Project Report for Data Analysis

COVID-19 Coronavirus Pandemic

Author Lombardia Giuseppe

Data Analysis course

Prof. Antonio Punzo

MSc in Data Science for Management

Contents

1 Introduction

This report is based on a Kaggle dataset called COVID-19 Coronavirus Pandemic (the dataset is available here). The dataset was obtained from Worldometers website. Inside it contains some variables such as Total cases, Total Deaths, Total Cases//1M pop, Total Deaths/1M pop, Death percentage related to COVID 19 Coronavirus pandemic. This dataset provides a comprehensive overview of the COVID-19 situation worldwide, presenting up-to-date data on confirmed cases, deaths, and population metrics for each country. Through its columns, we can understand not only the extent of virus spread in absolute terms, but also the relative impact on the population of different countries, measured in cases and deaths per million population, as well as the lethality of the virus expressed as a percentage of deaths to total cases. These data are essential for analyzing the spread of the virus, assessing the effectiveness of containment measures, and better understanding how different regions of the world have been affected by the pandemic.

2 Data structure

The provided code accomplishes the following:

It initiates by loading data from a CSV file titled "COVID-19 Coronavirus.csv" and stores it within a dataframe labeled 'dataset'. Subsequently, the 'attach()' function is employed to link the 'dataset' dataframe, facilitating direct access to its columns without the necessity of explicitly specifying the dataframe name. Nevertheless, it's important to note that using 'attach()' may lead to ambiguity issues and is generally not recommended. Following this, three columns ('Other.names', 'ISO.3166.1.alpha.3.CODE', and 'Continent') are removed from the dataset dataframe by assigning them to NULL, effectively excluding them. Furthermore, the 'summary()' function generates a statistical summary of the 'dataset' dataframe, providing counts, minimum and maximum values, means, and quantiles for each column. Lastly, the 'str()' function is utilized to display the structure of the 'dataset' dataframe, revealing the data type and a brief preview of the data within each column. In summary, this code facilitates the preparation and initial exploration of the dataset, thereby setting the stage for further analysis and manipulation.

```
dataset <- read.csv("COVID-19 Coronavirus.csv")
attach(dataset)
dataset$Other.names <- NULL
dataset$ISO.3166.1.alpha.3.CODE <- NULL
dataset$Continent <- NULL
summary(dataset)
str(dataset)
```

```
'data.frame':   225 obs. of  7 variables:
 $ Country          : chr  "Afghanistan" "Albania" "Algeria" "Andorra" ...
 $ Population       : int  40462186 2872296 45236699 77481 34654212 15237 99348 45921761 2972939 107560
...
 $ Total.Cases      : int  177827 273870 265691 40024 99194 2700 7493 9041124 422574 34051 ...
 $ Total.Deaths     : int  7671 3492 6874 153 1900 9 135 128065 8617 212 ...
 $ Tot.Cases..1M.pop: int  4395 95349 5873 516565 2862 177200 75422 196881 142140 316577 ...
 $ Tot.Deaths.1M.pop: int  190 1216 152 1975 55 591 1359 2789 2898 1971 ...
 $ Death.percentage : num  4.314 1.275 2.587 0.382 1.915 ...
```

Figure 1: Dataset

The variable types are:

- **Country**: Categorical variable with 225 observations
- **Population**: Number of population of each country, it is a continuous numerical variable
- **Total Cases**: Number of total cases for each country, it is a continuous numerical variable
- **Total Deaths**: Number of total deaths for each country, it is a continuous numerical variable
- **Total Cases 1M Pop.**: Total number of cases relative to the million, it is a continuous numerical variable
- **Total Deaths 1M Pop**: Total number of deaths relative to the million, it is a continuous numerical variable
- **Death percentage**: This is the lethality rate of the disease, it is a number contained in the range between 0 and 100.

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

```
  Country           Population         Total.Cases        Total.Deaths     Tot.Cases..1M.pop
Length:225         Min.   :8.050e+02   Min.   :       1   Min.   :      0   Min.   :     9
Class :character   1st Qu.:5.666e+05   1st Qu.:   24071   1st Qu.:    189   1st Qu.: 11384
Mode  :character   Median :5.828e+06   Median :  163936   Median :   1965   Median : 88987
                   Mean   :3.507e+07   Mean   : 2184781   Mean   :  27448   Mean   :136900
                   3rd Qu.:2.191e+07   3rd Qu.: 1092547   3rd Qu.:  13660   3rd Qu.:223335
                   Max.   :1.439e+09   Max.   :81839052   Max.   :1008222   Max.   :696044
Tot.Deaths.1M.pop Death.percentage
Min.   :   0      Min.   : 0.0000
1st Qu.: 123      1st Qu.: 0.5113
Median : 708      Median : 1.0369
Mean   :1097      Mean   : 1.4441
3rd Qu.:1795      3rd Qu.: 1.9770
Max.   :6286      Max.   :18.1518
```

Figure 2: Summary Dataset

3 Univariate Analysis

Univariate analysis is a statistical technique used to examine and describe the characteristics of a single variable within a data set. This type of analysis is fundamental in statistical science and data analysis because it provides a detailed and in-depth overview of one variable at a time, allowing us to understand its distributions, trends, relationships and statistical properties.

3.1 Country

Country is a categorical variable, that indicates the country of the world based on our study and dataset. Of course, due to the type of categorical variable, the table() output will be this because the variable Country has 225 levels. So the frequency is obviously is 1 for every country.

