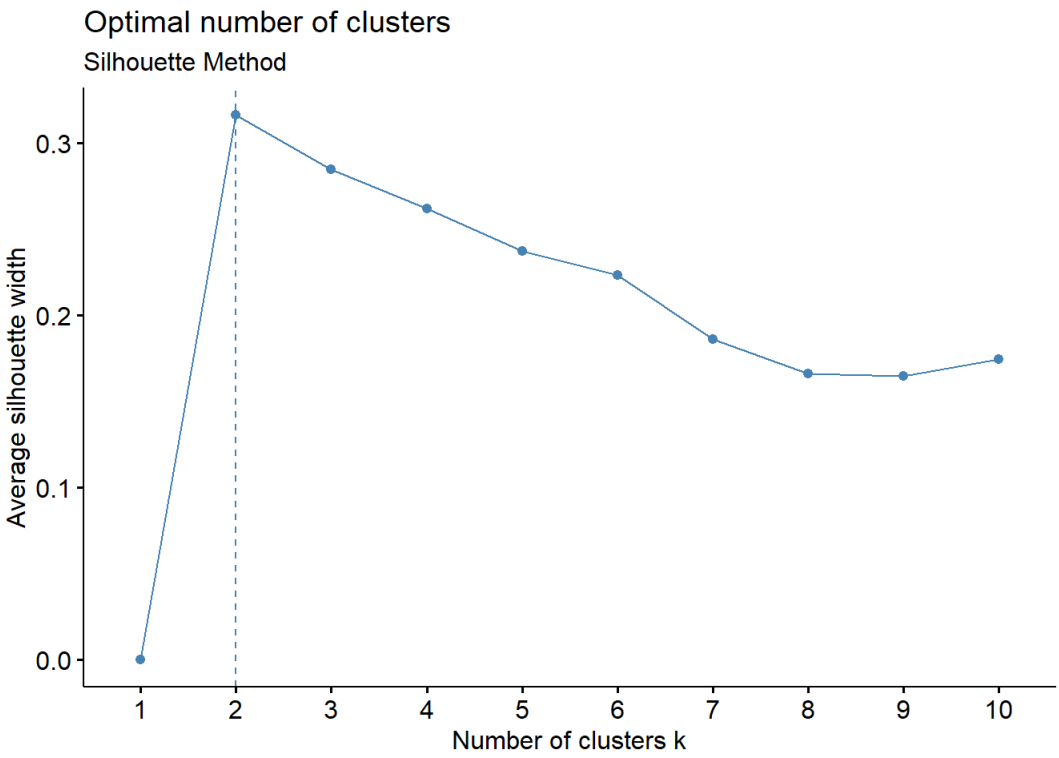
```
t9 <- table(nb9$Best.nc[1,])

par(mfrow= c(1,1))
barplot(t9,xlab="Number of Clusters",ylab="Number of Criteria",
        main="Number of Clusters chosen by 26 Criteria")
```




```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, cluster::pam, diss=D_euc, metric="euclidean", method="wss")+geom_vline(xintercept = 4, linetype=2)+labs(subtitle = "Elbow Metho
d")
```

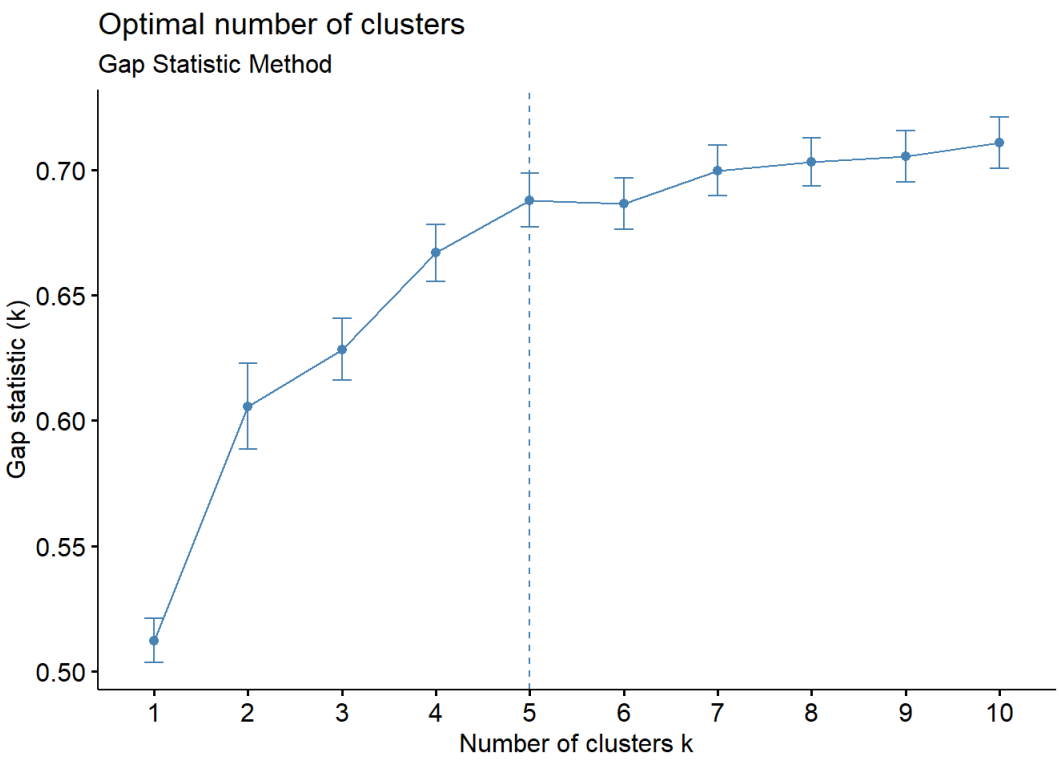



```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, cluster::pam,diss=D_euc, metric="euclidean", method="silhouette")+labs(subtitle = "Silhouette Method")
```




```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, cluster::pam, diss=D_euc, metric="euclidean", 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 sample]:
## ..... 50
## ..... 100
## ..... 150
## ..... 200
## ..... 250
## ..... 300
## ..... 350
## ..... 400
## ..... 450
## ..... 500
```



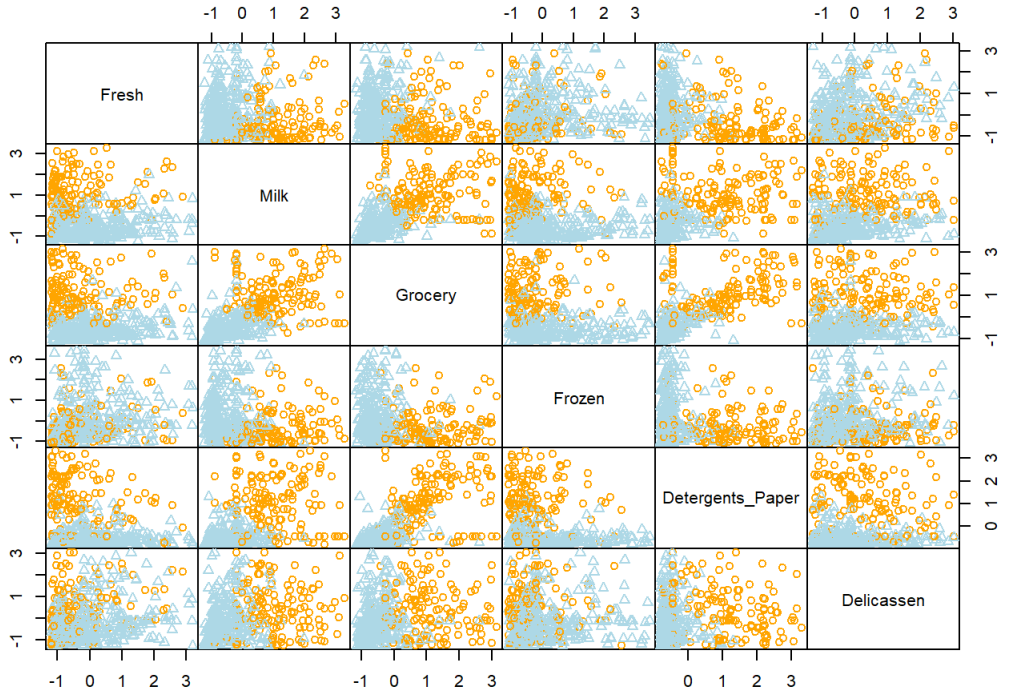

```
#according with this method, 2 clusters is the most suggested choice

set.seed(123)
pam.euc<-pam(scale_df,2, metric="euclidean")
print(pam.euc)
```

```
## Medoids:
##       ID       Fresh       Milk    Grocery
## [1,] 198 -0.90006960  0.8274121  0.7813425
## [2,] 338 -0.08071426 -0.8625813 -0.6756675
##          Frozen Detergents_Paper Delicassen
## [1,] -0.4616041         1.110714  0.4565594
## [2,] -0.2266431        -0.604088 -0.2573661
## Clustering vector:
##   [1] 1 1 1 2 2 1 2 1 2 1 1 2 1 1 1 2 1 2 1 2
##  [21] 2 2 2 1 1 2 2 2 1 2 1 2 2 2 2 1 2 1 1 2
##  [41] 2 2 1 1 1 1 1 2 1 2 2 2 2 1 2 2 2 1 2 1
##  [61] 1 2 1 1 2 2 2 1 1 2 2 1 2 2 1 2 2 1 2 2
##  [81] 2 1 1 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2
## [101] 1 1 1 1 2 2 1 1 1 2 2 1 2 2 2 2 2 2 2 2
## [121] 2 2 2 1 2 2 2 1 1 2 2 2 2 2 2 2 1 1 2 2
## [141] 2 1 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 2 1 1
## [161] 1 2 2 2 1 1 1 2 2 2 1 1 1 1 2 1 2 2 2 2
## [181] 1 1 1 2 2 2 2 1 1 1 2 2 2 1 2 2 2 1 2 2
## [201] 1 1 1 2 2 1 2 1 1 1 2 2 2 1 1 1 2 2 1 2
## [221] 2 1 2 2 2 2 1 2 2 2 2 1 2 2 2 2 2 2 2 2
## [241] 2 2 2 2 2 1 1 2 2 2 2 2 2 2 1 2 2 2 2 2
## [261] 2 2 2 2 2 1 1 1 2 1 2 2 2 1 2 2 2 2 2 2 1
## [281] 2 1 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 2 2 1 2
## [301] 1 1 1 1 1 1 1 1 2 2 1 2 2 1 2 2 1 2 2 2 1
## [321] 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 2 2
## [341] 1 1 1 2 2 2 1 1 1 2 1 2 1 2 1 2 2 2 1 2 2
## [361] 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 1 2 2 2
## [381] 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2
## [401] 2 2 2 2 2 2 2 2 1 2 2 2 2 2 1 1 2 1 1 1 1 2
## [421] 1 1 2 2 2 2 2 1 2 2 2 2 1 2 2 2 2 2 1 2 1 2 2
## Objective function:
##    build     swap
## 2.087990 1.947473
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering"
##  [4] "objective"  "isolation"  "clusinfo"
##  [7] "silinfo"    "diss"       "call"
## [10] "data"
```

```
pam.euc$clusinfo
```

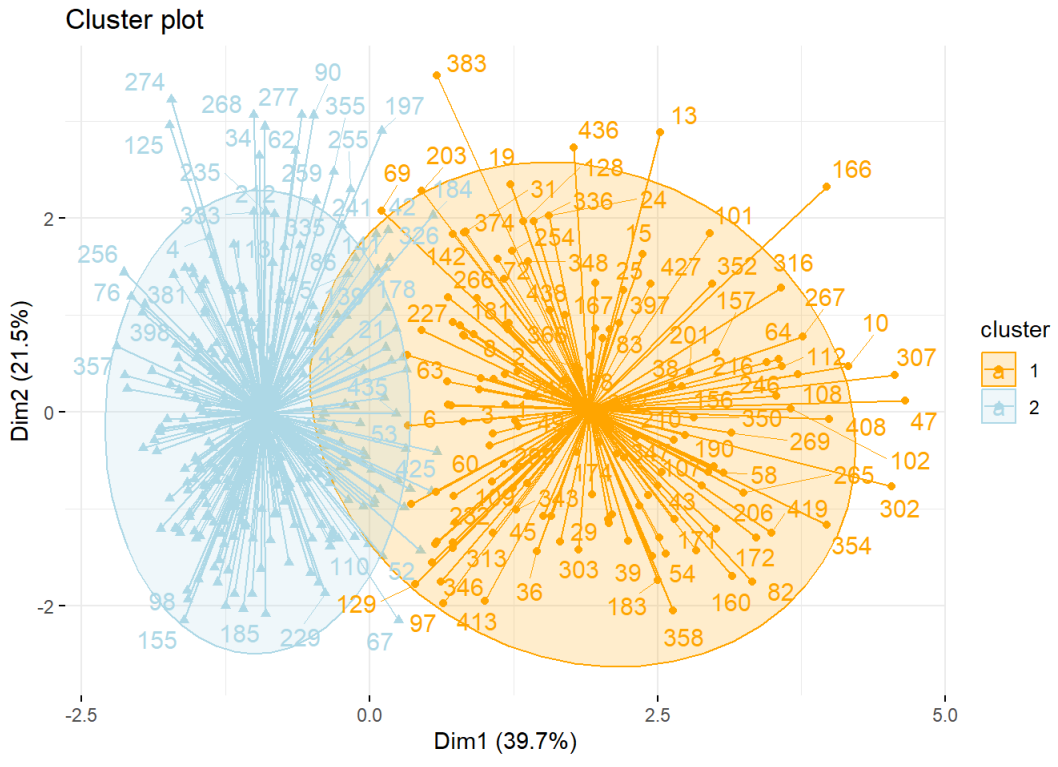
```
##      size max_diss  av_diss diameter
## [1,]  144 4.448941 2.331053 6.845932
## [2,]  296 4.676644 1.760867 6.241928
##      separation
## [1,]  0.4495802
## [2,]  0.4495802
```

```
pam.euc.cl<-pam.euc$clustering

par(mfrow=c(1,1))
pairs(scale_df, gap=0, pch=pam.euc.cl, col=c("orange","lightblue")[pam.euc.cl])
```



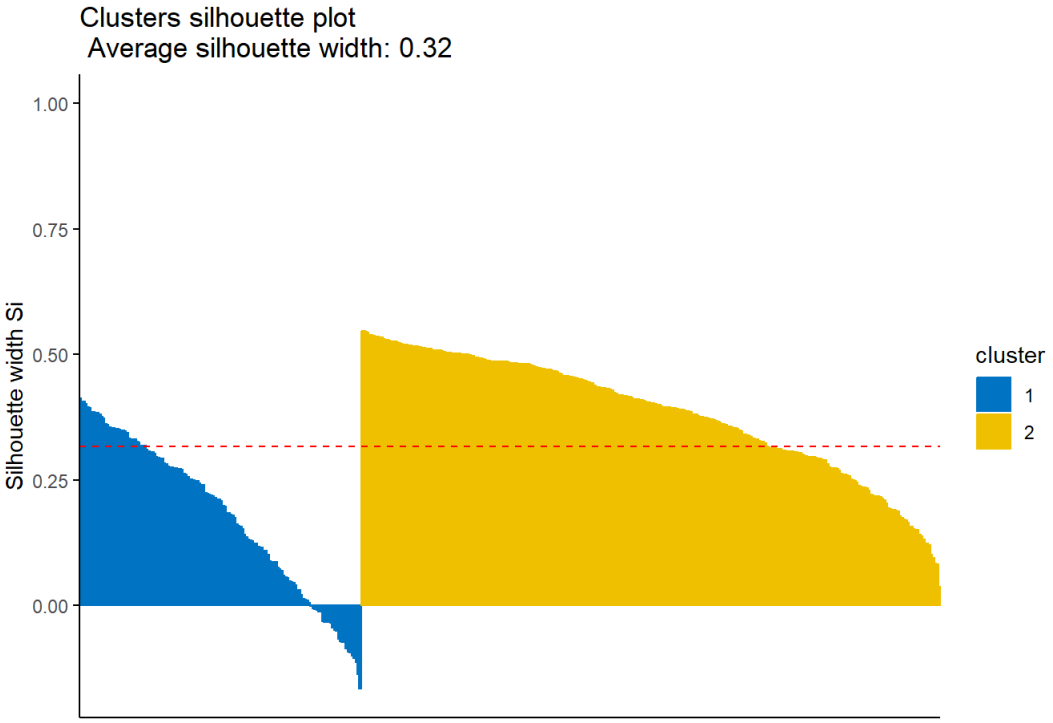

```
par(mfrow=c(1,1))
fviz_cluster(pam.euc, data=scale_df, palette=c("orange","lightblue"),ellipse.type="t", star.plot=TRUE,
             repel = TRUE, ggtheme=theme_minimal())
```

```
## Warning: ggrepel: 306 unlabeled data points
## (too many overlaps). Consider increasing
## max.overlaps
```



```
#INTERNAL VALIDATION
pam.euc.ec<-eclust(scale_df, "pam", k=2, k.max=14, graph=FALSE, hc_metric="euclidean")
fviz_silhouette(pam.euc, palette="jco", ggtheme=theme_classic())
```

```
##   cluster size ave.sil.width
## 1       1  144          0.18
## 2       2  296          0.38
```




```
silinfo.pam.euc<-pam.euc$silinfo
silinfo.pam.euc$avg.width
```

```
## [1] 0.3161328
```

```
silinfo.pam.euc$clus.avg.widths
```

```
## [1] 0.1764459 0.3840886
```

```
neg_sil.pam.euc<-which(silinfo.pam.euc$widths[,"sil_width"]<0)
silinfo.pam.euc$widths[neg_sil.pam.euc,,drop=FALSE]
```

```
##     cluster neighbor    sil_width
## 31        1        2 -0.001932810
## 285       1        2 -0.005208623
## 232       1        2 -0.008347689
## 142       1        2 -0.009818957
## 44        1        2 -0.012667465
## 208       1        2 -0.012814031
## 305       1        2 -0.031144962
## 301       1        2 -0.032390094
## 6         1        2 -0.033239000
## 383       1        2 -0.033731889
## 63        1        2 -0.033965070
## 203       1        2 -0.044976498
## 227       1        2 -0.048481027
## 173       1        2 -0.050845537
## 97        1        2 -0.066610021
## 273       1        2 -0.071529704
## 138       1        2 -0.073406775
## 346       1        2 -0.073736762
## 304       1        2 -0.086582486
## 61        1        2 -0.092357196
## 69        1        2 -0.093380979
## 129       1        2 -0.100450268
## 104       1        2 -0.104388582
## 154       1        2 -0.112567399
## 414       1        2 -0.136874823
## 209       1        2 -0.165363755
```

```
pam.euc.link<-cluster.stats(D_euc, pam.euc.ec$cluster)
pam.euc.link$dunn
```

```
## [1] 0.06567114
```

```
#EXTERNAL VALIDATION: confusion matrix, correct Rand index, Meila's VI index
table(data$Channel, pam.euc.ec$cluster)
```

```
##
##      1   2
##   1  36 262
##   2 108  34
```

```
#In this case there' s no perfect agreement between the external information and the cluster structure

tipes<-as.numeric(data$Channel)
clust_stats.pam.euc<-cluster.stats(d=D_euc, tipes, pam.euc$cluster)
clust_stats.pam.euc$corrected.rand
```

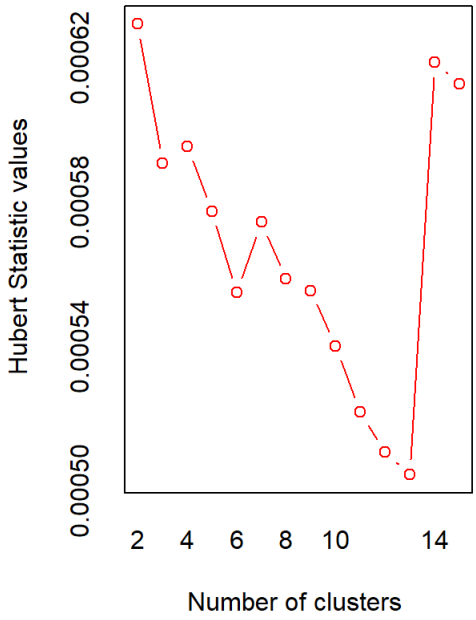
```
## [1] 0.4557572
```

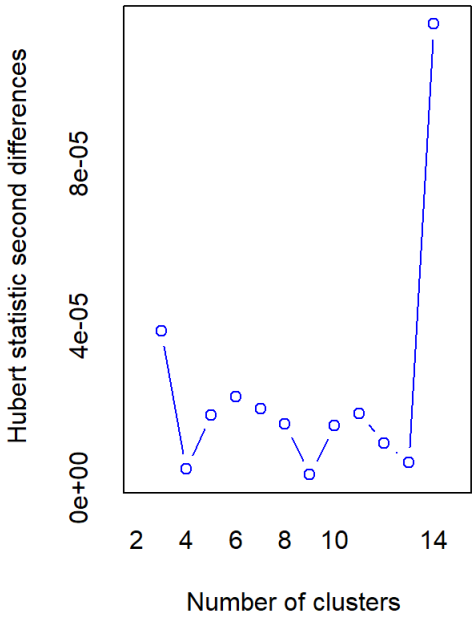
```
#According to the correct Rand index there is a good agreement between the seed types and the cluster
#solution
clust_stats.pam.euc$vi
```

```
## [1] 0.8511411
```

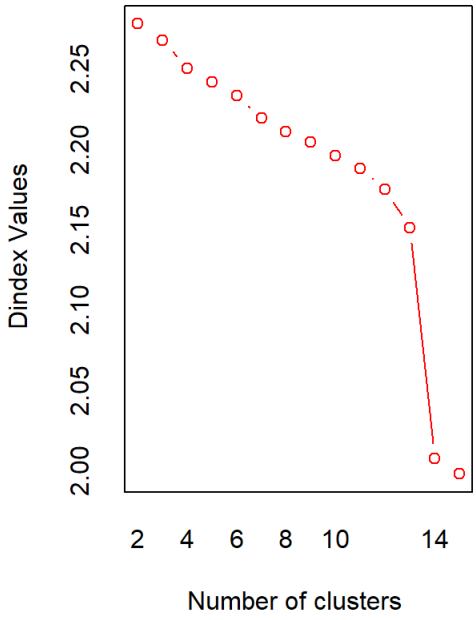
```
#PAM Method (Manhattan distance)
nb10<-NbClust(scale_df, distance="manhattan", method="centroid")
```