Afghanistan	Albania
1	1
Algeria	Andorra
1	1
Angola	Anguilla
1	1
Antigua and Barbuda	Argentina
1	1
Armenia	Aruba
1	1
Australia	Austria
1	1
Azerbaijan	Bahamas
1	1
Bahrain	Bangladesh
1	1
Barbados	Belarus
1	1
Belgium	Belize
1	1
Benin	Bermuda
1	1
Bhutan	Bolivia

Figure 3: Table country

3.2 Population

Population is a continuous variable that indicates the total amount of population for each country of the world. This variable is a continuous random variable with a support $[0, +\infty)$. First of all, we take a look at the histogram and boxplot in order to obtain an overview of this variable.

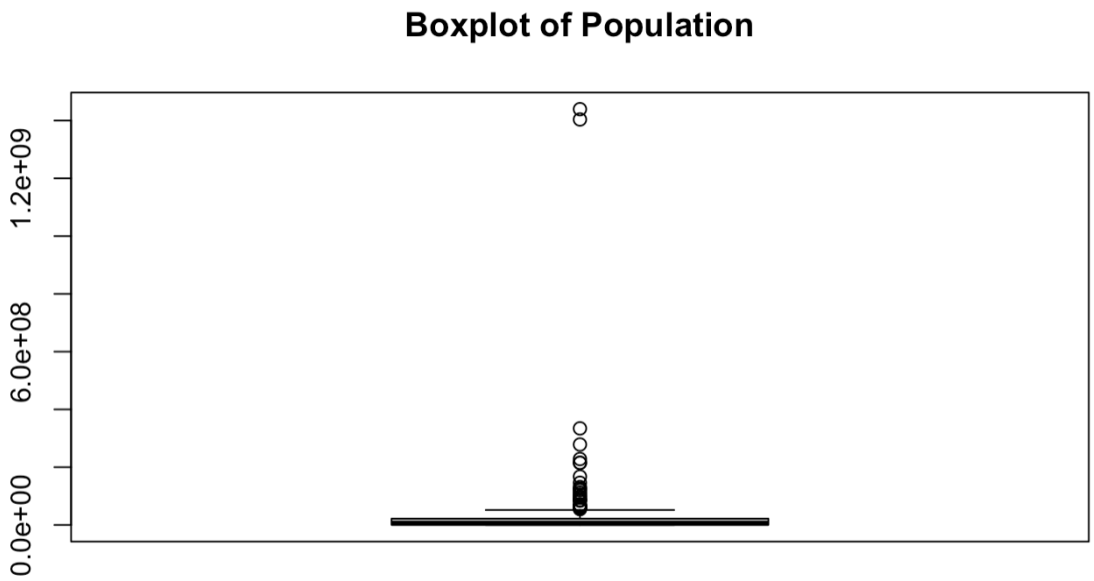


Figure 4: Boxplot of population

```
    summary(population)
         Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
8.050e+02 5.666e+05 5.828e+06 3.507e+07 2.191e+07 1.439e+09
```

The command above is used to obtain a concise overview of the data contained in the variable "population". This function provides a series of descriptive statistics, such as the mean, median, minimum, maximum, first quartile, third quartile, as well as the total number of non-missing values in the variable. Essentially, it helps quickly understand the distribution and main characteristics of the analyzed population.

- **Min.**: The minimum value in the "population" variable. In this case, it is 805.
- **1st Qu.**: The first quartile, which separates the bottom 25% of the data from the top 75%. In this case, it is 566,600.
- **Median**: The median, which is the central value of the distribution. In this case, it is 5,828,000.
- **Mean**: The arithmetic mean of the data in the "population" variable. In this case, it is 35,070,000.
- **3rd Qu.**: The third quartile, which separates the bottom 75% of the data from the top 25%. In this case, it is 21,910,000.
- **Max.**: The maximum value in the "population" variable. In this case, it is 1,439,000,000.

The command "table(population)" is used to create a frequency table of the different values present in the variable "population". This command returns a table that lists each unique value in the "population" variable along with the number of times each value appears in the data. This can be useful for examining the distribution of the data and understanding the frequency of each value within the dataset.

```
    > table(Population)
Population
      805      1645      3657      4997      5744      6109      9930     10894
        1         1         1         1         1         1         1         1
    15237     17592     18245     26650     30583     33673     34056     38320
        1         1         1         1         1         1         1         1
    39634     39729     39820     43728     49188     53858     56942     59889
        1         1         1         1         1         1         1         1
    61875     67073     72299     77481     85821     99348     99413    107560
        1         1         1         1         1         1         1         1
   107792    111557    113436    117134    122656    165268    176668    185096
        1         1         1         1         1         1         1         1
   200722    226281    283751    284330    287991    290302    312224    319701
        1         1         1         1         1         1         1         1
   345120    374756    399822    400244    410260    443602    444812    557204
        1         1         1         1         1         1         1         1
   566557    595833    623031    628205    643801    664828    716351    786480
        1         1         1         1         1         1         1         1
   793196    902011    906497    907817   1013146   1181191   1222745   1275463
        1         1         1         1         1         1         1         1
  1328097   1362386   1407422   1483588   1804995   1849698   2049374   2079438
        1         1         1         1         1         1         1         1
  2083224   2171978   2317612   2434708   2535418   2621429   2655811   2807805
        1         1         1         1         1         1         1         1
  2872296   2972939   2983794   3245097   3370682   3494806   3632329   3975762
        1         1         1         1         1         1         1         1
  4017550   4060951   4381108   4433639   4863443   4976719   5002100   5034333
        1         1         1         1         1         1         1         1
  5175547   5265647   5308883   5333815   5464272   5495449   5555788   5755689
        1         1         1         1         1         1         1         1
  5827911   5930887   6543499   6712569   6762511   6771939   6856886   7034832
        1         1         1         1         1         1         1         1
  7285892   7460338   7603455   8260822   8618172   8675762   8765420   9096360
        1         1         1         1         1         1         1         1
  9243590   9326000   9443882   9617409   9912437  10099567  10144662  10180299
        1         1         1         1         1         1         1         1
 10209507  10299156  10333930  10380442  10743762  11038333  11314513  11423439
        1         1         1         1         1         1         1         1
 11645833  11677924  11951714  12035092  12510155  12678649  13513881  13755881
        1         1         1         1         1         1         1         1
 15241601  16668781  17123941  17201245  17250246  17515750  18111933  18244381
        1         1         1         1         1         1         1         1
 18495493  19013049  19169833  19284482  19403451  19994654  21271006  21570428
        1         1         1         1         1         1         1         1
 21905848  23892241  25738714  26017767  27520953  27701805  28294895  28936285
        1         1         1         1         1         1         1         1
```

```
  30053867   30975258   32207812   32787052   33091831   33775745   34318156   34654212
         1          1          1          1          1          1          1          1
  35762746   37676342   37774045   38321435   40462186   41801625   43273831   45236699
         1          1          1          1          1          1          1          1
  45640385   45921761   46786482   48267221   51346429   51832231   55048340   55843563
         1          1          1          1          1          1          1          1
  60306185   60617532   62710097   65526369   68510300   70106601   84252947   85874667
         1          1          1          1          1          1          1          1
  85927644   94323344   98871712  105711844  112133868  119945147  125798669  131303955
         1          1          1          1          1          1          1          1
 146044010  167561502  215077352  215204501  228397520  278586508  334400597 1403754381
         1          1          1          1          1          1          1          1
1439323776
         1
```

I will now show the population variable graphically using the following commands.

```
hist(Population, breaks = 225, col="red",
main = "Histogram of population", freq = FALSE)
box()
lines(density(Population), col="black")
```

- **hist(Population, breaks = 225, col="red", main = "Histogram of population", freq = FALSE)**: This command creates a histogram of the variable "Population". The parameter `breaks = 225` specifies the number of intervals (bins) into which the data range is divided. The `col="red"` argument sets the color of the bars to red, and `main = "Histogram of population"` sets the main title of the histogram. The parameter `freq = FALSE` indicates that the y-axis should display densities rather than frequencies.
- **box()**: This function adds a box around the current plot.
- **lines(density(Population), col="black")**: This command overlays a density plot on the histogram. The `density()` function calculates a kernel density estimate of the variable "Population". The `lines()` function then plots this density estimate over the histogram, with the `col="black"` argument specifying the color of the density line.

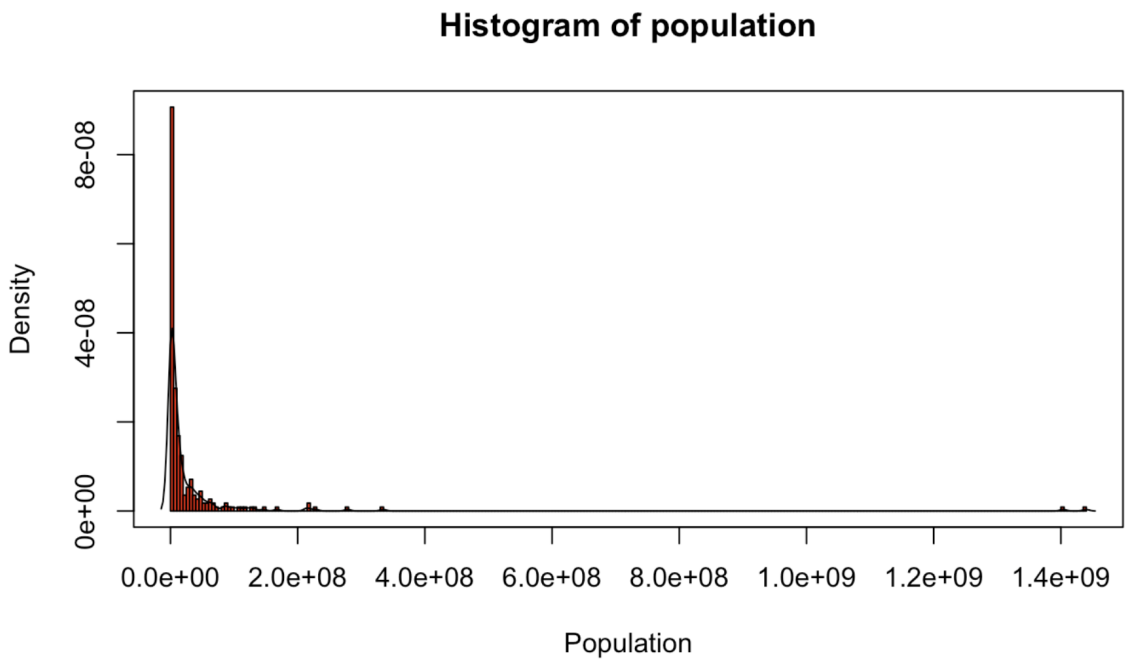


Figure 5: Histogram of population

```
    > skewness(Population)
[1] 8.942814
> kurtosis(Population)
[1] 88.43826
> sd(Population)
[1] 139241848
> var(Population)
[1] 1.938829e+16
```