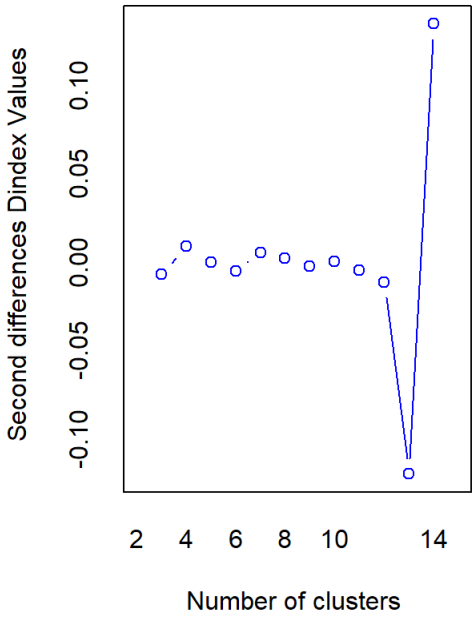
```
## Warning in pf(beale, pp, df2): Si è prodotto
## un NaN
```






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



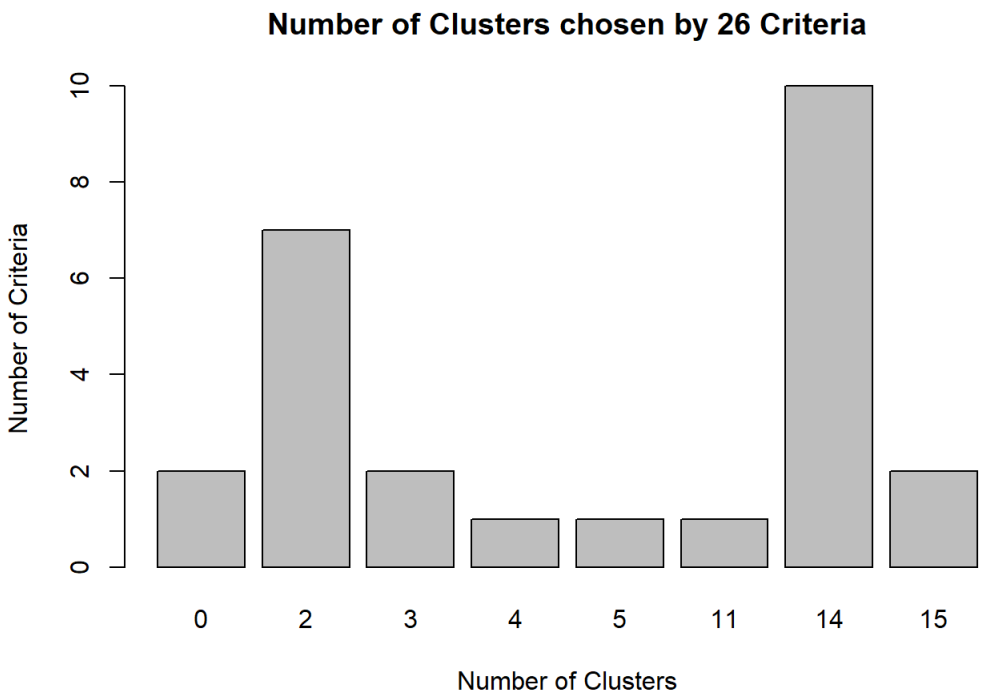


```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 7 proposed 2 as the best number of clusters
## * 2 proposed 3 as the best number of clusters
## * 1 proposed 4 as the best number of clusters
## * 1 proposed 5 as the best number of clusters
## * 1 proposed 11 as the best number of clusters
## * 10 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 is  14
##
##
## *******************************************************************
```

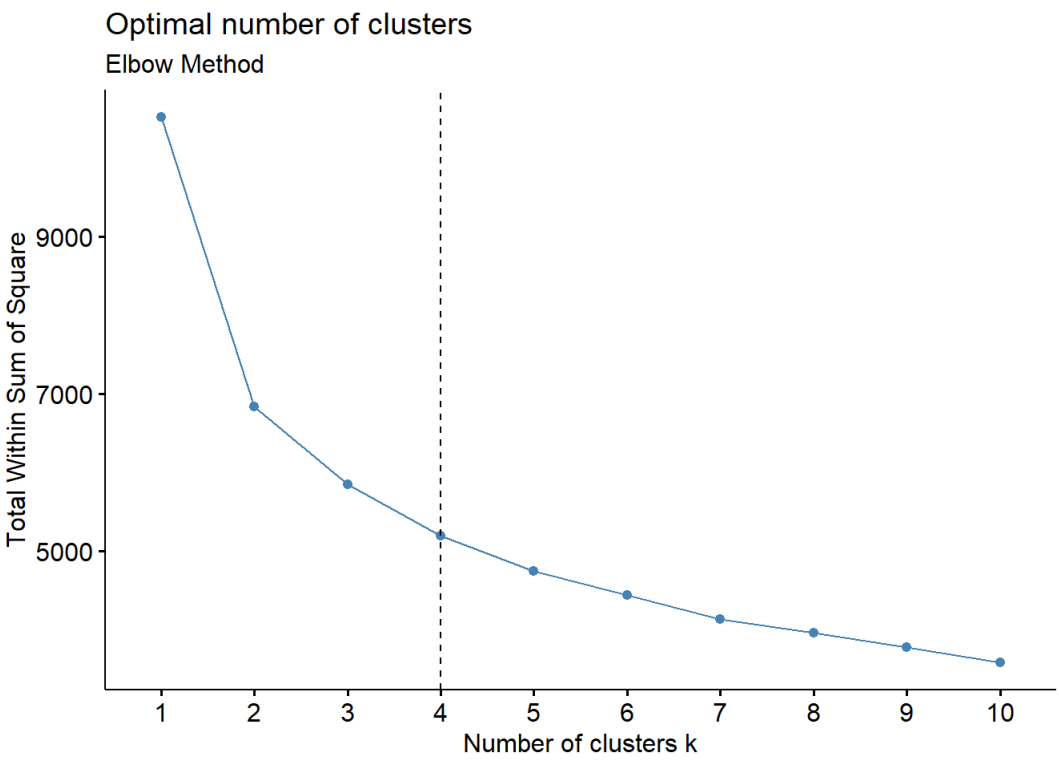
```
summary(nb10)
```

```
##                  Length Class  Mode
## All.index         364    -none- numeric
## All.CriticalValues  42    -none- numeric
## Best.nc            52    -none- numeric
## Best.partition     440    -none- numeric
```

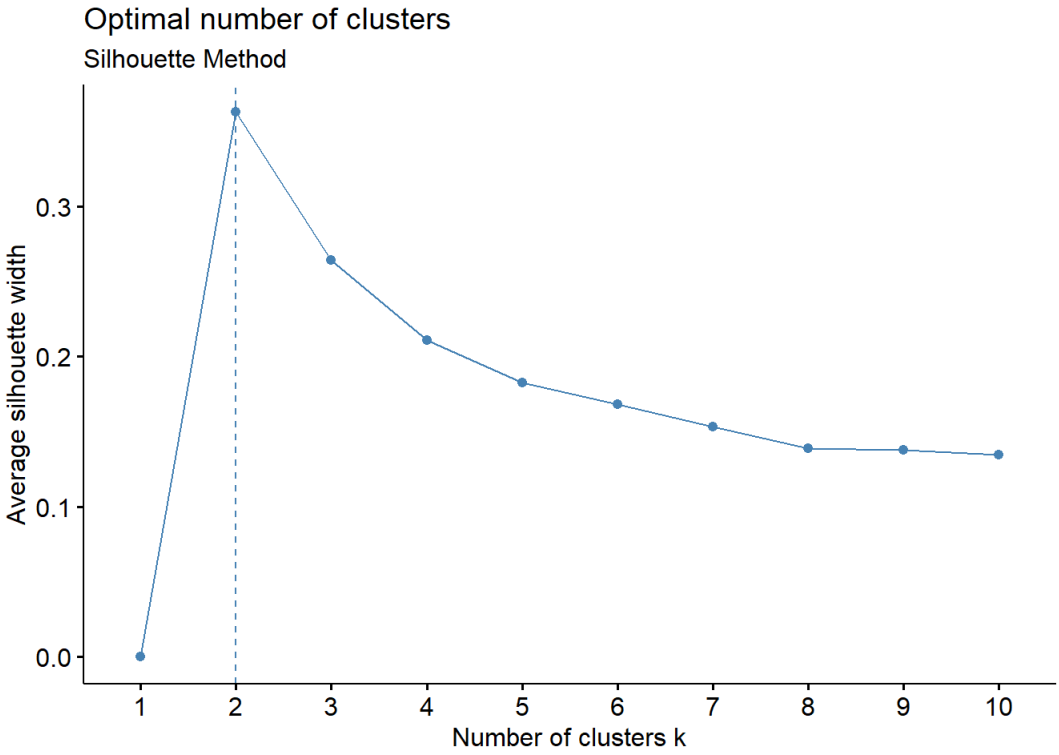
```
t10 <- table(nb10$Best.nc[1,])

par(mfrow= c(1,1))
barplot(t10,xlab="Number of Clusters",ylab="Number of Criteria",
        main="Number of Clusters chosen by 26 Criteria")
```




```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, cluster::pam, diss=D_man, metric="manhattan", method="wss")+geom_vline(xintercept = 4, linetype=2)+labs(subtitle = "Elbow Method")
```

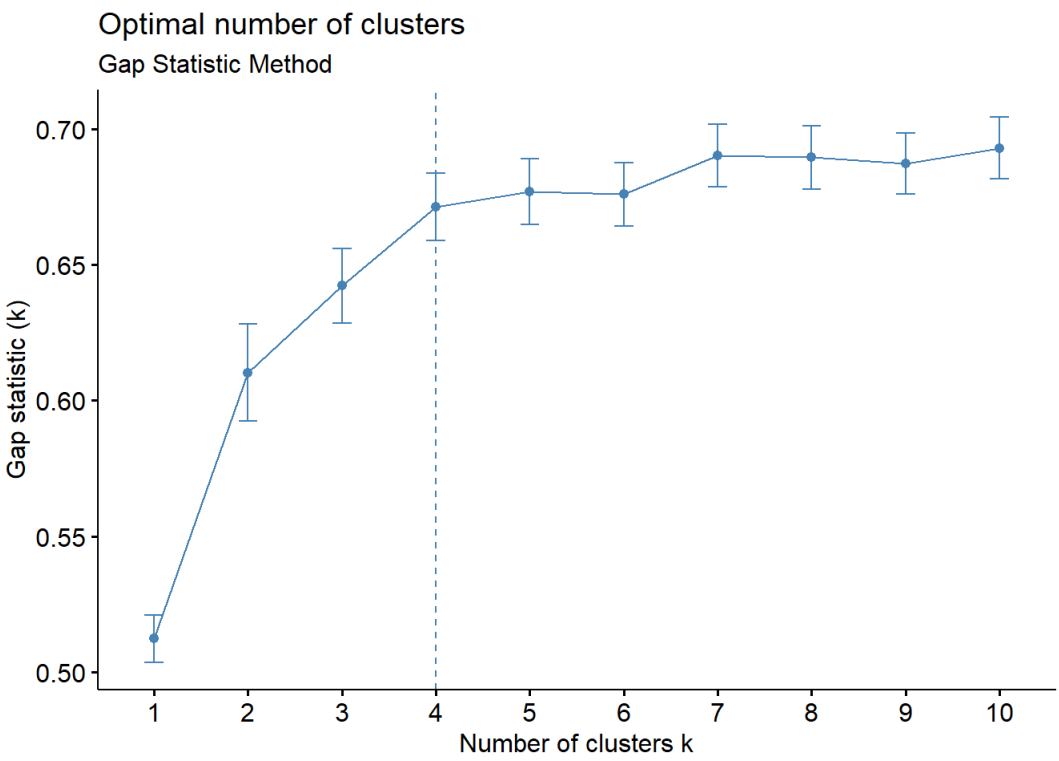



```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, cluster::pam,diss=D_man, metric="manhattan", method="silhouette")+labs(subtitle = "Silhouette Method")
```




```
par(mfrow= c(1,1))
fviz_nbclust(scale_df, cluster::pam, diss=D_man, metric="manhattan", 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 sample]:
## ..... 50
## ..... 100
## ..... 150
## ..... 200
## ..... 250
## ..... 300
## ..... 350
## ..... 400
## ..... 450
## ..... 500
```