- **Skewness**: The skewness of the "Population" variable is approximately 8.942814. Skewness measures the symmetry of the distribution. A positive skewness indicates a right-skewed distribution, where the tail on the right side is longer or fatter than the left side.
- **Kurtosis**: The kurtosis of the "Population" variable is approximately 88.43826. Kurtosis measures the tailedness or peakedness of a distribution. A high kurtosis value suggests that the distribution has heavy tails and is more peaked than a normal distribution.
- **Standard Deviation (SD)**: The standard deviation of the "Population" variable is approximately 139,241,848. The standard deviation measures the dispersion or spread of the data points around the mean. A higher standard deviation indicates greater variability in the data.
- **Variance**: The variance of the "Population" variable is approximately 1.938829e+16 (which is in scientific notation, equivalent to 1.938829×10^{16}). Variance measures the average squared deviation of each data point from the mean. It provides a measure of the spread of the data points. A higher variance indicates greater variability in the data.

```
    Pop <- Population
fit.GA <- histDist(Pop, family = GA, nbins = 225,
main = "Gamma distribution")
fit.EXP <- histDist(Pop, family = EXP, nbins = 225,
main = "Exponential distribution")
fit.IG <- histDist(Pop, family = IG, nbins = 225,
main = "Inversian Gaussian distribution")
fit.LOGNO <- histDist(Pop, family = LOGNO, nbins = 225,
main = "Log-Normal distribution")
fit.WEI <- histDist(Pop, family = WEI, nbins = 225,
main = "Weibull distribution")
fit.GIG <- histDist(Pop, family = GIG, nbins = 225,
main = "Generalized Inversed Gaussian")
```

I tried to fit different models to 'Population' distribution evaluating the Akaike Information Criterion (AIC) and the Bayesian Information criterion (BIC), The lower are the indexes, the better is the fitting. I evaluated both the single parameter distributions and mixture distributions.

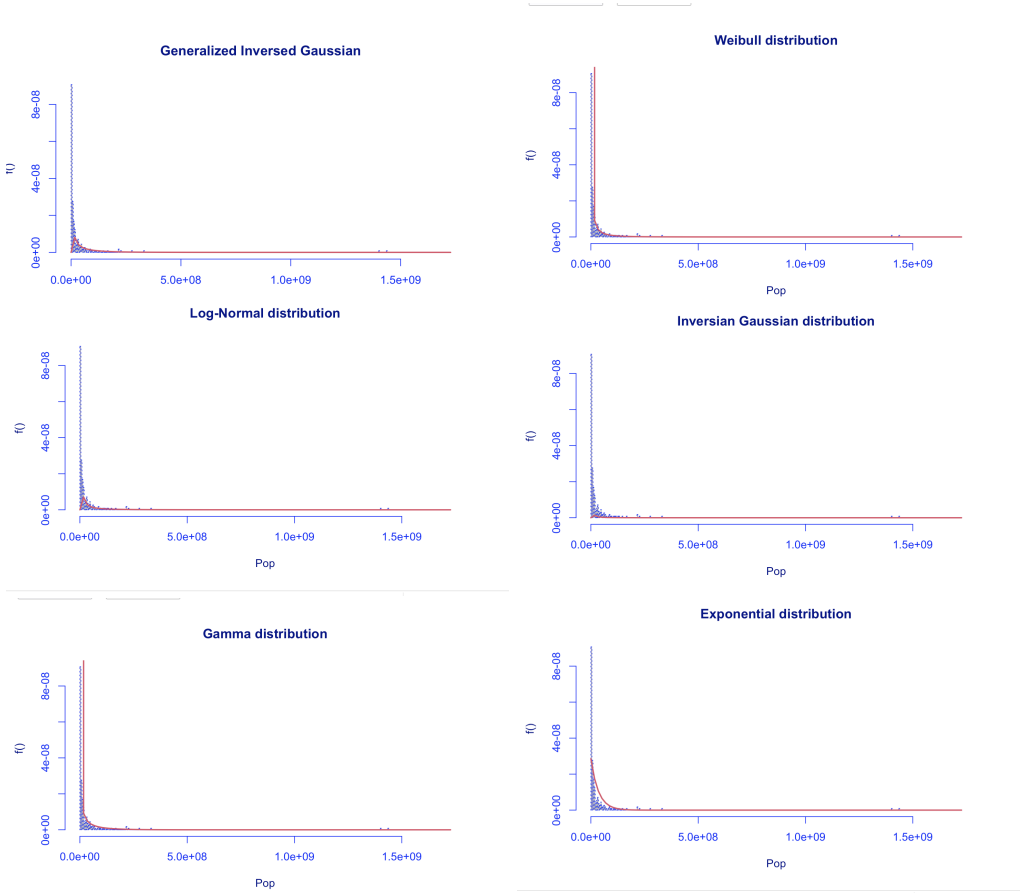


Figure 6: fitting models

```
data.frame(
    row.names = c(
        "Gamma", "Exponential", "Inversian Gaussian",
        "Log-Normal", "Weibull", "Generalized Inversian Gaussian"
    ),
    "num_of_par" = c(
        fit.GA$df.fit, fit.EXP$df.fit, fit.IG$df.fit,
        fit.LOGNO$df.fit, fit.WEI$df.fit, fit.GIG$df.fit
    ),
    "logLikelihood" = c(
        logLik(fit.GA), logLik(fit.EXP), logLik(fit.IG),
        logLik(fit.LOGNO), logLik(fit.WEI), logLik(fit.GIG)
    ),
    "AIC" = c(
        AIC(fit.GA), AIC(fit.EXP), AIC(fit.IG),
        AIC(fit.LOGNO), AIC(fit.WEI), AIC(fit.GIG)
    ),
    "BIC" = c(
        fit.GA$sbc, fit.EXP$sbc, fit.IG$sbc,
        fit.LOGNO$sbc, fit.WEI$sbc, fit.GIG$sbc
    ),
    "MU" = c(
        fitted(fit.GA, "mu")[1], fitted(fit.EXP, "mu")[1],
        fitted(fit.IG, "mu")[1], fitted(fit.LOGNO, "mu")[1],
        fitted(fit.WEI, "mu")[1], fitted(fit.GIG, "mu")[1]
    ),
    "SIGMA" = c(
        fitted(fit.GA, "sigma")[1], NA, fitted(fit.IG, "sigma")[1],
        fitted(fit.LOGNO, "sigma")[1], fitted(fit.WEI, "sigma")[1],
        fitted(fit.GIG, "sigma")[1]
    ),
    "NU" = c(
        fitted(fit.GIG, "nu")[1], NA, NA, NA, NA, NA
    )
)
```

	num_of_par <dbl>	logLikelihood <dbl>	AIC <dbl>	BIC <dbl>	MU <dbl>	SIGMA <dbl>	NU <dbl>
Gamma	2	-3930.195	7864.391	7871.223	3.507320e+07	1.851404849	0.1793046
Exponential	1	-4133.913	8269.827	8273.243	3.507321e+07	*NA*	*NA*
Inversian Gaussian	2	-4138.977	8281.954	8288.786	3.507324e+07	0.004000674	*NA*
Log-Normal	2	-3918.730	7841.459	7848.292	1.499854e+01	2.715862092	*NA*
Weibull	2	-3910.364	7824.729	7831.561	1.164021e+07	0.439555440	*NA*
Generalized Inversian Gaussian	3	-3916.992	7839.985	7850.233	3.507323e+07	11.643017889	*NA*

Figure 7: dataframe

According to the analysis I executed on single distribution models, the best model to fit is the **Weibull Distribution**. I also tried to fit the distribution with a Weibull mixture, first with k=2 and then with k=3.

```
mix.WEI.1<- gamlssMXfits(n=5,Population~1,family = WEI, K=2, data = NULL)
mix.WEI<- gamlssMXfits(n=5,Population~1,family = WEI, K=3, data = NULL)
```

```
    Warning: Algorithm RS has not yet convergedmodel= 1
model= 2
Warning: Algorithm RS has not yet convergedWarning:
Algorithm RS has not yet convergedmodel= 3
model= 4
Warning: Algorithm RS has not yet convergedmodel= 5
Warning: Algorithm RS has not yet convergedmodel= 1
Warning: Algorithm RS has not yet convergedWarning:
Algorithm RS has not yet convergedmodel= 2
Warning: Algorithm RS has not yet convergedmodel= 3
Warning: Algorithm RS has not yet convergedWarning:
Algorithm RS has not yet convergedmodel= 4
model= 5
```

	AIC <dbl>	BIC <dbl>
Weibull mixture with K=2	8240.815	8257.896
Weibull mixture with K=3	8246.817	8274.146

2 rows

Figure 8: dataframe

According to the obtained value I choose Weibull mixture K=2 because produce a better result than Weibull mixture K=3.

3.2.1 References of Theory

When we want to compare the goodness of fit of a model among different models, we need a value that allows us to compare these models, in particular, we will use 2 coefficients:

- **AIC**
- **BIC**

Akaike Information Criterion (AIC)

This is the function of AIC where:

$$\text{AIC} = -n^2 \times \ln(L) + 2 \times nk$$

- **n**: number of observations
- **L**: Likelihood function computed at the maximum point
- **k**: number of free parameters

We will choose the model with a lesser AIC.

Bayesian Information Criterion (BIC)

This is the function of BIC where:

$$\text{BIC} = -2 \times \ln(L) + \ln(n) \times k$$

- **n**: number of observations
- **L**: Likelihood function computed at the maximum point
- **k**: number of free parameters

We will choose models with a lesser BIC.

As we can see, the value of AIC penalizes models with a greater number of parameters less than the value of BIC.

```
 print(mix.WEI.1)

Mixing Family:  c("WEI", "WEI")

Fitting method: EM algorithm

Call:  gamlssMX(formula = Population ~ 1, family = WEI, K = 2,      data = NULL)

Mu Coefficients for model: 1
(Intercept)
      22.02
Sigma Coefficients for model: 1
(Intercept)
     -1.665
Mu Coefficients for model: 2
(Intercept)
      22.02
Sigma Coefficients for model: 2
(Intercept)
     -1.665

Estimated probabilities: 0.5054075 0.4945925

Degrees of Freedom for the fit: 5 Residual Deg. of Freedom   220
Global Deviance:     8230.82
            AIC:     8240.82
            SBC:     8257.9
```

```
    > mix.WEI.1$prob
[1] 0.5054075 0.4945925
```

- The vector `mix.WEI.1$prob` contains the probabilities associated with each of the two components of the Weibull mixture. In this case, there are two components in the Weibull mixture, so there are two values in the `prob` vector.

- The first value, 0.5054075, represents the probability associated with the first component of the Weibull mixture. This means that approximately 50.54% of the observations are modeled by the Weibull distribution associated with this component.
- The second value, 0.4945925, represents the probability associated with the second component of the Weibull mixture. This indicates that approximately 49.46% of the observations are modeled by the Weibull distribution associated with this component.
- Essentially, the probabilities indicate the proportion of data attributed to each component in the overall mixture. If one of the probabilities is much higher than the other, it means that that component predominantly contributes to the overall distribution. However, since the two probabilities are similar in this case, both components have a significant impact on the overall structure of the Weibull mixture distribution.