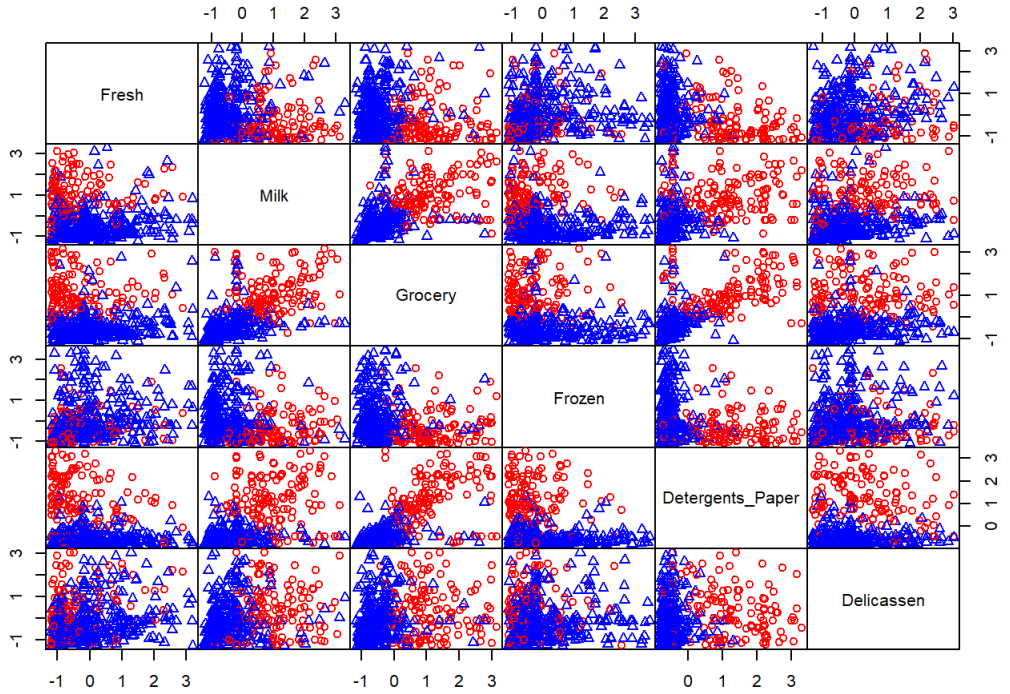

```
#according with this method, 2 clusters is the most suggested choice

set.seed(123)
pam.man<-pam(scale_df,2, metric="manhattan")
print(pam.man)
```

```
## Medoids:
##       ID       Fresh       Milk    Grocery
## [1,] 198 -0.90006960  0.8274121  0.7813425
## [2,] 338 -0.08071426 -0.8625813 -0.6756675
##          Frozen Detergents_Paper Delicassen
## [1,] -0.4616041         1.110714  0.4565594
## [2,] -0.2266431        -0.604088 -0.2573661
## Clustering vector:
##   [1] 1 1 1 2 2 1 2 1 2 1 1 2 1 1 1 2 1 2 1 2
##  [21] 2 2 2 1 1 1 2 2 1 2 1 2 2 2 2 1 2 1 1 2
##  [41] 2 2 1 2 1 1 1 2 1 2 2 2 2 1 2 2 2 1 2 1
##  [61] 1 2 1 1 2 2 1 1 1 2 2 2 2 2 1 2 2 2 2 2
##  [81] 2 1 1 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2
## [101] 1 1 1 2 2 2 1 1 1 2 2 1 2 2 2 2 2 2 2 2
## [121] 2 2 2 1 2 2 2 1 2 2 2 2 2 2 2 2 1 1 2 2
## [141] 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 2 1 1
## [161] 1 2 2 2 1 1 1 2 2 2 1 1 1 1 2 1 2 2 2 2
## [181] 1 1 1 2 2 2 2 1 1 1 2 2 2 1 2 2 2 1 2 2
## [201] 1 1 2 2 2 1 2 1 1 1 2 2 2 1 1 1 2 2 1 2
## [221] 2 1 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2
## [241] 2 2 2 2 2 1 1 2 2 2 2 2 2 2 1 2 2 2 2 2
## [261] 2 2 2 2 2 1 1 1 2 1 2 2 2 1 2 2 2 2 2 1
## [281] 2 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 1 2
## [301] 1 1 1 1 1 1 1 1 2 2 1 2 2 2 2 2 2 1 2 2 2
## [321] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2
## [341] 1 1 1 2 2 1 1 1 2 1 2 1 2 1 2 2 2 1 2 2
## [361] 2 2 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 1 2 2 1
## [381] 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2
## [401] 2 2 2 2 2 2 2 1 2 2 2 2 1 2 2 1 1 1 1 2
## [421] 1 1 2 2 1 2 1 2 2 2 1 2 2 2 2 1 2 2 2 2
## Objective function:
##   build    swap
## 3.92743 3.64284
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering"
##  [4] "objective"  "isolation"  "clusinfo"
##  [7] "silinfo"    "diss"       "call"
## [10] "data"
```

```
pam.man$clusinfo
```

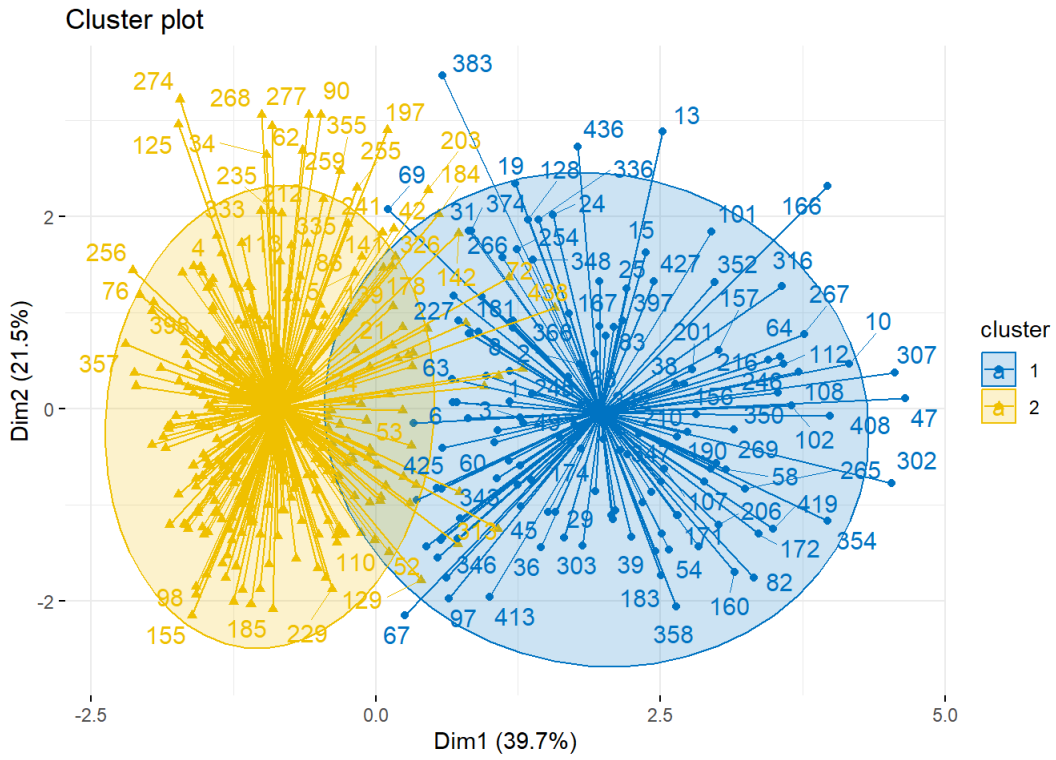
```
##      size max_diss  av_diss diameter
## [1,]  134 9.230000 4.467785 15.42502
## [2,]  306 8.567188 3.281590 13.29917
##      separation
## [1,]   1.428449
## [2,]   1.428449
```

```
pam.man.cl<-pam.man$clustering

par(mfrow=c(1,1))
pairs(scale_df, gap=0, pch=pam.man.cl, col=c("red","blue")[pam.man.cl])
```



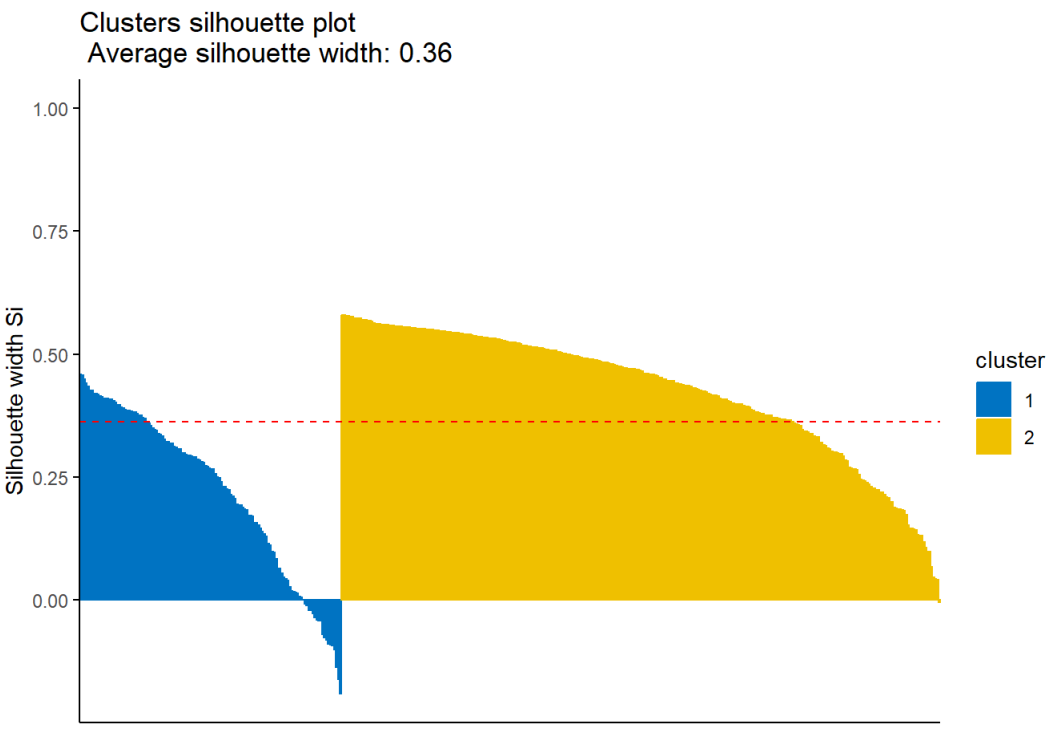

```
par(mfrow=c(1,1))
fviz_cluster(pam.man, data=scale_df, palette="jco",ellipse.type="t", star.plot=TRUE,
             repel = TRUE, ggtheme=theme_minimal())
```

```
## Warning: ggrepel: 312 unlabeled data points
## (too many overlaps). Consider increasing
## max.overlaps
```



```
#INTERNAL VALIDATION
pam.man.ec<-eclust(scale_df, "pam", k=2, k.max=14, graph=FALSE, hc_metric="manhattan")
fviz_silhouette(pam.man, palette="jco", ggtheme=theme_classic())
```

```
##   cluster size ave.sil.width
## 1       1  134         0.22
## 2       2  306         0.43
```




```
silinfo.pam.man<-pam.man$silinfo
silinfo.pam.man$avg.width
```

```
## [1] 0.3625634
```

```
silinfo.pam.man$clus.avg.widths
```

```
## [1] 0.2188210 0.4255094
```

```
neg_sil.pam.man<-which(silinfo.pam.man$widths[,"sil_width"]<0)
silinfo.pam.man$widths[neg_sil.pam.man,,drop=FALSE]
```

```
##     cluster neighbor    sil_width
## 301       1        2 -0.008373414
## 383       1        2 -0.010872471
## 31        1        2 -0.020903168
## 266       1        2 -0.021757332
## 374       1        2 -0.027234916
## 26        1        2 -0.036319738
## 425       1        2 -0.041361922
## 138       1        2 -0.042102205
## 61        1        2 -0.043060946
## 6         1        2 -0.069794715
## 305       1        2 -0.076009891
## 273       1        2 -0.081275270
## 304       1        2 -0.089776631
## 380       1        2 -0.091379341
## 202       1        2 -0.093476522
## 67        1        2 -0.100799587
## 209       1        2 -0.136725509
## 69        1        2 -0.161222717
## 154       1        2 -0.190469499
## 72        2        1 -0.005383341
```

```
pam.man.link<-cluster.stats(D_man, pam.man.ec$cluster)
pam.man.link$dunn
```

```
## [1] 0.05669353
```

```
#EXTERNAL VALIDATION: confusion matrix, correct Rand index, Meila's VI index
table(data$Channel, pam.man.ec$cluster)
```

```
##
##      1   2
##   1  36 262
##   2 108  34
```

```
#In this case there' s no perfect agreement between the external information and the cluster structure

tipes<-as.numeric(data$Channel)
clust_stats.pam.man<-cluster.stats(d=D_man, tipes, pam.man$cluster)
clust_stats.pam.man$corrected.rand
```

```
## [1] 0.4791157
```

```
#According to the correct Rand index there is a good agreement between the seed types and the cluster
#solution
clust_stats.pam.man$vi
```

```
## [1] 0.8167986
```

```
## Model Based Clustering
#Find the best parsimonious configuration of the Gaussian mixture distribution and the most appropriate
#number of K components.
model<-Mclust(scale_df, G=1:9, modelNames=NULL)
```

```
## fitting ...
##
  |
  |                      |   0%
  |
  |                      |   1%
  |
  |=                     |   2%
  |
  |=                     |   3%
  |
  |=                     |   4%
  |
  |==                    |   5%
  |
  |==                    |   6%
  |
  |==                    |   7%
  |
  |===                   |   8%
  |
  |===                   |   9%
  |
  |====                  |  10%
  |
  |====                  |  11%
  |
  |====                  |  12%
  |
  |====                  |  13%
  |
  |=====                 |  13%
  |
  |=====                 |  14%
  |
  |=====                 |  15%
  |
  |======                |  16%
  |
  |======                |  17%
  |
  |======                |  18%
  |
```

|======= | 19%
|
|======= | 20%
|
|======= | 21%
|
|======== | 22%
|
|======== | 23%
|
|======== | 24%
|
|========= | 24%
|
|========= | 25%
|
|========= | 26%
|
|========= | 27%
|
|========== | 28%
|
|========== | 29%
|
|========== | 30%
|
|=========== | 31%
|
|=========== | 32%
|
|============ | 33%
|
|============ | 34%
|
|============ | 35%
|
|============= | 36%
|
|============= | 37%
|
|============= | 38%
|
|============== | 39%
|
|============== | 40%
|
|============== | 41%
|
|=============== | 42%
|
|=============== | 43%
|
|=============== | 44%
|
|================ | 45%
|
|================ | 46%
|
|================= | 47%
|
|================= | 48%
|
|================= | 49%
|
|================= | 50%
|
|================== | 50%
|
|================== | 51%
|
|================== | 52%
|
|================== | 53%
|
|=================== | 54%
|
|=================== | 55%
|

|========== | 56%
|
|========== | 57%
|
|========== | 58%
|
|=========== | 59%
|
|=========== | 60%
|
|=========== | 61%
|
|============ | 62%
|
|============ | 63%
|
|============ | 64%
|
|============ | 65%
|
|============ | 66%
|
|============ | 67%
|
|============= | 68%
|
|============= | 69%
|
|============= | 70%
|
|============= | 71%
|
|============= | 72%
|
|============== | 73%
|
|============== | 74%
|
|============== | 75%
|
|============== | 76%
|
|=============== | 76%
|
|=============== | 77%
|
|=============== | 78%
|
|================ | 79%
|
|================ | 80%
|
|================ | 81%
|
|================= | 82%
|
|================= | 83%
|
|================= | 84%
|
|================== | 85%
|
|================== | 86%
|
|================== | 87%
|
|================== | 87%
|
|================== | 88%
|
|================== | 89%
|
|================== | 90%
|
|=================== | 91%
|
|=================== | 92%
|

```
|================================= | 93%
|
|================================= | 94%
|
|================================= | 95%
|
|================================== | 96%
|
|================================== | 97%
|
|================================== | 98%
|
|===================================| 99%
|
|===================================| 100%
```

```
summary(model)
```

```
## ----------------------------------------------
## Gaussian finite mixture model fitted by EM
## algorithm
## ----------------------------------------------
##
## Mclust VVI (diagonal, varying volume and
## shape) model with 7 components:
##
##  log-likelihood   n df       BIC       ICL
##       -2639.456 440 90 -5826.722 -5968.061
##
## Clustering table:
##  1  2  3  4  5  6  7
## 52 45 60 73 51 91 68
```

```
#soft assignment
head(round(model$z, 6), 30)
```

```
##           [,1]     [,2]     [,3]     [,4]
##  [1,] 0.560121 0.000000 0.000001 0.438575
##  [2,] 0.910448 0.000000 0.007842 0.081711
##  [3,] 0.639089 0.000000 0.000137 0.360774
##  [4,] 0.000022 0.806541 0.000000 0.028209
##  [5,] 0.012477 0.000000 0.000000 0.987523
##  [6,] 0.015014 0.000000 0.000246 0.158544
##  [7,] 0.020321 0.000000 0.000001 0.979622
##  [8,] 0.077085 0.000000 0.001020 0.921895
##  [9,] 0.003001 0.011037 0.055646 0.346657
## [10,] 0.795685 0.000000 0.204315 0.000000
## [11,] 0.995084 0.000000 0.000000 0.004916
## [12,] 0.000060 0.108758 0.000000 0.007390
## [13,] 0.999210 0.000000 0.000000 0.000790
## [14,] 0.999960 0.000000 0.000000 0.000040
## [15,] 0.996634 0.000000 0.000000 0.003366
## [16,] 0.000024 0.138404 0.000000 0.003977
## [17,] 0.018093 0.000000 0.981776 0.000131
## [18,] 0.000143 0.016469 0.001322 0.007001
## [19,] 0.054668 0.000000 0.000000 0.945332
## [20,] 0.048017 0.000000 0.051606 0.900313
## [21,] 0.005610 0.000000 0.000000 0.977741
## [22,] 0.000002 0.145662 0.000000 0.000149
## [23,] 0.007771 0.000000 0.000000 0.917727
## [24,] 1.000000 0.000000 0.000000 0.000000
## [25,] 0.999467 0.000000 0.000000 0.000533
## [26,] 0.052406 0.000000 0.000000 0.947594
## [27,] 0.000003 0.068668 0.000000 0.000321
## [28,] 0.000001 0.001045 0.000000 0.000142
## [29,] 0.021179 0.000000 0.978817 0.000004
## [30,] 0.000044 0.382094 0.000004 0.006140
##           [,5]     [,6]     [,7]
##  [1,] 0.000000 0.000000 0.001303
##  [2,] 0.000000 0.000000 0.000000
##  [3,] 0.000000 0.000000 0.000000
##  [4,] 0.000000 0.165227 0.000000
##  [5,] 0.000000 0.000000 0.000000
##  [6,] 0.000000 0.000000 0.826195
##  [7,] 0.000000 0.000000 0.000057
##  [8,] 0.000000 0.000000 0.000000
##  [9,] 0.000000 0.000000 0.583660
## [10,] 0.000000 0.000000 0.000000
## [11,] 0.000000 0.000000 0.000000
## [12,] 0.097306 0.051464 0.735022
## [13,] 0.000000 0.000000 0.000000
## [14,] 0.000000 0.000000 0.000000
## [15,] 0.000000 0.000000 0.000000
## [16,] 0.000691 0.000350 0.856552
## [17,] 0.000000 0.000000 0.000000
## [18,] 0.000000 0.000027 0.975037
## [19,] 0.000000 0.000000 0.000000
## [20,] 0.000000 0.000000 0.000065
## [21,] 0.000000 0.000000 0.016648
## [22,] 0.038122 0.816064 0.000001
## [23,] 0.000000 0.000000 0.074502
## [24,] 0.000000 0.000000 0.000000
## [25,] 0.000000 0.000000 0.000000
## [26,] 0.000000 0.000000 0.000000
## [27,] 0.005805 0.925195 0.000008
## [28,] 0.814515 0.148864 0.035433
## [29,] 0.000000 0.000000 0.000000
## [30,] 0.000003 0.000175 0.611539
```