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

mu.hat2 <- exp(mix.WEI.1[["models"]][[2]][["mu.coefficients"]])
sigma.hat2 <- exp(mix.WEI.1[["models"]][[2]][["sigma.coefficients"]])

hist(Population, breaks = 225, freq = FALSE, main = "Mixture Weibull distribution K = 2")

lines(seq(min(Population), max(Population),
length = length(Population)),
      mix.WEI.1[["prob"]][1] * dWEI(seq(min(Population),
      max(Population), length = length(Population)),
      mu = mu.hat1, sigma = sigma.hat1),
      lty = 2, lwd = 3, col = 2)

lines(seq(min(Population), max(Population),
length = length(Population)),
      mix.WEI.1[["prob"]][2] * dWEI(seq(min(Population),
      max(Population), length = length(Population)),
      mu = mu.hat2, sigma = sigma.hat2),
      lty = 2, lwd = 3, col = 3)
```

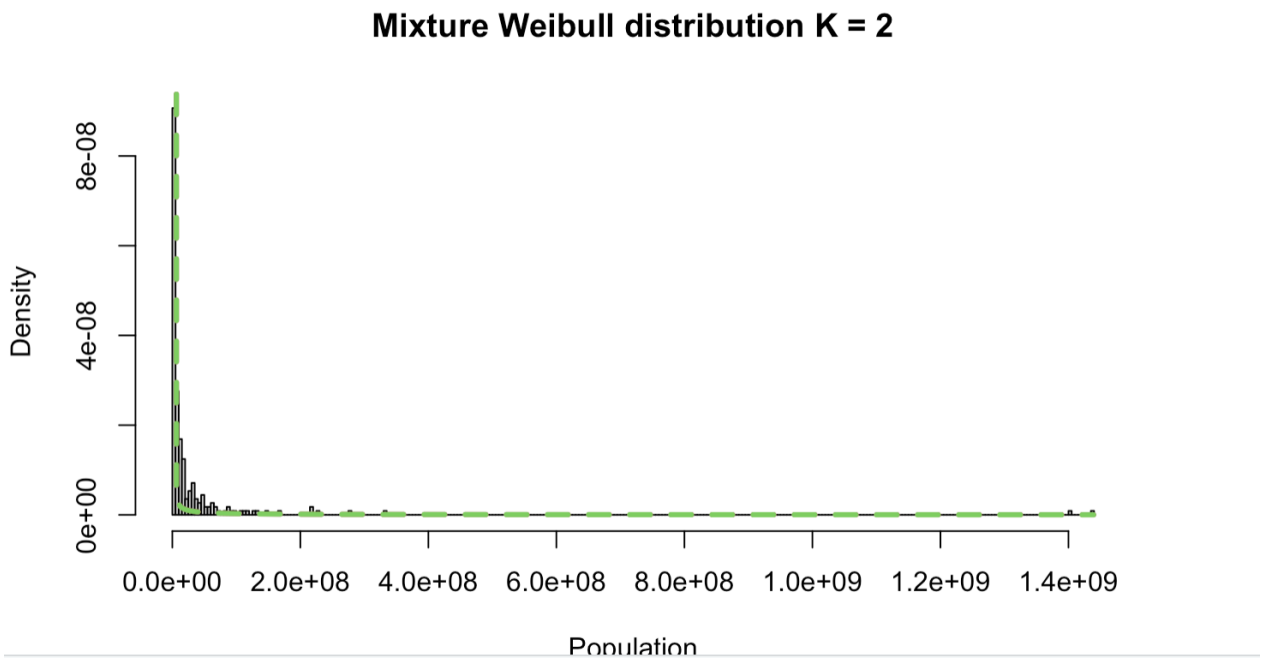


Figure 9: Mixture Weibull Distribution K=2

3.3 Total Cases

Total cases is a numeric and continuous variable. First I plot the boxplot and then the histogram of total cases

```
> summary(Total.Cases)
Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
   1     24071    163936   2184781   1092547  81839052
> skewness(Total.Cases)
[1] 7.356084
> kurtosis(Total.Cases)
[1] 71.0178
hist(Total.Cases, breaks = 224, col = "blue",
main = "Histogram of Total cases", freq = FALSE)
box()
lines(density(Total.Cases), col="red")
```

Figure 10: Boxplot of Total cases

```
    TotCas <- Total.Cases
fit.GA.1 <- histDist(TotCas, family = GA, nbins = 224,
main = "Gamma distribution")
fit.EXP.1 <- histDist(TotCas, family = EXP, nbins = 224,
main = "Exponential distribution")
fit.IG.1 <- histDist(TotCas, family = IG, nbins = 224,
main = "Inversian Gaussian distribution")
fit.LOGNO.1 <- histDist(TotCas, family = LOGNO, nbins = 224,
main = "Log-Normal distribution")
fit.WEI.1 <- histDist(TotCas, family = WEI, nbins = 224,
main = "Weibull distribution")
fit.GIG.1 <- histDist(TotCas, family = GIG, nbins = 224,
main = "Generalized Inversed Gaussian")
```

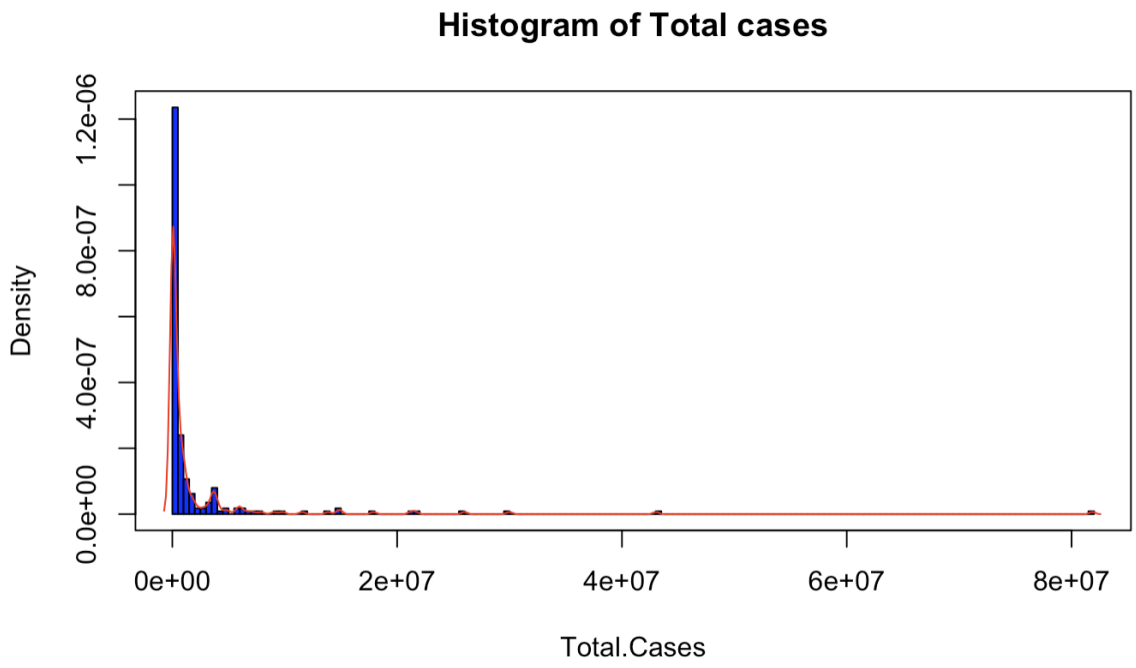


Figure 11: Histogram of Total cases