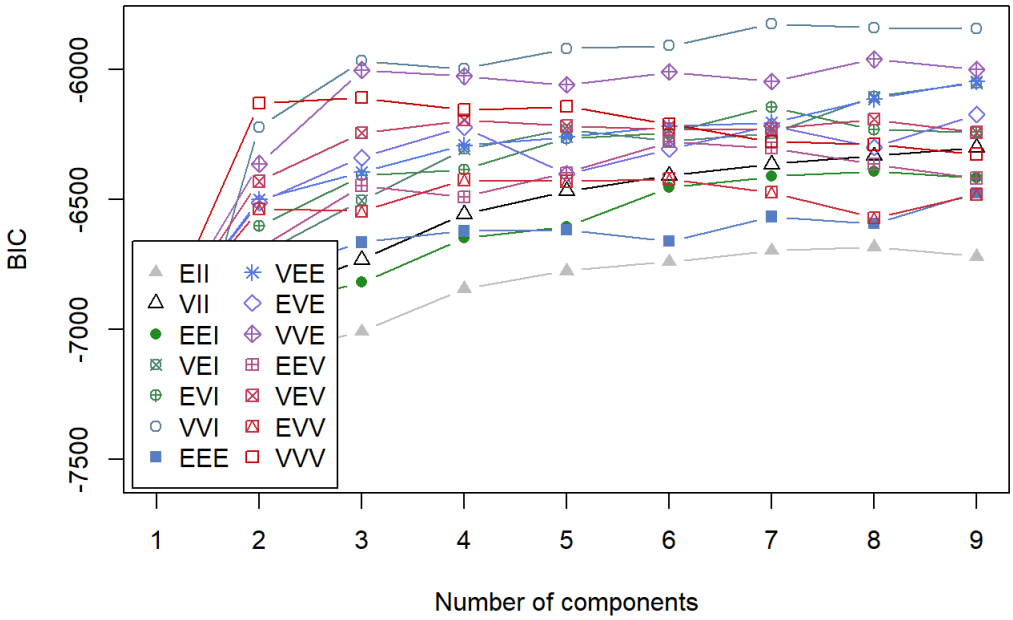
```
#hard assignment
head(model$classification, 30)
```

```
##  [1] 1 1 1 2 4 7 4 4 7 1 1 7 1 1 1 7 3 7 4 4
## [21] 4 6 4 1 1 4 6 5 3 7
```

```
#best BIC Value
summary(model$BIC)
```

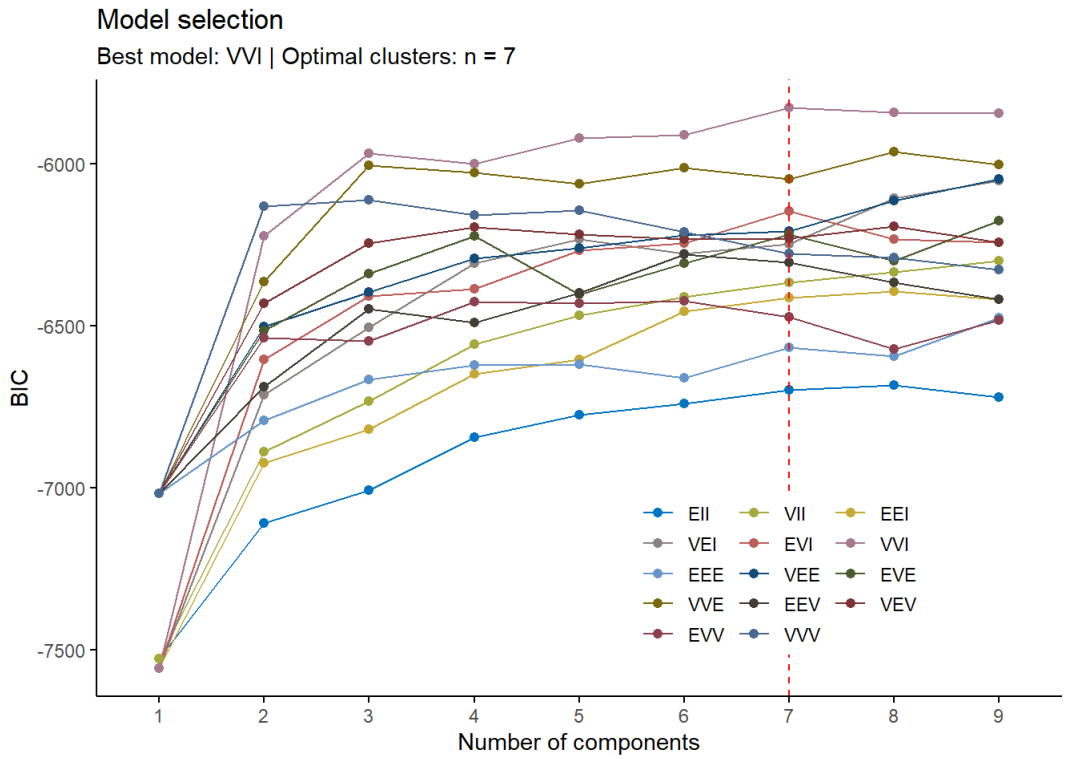
```
## Best BIC values:
##             VVI,7      VVI,8      VVI,9
## BIC      -5826.722 -5840.46456 -5844.55868
## BIC diff     0.000   -13.74227   -17.83639
```

```
par(mfrow=c(1,1))
plot(model, what="BIC", ylim=range(model$BIC, na.rm=TRUE), legendArgs=list(x="bottomleft"))
```

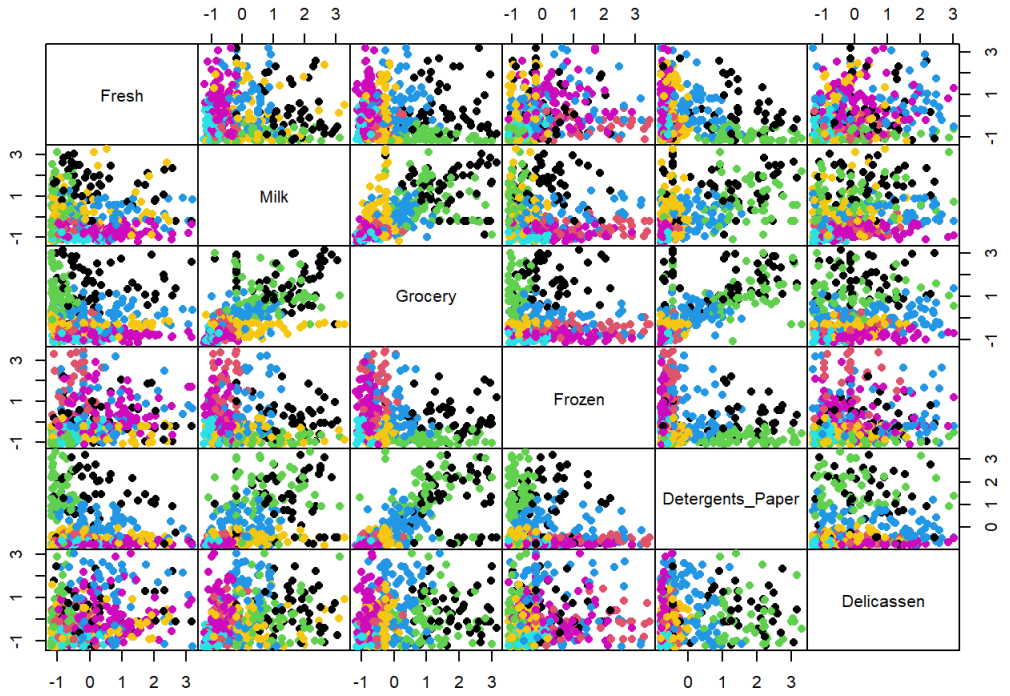



```
par(mfrow=c(1,1))
fviz_mclust(model, "BIC", palette="jco")
```

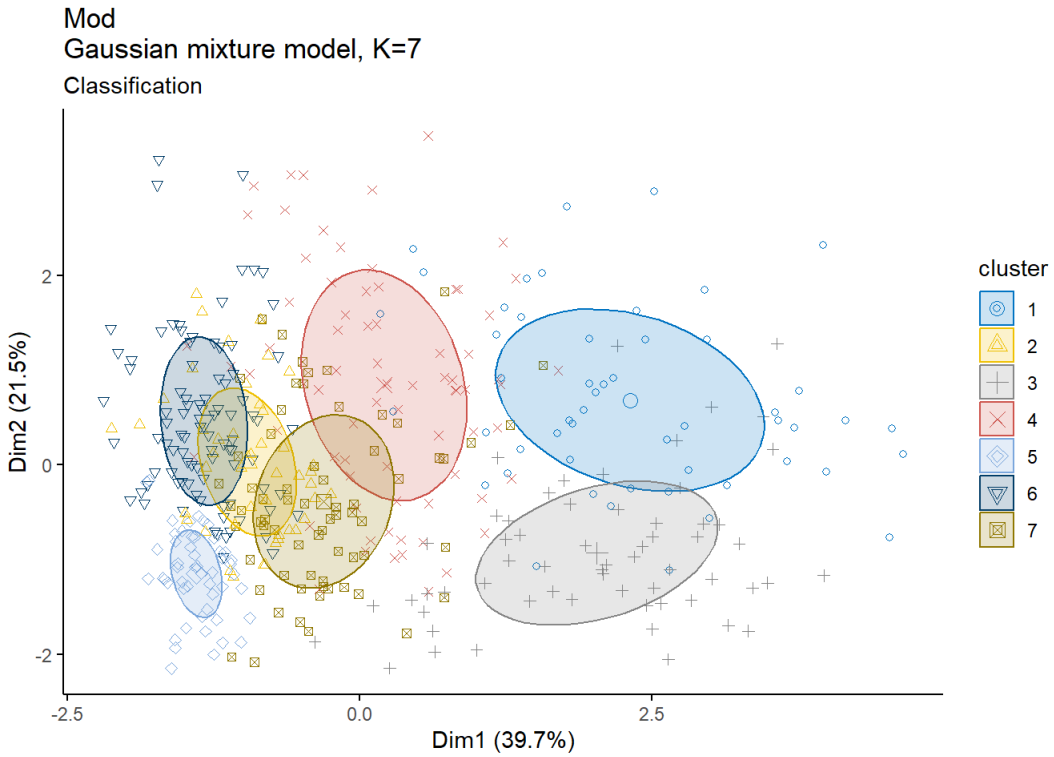
```
## Warning: `gather_()` was deprecated in tidyr 1.2.0.
## ℹ Please use `gather()` instead.
## ℹ The deprecated feature was likely used in
##   the factoextra package.
##   Please report the issue at
##   <]8;;https://github.com/kassambara/factoextra/issueshttps://github.com/kassambara/factoextra/issues]8;;>.
```



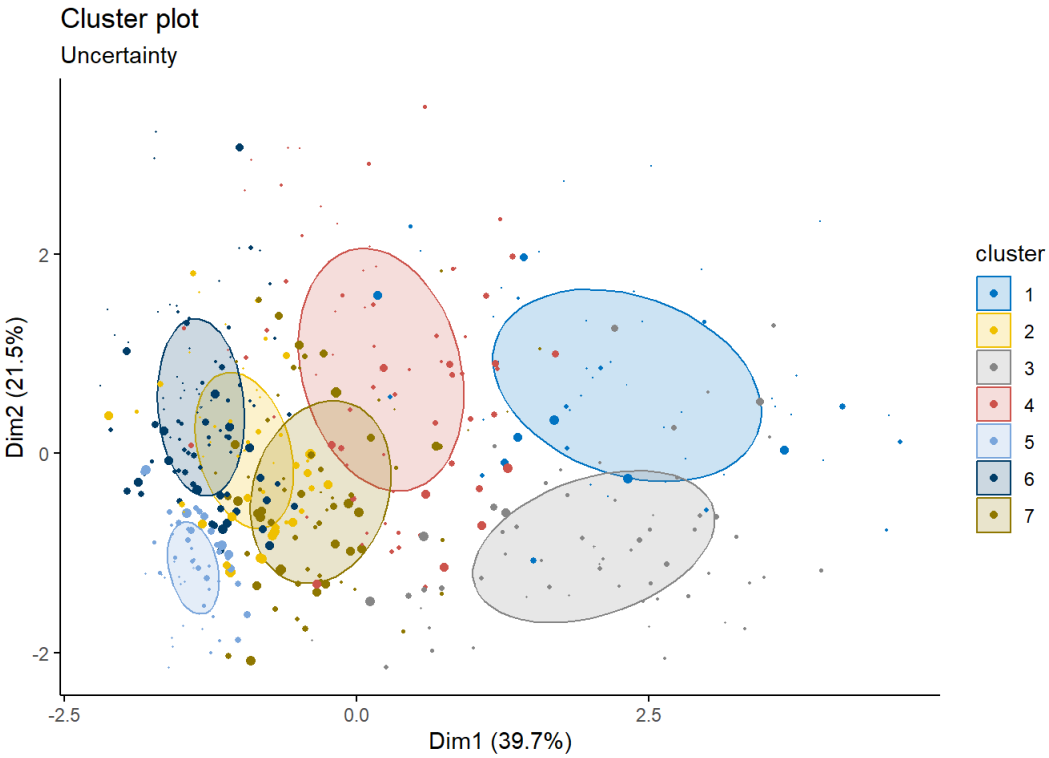
```
#According with the BIC criterion the best choice is a model with seven components (clusters) and the model is
#VVI: variable volume, variable shape and orentation.
#Plotting the classification:
par(mfrow=c(1,1))
pairs(scale_df, gap=0, pch=16, col=model$classification)
```




```
par(mfrow=c(1,1))
fviz_mclust(model, "classification", geom="point", pointsize=1.5, palette="jco", main="Mod
Gaussian mixture model, K=7")
```




```
par(mfrow=c(1,1))
fviz_mclust(model, "uncertainty", palette="jco")
```




```
#Larger points indicate the more uncertain observations.

#External validation
#Confusion matrix
table(data$Channel, model$classification)
```

```
##
##    1  2  3  4  5  6  7
##  1 10 38 16 41 51 90 52
##  2 42  7 44 32  0  1 16
```

```
#In this case there' s no perfect agreement between the external information and the cluster structure
#because there are four clusters and three category. Anyway we can see that:

#Corrected Rand index
adjustedRandIndex(data$Channel, model$classification)
```

```
## [1] 0.1016868
```

```
#According to the adjusted Rand index, the agreement between seeds type and the clusters obtained via
#model-based clustering, is poor. This is due to the fact that the number of clusters do not match the
#external labels.

#Best clustering method
#In order to chose the best model of clustering analysis among all the "clValid" is going to be used.
clmethod<-c("hierarchical","kmeans","pam")
rownames(scale_df)<-rownames(data)

#AVG
avg.euc_valid<-clValid(scale_df, nClust=2:7, clMethods=clmethod, validation=c("internal","stability"), maxitems=600,
metric="euclidean", method="average")
summary(avg.euc_valid)
```

```
##
## Clustering Methods:
##  hierarchical kmeans pam
##
## Cluster sizes:
##  2 3 4 5 6 7
##
## Validation Measures:
##                              2       3       4       5       6       7
##
## hierarchical APN            0.1078  0.0806  0.1147  0.1528  0.2259  0.2279
##              AD             3.2045  2.9890  2.8876  2.8078  2.7895  2.7016
##              ADM            0.5054  0.8084  0.7270  0.6590  0.8090  0.7003
##              FOM            0.9802  0.9764  0.9358  0.9308  0.9310  0.8923
##              Connectivity   28.5548 73.7091 79.6476 80.6238 82.7528 90.8544
##              Dunn           0.1682  0.1279  0.1334  0.1334  0.1334  0.1341
##              Silhouette     0.2551  0.2956  0.2824  0.2477  0.2448  0.2459
## kmeans       APN            0.0699  0.1914  0.3272  0.3084  0.3124  0.2617
##              AD             2.7403  2.6010  2.5683  2.4085  2.3396  2.2322
##              ADM            0.2227  0.5840  0.8863  0.8256  0.8515  0.7295
##              FOM            0.8854  0.8839  0.8854  0.8822  0.8791  0.8723
##              Connectivity   69.8119 116.2147 165.6528 166.5817 190.6087 204.3385
##              Dunn           0.0972  0.0640  0.0342  0.0360  0.0651  0.0577
##              Silhouette     0.3265  0.2594  0.2420  0.2596  0.2606  0.2539
## pam          APN            0.1271  0.2788  0.2428  0.3918  0.4703  0.4459
##              AD             2.7855  2.6700  2.4827  2.4452  2.4130  2.3098
##              ADM            0.4078  0.7481  0.6630  0.9331  1.0526  0.9708
##              FOM            0.9056  0.8932  0.8906  0.8973  0.8898  0.8782
##              Connectivity   72.4151 101.8147 160.4833 174.4901 223.2544 248.8381
##              Dunn           0.0657  0.0657  0.0543  0.0459  0.0535  0.0500
##              Silhouette     0.3161  0.2849  0.2618  0.2371  0.2231  0.1863
##
## Optimal Scores:
##
##              Score   Method       Clusters
## APN          0.0699 kmeans       2
## AD           2.2322 kmeans       7
## ADM          0.2227 kmeans       2
## FOM          0.8723 kmeans       7
## Connectivity 28.5548 hierarchical 2
## Dunn         0.1682 hierarchical 2
## Silhouette   0.3265 kmeans       2
```

```
#According to three of the seven indices the best method is the K-means with two clusters.

avg.man_valid<-clValid(scale_df, nClust=2:7, clMethods=clmethod, validation=c("internal","stability"), maxitems=600,
                       metric="manhattan", method="average")
summary(avg.man_valid)
```

```
##
## Clustering Methods:
##  hierarchical kmeans pam
##
## Cluster sizes:
##  2 3 4 5 6 7
##
## Validation Measures:
##                              2       3       4       5       6       7
##
## hierarchical APN            0.0544  0.1266  0.1449  0.1720  0.1923  0.2247
##              AD             6.3121  5.6662  5.4306  5.3706  5.3315  5.2623
##              ADM            0.1956  0.5545  0.5289  0.5797  0.6206  0.7434
##              FOM            0.9810  0.9414  0.9063  0.9011  0.9010  0.8917
##              Connectivity   3.8579 43.8067 72.4373 76.0865 76.8889 95.2250
##              Dunn           0.2560  0.0852  0.1079  0.1079  0.1079  0.1079
##              Silhouette     0.3248  0.3378  0.3156  0.2985  0.2651  0.2132
## kmeans       APN            0.0699  0.1530  0.2015  0.2206  0.2546  0.3926
##              AD             5.2488  4.9612  4.7774  4.5724  4.4962  4.5858
##              ADM            0.2227  0.4593  0.6374  0.6543  0.7385  1.0464
##              FOM            0.8854  0.8785  0.8810  0.8829  0.8847  0.8753
##              Connectivity  72.2357 122.6774 153.4571 168.7552 198.9194 239.1008
##              Dunn           0.0903  0.0493  0.0493  0.0536  0.0536  0.0535
##              Silhouette     0.3673  0.2563  0.2430  0.2407  0.2311  0.1731
## pam          APN            0.0984  0.3804  0.4140  0.4411  0.4445  0.4830
##              AD             5.3068  5.2233  5.0402  4.8780  4.6965  4.5648
##              ADM            0.2783  0.8570  0.9937  1.0617  0.9849  0.9839
##              FOM            0.9000  0.8801  0.8991  0.8958  0.8882  0.8745
##              Connectivity  80.9655 171.1123 225.9171 246.4762 292.8071 311.1234
##              Dunn           0.0926  0.0380  0.0395  0.0523  0.0499  0.0175
##              Silhouette     0.3626  0.2639  0.2110  0.1826  0.1681  0.1532
##
## Optimal Scores:
##
##             Score  Method       Clusters
## APN          0.0544 hierarchical 2
## AD           4.4962 kmeans       6
## ADM          0.1956 hierarchical 2
## FOM          0.8745 pam          7
## Connectivity 3.8579 hierarchical 2
## Dunn         0.2560 hierarchical 2
## Silhouette   0.3673 kmeans       2
```

```
#According to four of the seven indices the best method is the Hierarchical one with two clusters.

#COMPLETE
com.euc_valid<-clValid(scale_df, nClust=2:7, clMethods=clmethod, validation=c("internal","stability"), maxitems=600,
                       metric="euclidean", method="complete")
summary(com.euc_valid)
```

```
##
## Clustering Methods:
##  hierarchical kmeans pam
##
## Cluster sizes:
##  2 3 4 5 6 7
##
## Validation Measures:
##                              2       3       4       5       6       7
##
## hierarchical APN           0.2571  0.3660  0.4173  0.4028  0.4243  0.4537
##              AD            3.0848  2.9519  2.9055  2.7980  2.7585  2.6240
##              ADM           0.7609  1.1081  1.3101  1.2559  1.2864  1.2406
##              FOM           0.9592  0.9341  0.9335  0.9293  0.9235  0.8967
##              Connectivity 43.2567 113.5615 139.8940 160.4246 172.7806 179.1032
##              Dunn          0.1100  0.0735  0.0746  0.0764  0.0772  0.0801
##              Silhouette    0.3346  0.1553  0.1787  0.1748  0.1614  0.1687
## kmeans       APN           0.0699  0.1800  0.3131  0.2437  0.3018  0.3257
##              AD            2.7403  2.5806  2.5362  2.3629  2.3504  2.3042
##              ADM           0.2227  0.5429  0.8873  0.7274  0.8765  0.9092
##              FOM           0.8854  0.8810  0.8845  0.8852  0.8806  0.8792
##              Connectivity 69.8119 116.4905 151.1194 162.1857 201.3984 233.9901
##              Dunn          0.0972  0.0589  0.0593  0.0595  0.0599  0.0664
##              Silhouette    0.3265  0.2545  0.2652  0.2644  0.2549  0.2489
## pam          APN           0.1271  0.2788  0.2428  0.3918  0.4703  0.4459
##              AD            2.7855  2.6700  2.4827  2.4452  2.4130  2.3098
##              ADM           0.4078  0.7481  0.6630  0.9331  1.0526  0.9708
##              FOM           0.9056  0.8932  0.8906  0.8973  0.8898  0.8782
##              Connectivity 72.4151 101.8147 160.4833 174.4901 223.2544 248.8381
##              Dunn          0.0657  0.0657  0.0543  0.0459  0.0535  0.0500
##              Silhouette    0.3161  0.2849  0.2618  0.2371  0.2231  0.1863
##
## Optimal Scores:
##
##             Score   Method       Clusters
## APN          0.0699 kmeans       2
## AD           2.3042 kmeans       7
## ADM          0.2227 kmeans       2
## FOM          0.8782 pam          7
## Connectivity 43.2567 hierarchical 2
## Dunn         0.1100 hierarchical 2
## Silhouette   0.3346 hierarchical 2
```

```
#According to three of the seven indices the best method is the Hierarchical one with two clusters.


com.man_valid<-clValid(scale_df, nClust=2:7, clMethods=clmethod, validation=c("internal","stability"), maxitems=600,
                       metric="manhattan", method="complete")
summary(com.man_valid)
```

```