```
data.frame(
    row.names = c("Gamma", "Exponential",
    "Inversian Gaussian",
    "Log-Normal", "Weibull",
    "Generalized Inversian Gaussian"),
    "num_of_par" = c(
        fit.GA.1$df.fit, fit.EXP.1$df.fit, fit.IG.1$df.fit,
        fit.LOGNO.1$df.fit, fit.WEI.1$df.fit, fit.GIG.1$df.fit),
    "logLikelihood" = c(
        logLik(fit.GA.1), logLik(fit.EXP.1), logLik(fit.IG.1),
        logLik(fit.LOGNO.1), logLik(fit.WEI.1), logLik(fit.GIG.1)),
    "AIC" = c(
        AIC(fit.GA.1), AIC(fit.EXP.1), AIC(fit.IG.1),
        AIC(fit.LOGNO.1), AIC(fit.WEI.1), AIC(fit.GIG.1)),
    "BIC" = c(
        fit.GA.1$sbc, fit.EXP.1$sbc, fit.IG.1$sbc,
        fit.LOGNO.1$sbc, fit.WEI.1$sbc, fit.GIG.1$sbc),
    "MU" = c(
        fitted(fit.GA.1, "mu")[1], fitted(fit.EXP.1, "mu")[1],
        fitted(fit.IG.1, "mu")[1], fitted(fit.LOGNO.1, "mu")[1],
        fitted(fit.WEI.1, "mu")[1], fitted(fit.GIG.1, "mu")[1]),
    "SIGMA" = c(
        fitted(fit.GA.1, "sigma")[1], NA, fitted(fit.IG.1, "sigma")[1],
        fitted(fit.LOGNO.1, "sigma")[1], fitted(fit.WEI.1, "sigma")[1],
        fitted(fit.GIG.1, "sigma")[1]),
    "NU" = c(
        fitted(fit.GIG.1, "nu")[1], NA, NA, NA, NA, NA))
```

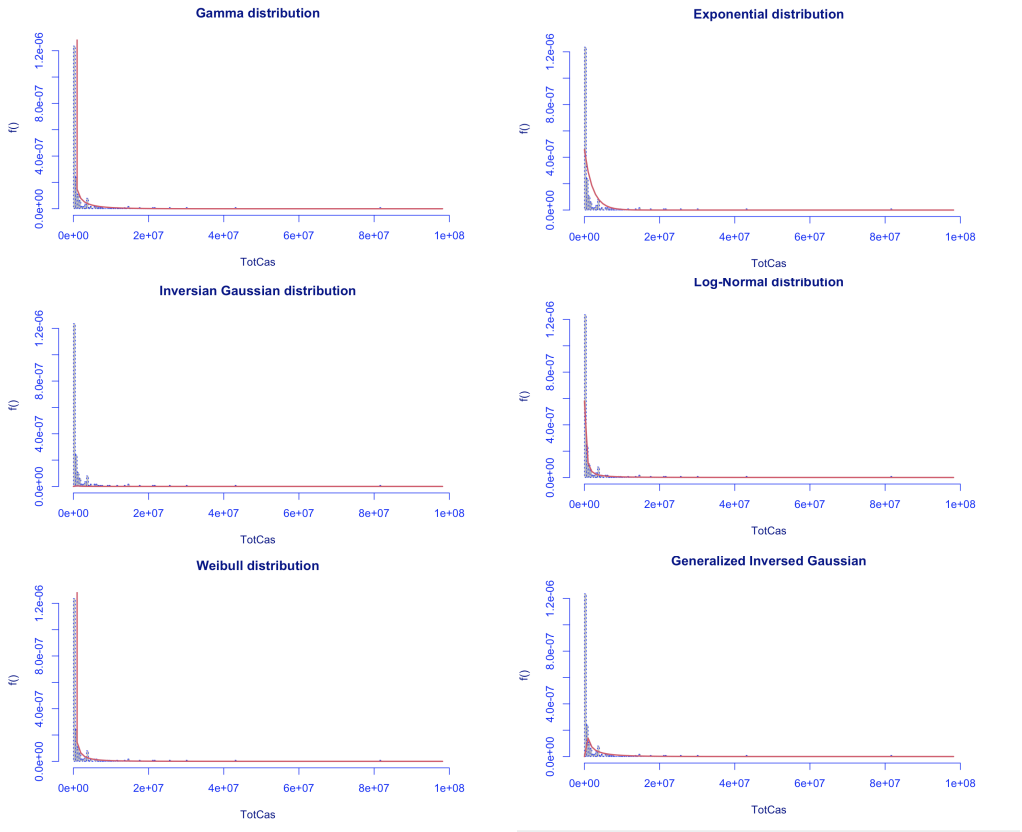


Figure 12: fitting models of Total Cases

Description: df [6 × 4]

	num_of_par <dbl>	logLikelihood <dbl>	AIC <dbl>	BIC <dbl>
Gamma	2	-3244.434	6492.868	6499.700
Exponential	1	-3509.331	7020.662	7024.078
Inversian Gaussian	2	-3784.215	7572.431	7579.263
Log-Normal	2	-3234.056	6472.111	6478.943
Weibull	2	-3223.666	6451.332	6458.165
Generalized Inversian Gaussian	3	-3239.899	6485.797	6496.045

6 rows

Figure 13: dataframe of Total cases

Description: df [2 × 2]

	AIC <dbl>	BIC <dbl>
Weibull mixture with K=3	6446.658	6473.987
Gamma mixture with K=3	6445.977	6473.306

2 rows

Figure 14: dataframe of Mixture Distributions

From the parameters just calculated, the Weibull distribution appears to have the lowest BIC and AIC values. Consequently I will again use this one which fits the data better. I will also give the GAMMA distribution a try.