##
## Clustering Methods:
##  hierarchical kmeans pam
##
## Cluster sizes:
##  2 3 4 5 6 7
##
## Validation Measures:
##                              2       3       4       5       6       7
##
## hierarchical APN           0.2144  0.3592  0.3975  0.4582  0.5005  0.4802
##              AD            5.6884  5.5859  5.4018  5.3238  5.2704  5.1591
##              ADM           0.7199  1.0396  1.1190  1.1871  1.2662  1.2809
##              FOM           0.9336  0.9244  0.9188  0.9133  0.9136  0.9073
##              Connectivity 73.1972 112.2873 149.7282 177.3341 186.7028 250.0575
##              Dunn          0.0652  0.0676  0.0745  0.0791  0.0853  0.0684
##              Silhouette    0.3176  0.2744  0.2322  0.2250  0.2146  0.1185
## kmeans       APN           0.0730  0.2048  0.3660  0.2624  0.3016  0.3128
##              AD            5.2453  5.0243  5.0052  4.6491  4.5480  4.4438
##              ADM           0.2445  0.5889  0.9875  0.7696  0.8277  0.8488
##              FOM           0.8856  0.8785  0.8770  0.8810  0.8823  0.8779
##              Connectivity 77.6964 125.1012 174.5480 168.7552 195.2595 224.9377
##              Dunn          0.0567  0.0544  0.0262  0.0536  0.0597  0.0599
##              Silhouette    0.3630  0.2752  0.2491  0.2407  0.2273  0.2254
## pam          APN           0.0984  0.3804  0.4140  0.4411  0.4445  0.4830
##              AD            5.3068  5.2233  5.0402  4.8780  4.6965  4.5648
##              ADM           0.2783  0.8570  0.9937  1.0617  0.9849  0.9839
##              FOM           0.9000  0.8801  0.8991  0.8958  0.8882  0.8745
##              Connectivity 80.9655 171.1123 225.9171 246.4762 292.8071 311.1234
##              Dunn          0.0926  0.0380  0.0395  0.0523  0.0499  0.0175
##              Silhouette    0.3626  0.2639  0.2110  0.1826  0.1681  0.1532
##
## Optimal Scores:
##
##              Score   Method       Clusters
## APN           0.0730 kmeans       2
## AD            4.4438 kmeans       7
## ADM           0.2445 kmeans       2
## FOM           0.8745 pam          7
## Connectivity 73.1972 hierarchical 2
## Dunn          0.0926 pam          2
## Silhouette    0.3630 kmeans       2
```

```
#According to three of the seven indices the best method is the K-means with two clusters.

#SINGLE
sin.euc_valid<-clValid(scale_df, nClust=2:7, clMethods=clmethod, validation=c("internal","stability"), maxitems=600,
                       metric="euclidean", method="single")
summary(sin.euc_valid)
```

```
##
## Clustering Methods:
##  hierarchical kmeans pam
##
## Cluster sizes:
##  2 3 4 5 6 7
##
## Validation Measures:
##                              2       3       4       5       6       7
##
## hierarchical APN           0.0023  0.0045  0.0098  0.0120  0.0180  0.0195
##              AD            3.1956  3.1834  3.1795  3.1727  3.1615  3.1513
##              ADM           0.0250  0.0310  0.0545  0.0625  0.0833  0.0994
##              FOM           0.9997  1.0000  0.9983  0.9990  0.9979  0.9978
##              Connectivity  3.9690  7.0409 10.0948 13.4155 16.1778 21.6357
##              Dunn          0.3055  0.2841  0.2735  0.2603  0.2539  0.2461
##              Silhouette    0.3019  0.2516  0.1770  0.1528  0.1527  0.1467
## kmeans       APN           0.0699  0.2203  0.3250  0.2273  0.2717  0.3517
##              AD            2.7403  2.6138  2.5556  2.3442  2.3078  2.3139
##              ADM           0.2227  0.6390  0.9270  0.6906  0.7772  0.9217
##              FOM           0.8854  0.8801  0.8839  0.8849  0.8825  0.8737
##              Connectivity 69.8119 122.9925 146.2694 157.3282 204.7353 232.2647
##              Dunn          0.0972  0.0716  0.0340  0.0628  0.0562  0.0688
##              Silhouette    0.3265  0.2731  0.2717  0.2633  0.2574  0.2564
## pam          APN           0.1271  0.2788  0.2428  0.3918  0.4703  0.4459
##              AD            2.7855  2.6700  2.4827  2.4452  2.4130  2.3098
##              ADM           0.4078  0.7481  0.6630  0.9331  1.0526  0.9708
##              FOM           0.9056  0.8932  0.8906  0.8973  0.8898  0.8782
##              Connectivity 72.4151 101.8147 160.4833 174.4901 223.2544 248.8381
##              Dunn          0.0657  0.0657  0.0543  0.0459  0.0535  0.0500
##              Silhouette    0.3161  0.2849  0.2618  0.2371  0.2231  0.1863
##
## Optimal Scores:
##
##              Score  Method       Clusters
## APN          0.0023 hierarchical 2
## AD           2.3078 kmeans       6
## ADM          0.0250 hierarchical 2
## FOM          0.8737 kmeans       7
## Connectivity 3.9690 hierarchical 2
## Dunn         0.3055 hierarchical 2
## Silhouette   0.3265 kmeans       2
```

```
#According to four of the seven indices the best method is the Hierarchical one with two clusters.


sin.man_valid<-clValid(scale_df, nClust=2:7, clMethods=clmethod, validation=c("internal","stability"), maxitems=600,
                       metric="manhattan", method="single")
summary(sin.man_valid)
```

```
##
## Clustering Methods:
##  hierarchical kmeans pam
##
## Cluster sizes:
##  2 3 4 5 6 7
##
## Validation Measures:
##                              2       3       4       5       6       7
##
## hierarchical APN           0.0030  0.0045  0.0075  0.0079  0.0102  0.0124
##              AD            6.3054  6.2767  6.2602  6.2274  6.2020  6.1787
##              ADM           0.0295  0.0354  0.0611  0.0572  0.0625  0.0698
##              FOM           0.9982  0.9972  0.9965  0.9968  0.9958  0.9955
##              Connectivity  3.8579  6.8869 11.7488 14.9000 17.9718 21.0008
##              Dunn          0.2560  0.2346  0.2151  0.2120  0.2085  0.1913
##              Silhouette    0.3248  0.2909  0.2172  0.1891  0.1516  0.1421
## kmeans       APN           0.0699  0.1935  0.2786  0.2480  0.3881  0.2919
##              AD            5.2488  5.0240  4.9337  4.6049  4.6808  4.4538
##              ADM           0.2227  0.6013  0.8991  0.7142  1.0342  0.8337
##              FOM           0.8854  0.8839  0.8842  0.8820  0.8850  0.8766
##              Connectivity 72.2357 126.3679 159.9206 163.6786 232.1246 230.3496
##              Dunn          0.0903  0.0493  0.0493  0.0517  0.0597  0.0404
##              Silhouette    0.3673  0.2427  0.2379  0.2378  0.2108  0.2148
## pam          APN           0.0984  0.3804  0.4140  0.4411  0.4445  0.4830
##              AD            5.3068  5.2233  5.0402  4.8780  4.6965  4.5648
##              ADM           0.2783  0.8570  0.9937  1.0617  0.9849  0.9839
##              FOM           0.9000  0.8801  0.8991  0.8958  0.8882  0.8745
##              Connectivity 80.9655 171.1123 225.9171 246.4762 292.8071 311.1234
##              Dunn          0.0926  0.0380  0.0395  0.0523  0.0499  0.0175
##              Silhouette    0.3626  0.2639  0.2110  0.1826  0.1681  0.1532
##
## Optimal Scores:
##
##             Score  Method       Clusters
## APN         0.0030 hierarchical 2
## AD          4.4538 kmeans       7
## ADM         0.0295 hierarchical 2
## FOM         0.8745 pam          7
## Connectivity 3.8579 hierarchical 2
## Dunn        0.2560 hierarchical 2
## Silhouette  0.3673 kmeans       2
```

```
#According to four of the seven indices the best method is the Hierarchical one with two clusters.

#WARD
ward.euc_valid<-clValid(scale_df, nClust=2:7, clMethods="hierarchical", validation=c("internal","stability"), maxitems=600,
                        metric="euclidean", method="ward")
```

```
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
```

```
summary(ward.euc_valid)
```

```
## 
## Clustering Methods:
##  hierarchical 
## 
## Cluster sizes:
##  2 3 4 5 6 7 
## 
## Validation Measures:
##                                  2        3        4        5        6        7
##                                                                                 
## hierarchical APN            0.1428   0.2824   0.3006   0.3397   0.3962   0.4236
##              AD             2.8377   2.6785   2.5709   2.4814   2.4056   2.3579
##              ADM            0.4118   0.7278   0.8036   0.8561   0.8998   0.9520
##              FOM            0.8945   0.8908   0.8875   0.8815   0.8755   0.8710
##              Connectivity  71.1377 112.4123 116.5683 130.6532 158.9024 192.4940
##              Dunn           0.1160   0.0574   0.0574   0.0574   0.0606   0.0606
##              Silhouette     0.3158   0.2142   0.2344   0.2260   0.2194   0.1301
## 
## Optimal Scores:
## 
##              Score   Method       Clusters
## APN          0.1428 hierarchical 2       
## AD           2.3579 hierarchical 7       
## ADM          0.4118 hierarchical 2       
## FOM          0.8710 hierarchical 7       
## Connectivity 71.1377 hierarchical 2       
## Dunn         0.1160 hierarchical 2       
## Silhouette   0.3158 hierarchical 2
```

```
ward.man_valid<-clValid(scale_df, nClust=2:7, clMethods="hierarchical", validation=c("internal","stability"), maxitems=600,
                   metric="manhattan", method="ward")
```

```
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
## Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
```

```
summary(ward.man_valid)
```

```
##
## Clustering Methods:
##  hierarchical
##
## Cluster sizes:
##  2 3 4 5 6 7
##
## Validation Measures:
##                             2        3        4        5        6        7
##
## hierarchical APN        0.1906   0.3533   0.4275   0.4247   0.4458   0.4968
##              AD         5.6300   5.4034   5.2418   4.9804   4.8358   4.7035
##              ADM        0.5692   0.9442   1.1237   1.0801   1.0834   1.0968
##              FOM        0.8904   0.8823   0.8819   0.8815   0.8787   0.8713
##              Connectivity   51.5067 102.2353 125.9837 144.3944 176.9262 190.0258
##              Dunn       0.0945   0.0470   0.0470   0.0549   0.0549   0.0549
##              Silhouette   0.3446   0.2146   0.1935   0.2142   0.1132   0.1135
##
## Optimal Scores:
##
##              Score   Method       Clusters
## APN          0.1906 hierarchical 2
## AD           4.7035 hierarchical 7
## ADM          0.5692 hierarchical 2
## FOM          0.8713 hierarchical 7
## Connectivity 51.5067 hierarchical 2
## Dunn         0.0945 hierarchical 2
## Silhouette   0.3446 hierarchical 2
```

```
##The conclusion is that the Hierarchical with k=2 clusters is the best clustering method.
```