```
    > mix.WEI.2 <- gamlssMXfits(n=5,TotCas~1,
    family = WEI, K = 3, data = NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
> mix.WEI.2$aic
[1] 6446.658
> mix.WEI.2$sbc
[1] 6473.987
> mix.WEI.2$prob
[1] 0.5703193 0.2395784 0.1901023

> mix.GA <- gamlssMXfits(n=5, TotCas~1,
family = GA, K=3, data=NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
> mix.GA$aic
[1] 6445.977
> mix.GA$sbc
[1] 6473.306
> mix.GA$prob
[1] 0.4811892 0.2132715 0.3055393


    mu.hat1 <- exp(mix.WEI.2[["models"]][[1]][["mu.coefficients"]])
sigma.hat1 <- exp(mix.WEI.2[["models"]][[1]][["sigma.coefficients"]])

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

```
mu.hat3 <- exp(mix.WEI.2[["models"]][[3]][["mu.coefficients"]])
sigma.hat3 <- exp(mix.WEI.2[["models"]][[3]][["sigma.coefficients"]])


mu.hat4 <- exp(mix.GA[["models"]][[1]][["mu.coefficients"]])
sigma.hat4 <- exp(mix.GA[["models"]][[1]][["sigma.coefficients"]])

mu.hat5 <-exp(mix.GA[["models"]][[2]][["mu.coefficients"]])
sigma.hat5 <- exp(mix.GA[["models"]][[2]][["sigma.coefficients"]])

mu.hat6 <- exp(mix.GA[["models"]][[3]][["mu.coefficients"]])
sigma.hat6 <- exp(mix.GA[["models"]][[3]][["sigma.coefficients"]])


hist(TotCas, breaks = 224, freq = FALSE, main = "Mixture Weibull distribution K = 3")
lines(seq(min(TotCas), max(TotCas), length = length(TotCas)),
      mix.WEI.2[["prob"]][1]*dWEI(seq(min(TotCas), max(TotCas), length = length(TotCas)),
      mu = mu.hat1, sigma = sigma.hat1), lty = 2, lwd = 3, col = 2)
lines(seq(min(TotCas), max(TotCas), length = length(TotCas)),
      mix.WEI.2[["prob"]][2]*dWEI(seq(min(TotCas), max(TotCas), length = length(TotCas)),
      mu = mu.hat2, sigma = sigma.hat2), lty = 2, lwd = 3, col = 3)
lines(seq(min(TotCas), max(TotCas), length = length(TotCas)),
      mix.WEI.2[["prob"]][3]*dWEI(seq(min(TotCas), max(TotCas), length = length(TotCas)),
      mu = mu.hat3, sigma = sigma.hat3), lty = 2, lwd = 3, col = 4)
lines(seq(min(TotCas), max(TotCas), length = length(TotCas)),
      mix.WEI.2[["prob"]][1]*dWEI(seq(min(TotCas),
      max(TotCas), length = length(TotCas)),
      mu = mu.hat1, sigma = sigma.hat1) + mix.WEI.2[["prob"]][2]*dWEI(seq(min(TotCas),
      max(TotCas),
      length = length(TotCas)),
      mu = mu.hat2, sigma = sigma.hat2) + mix.WEI.2[["prob"]][3]*dWEI(seq(min(TotCas),
      max(TotCas),
      length = length(TotCas)),
      mu = mu.hat3, sigma = sigma.hat3), lty = 1, lwd = 3, col = 1)


    hist(TotCas, breaks = 224, freq = FALSE, main = "Mixture Gamma distribution K = 3")
lines(seq(min(TotCas),max(TotCas),
length=length(TotCas)),
mix.GA[["prob"]][1]*dGA(seq(min(TotCas),
max(TotCas), length=length(TotCas)), mu = mu.hat4,
sigma = sigma.hat4), lty = 2, lwd = 3, col = 2)
lines(seq(min(TotCas),max(TotCas),
length=length(TotCas)),
mix.GA[["prob"]][2]*dGA(seq(min(TotCas),max(TotCas),
length=length(TotCas)),
mu = mu.hat5, sigma = sigma.hat5),
lty = 2, lwd = 3, col = 3)
lines(seq(min(TotCas),max(TotCas), length=length(TotCas)),
mix.GA[["prob"]][3]*dGA(seq(min(TotCas),
max(TotCas), length=length(TotCas)),
```

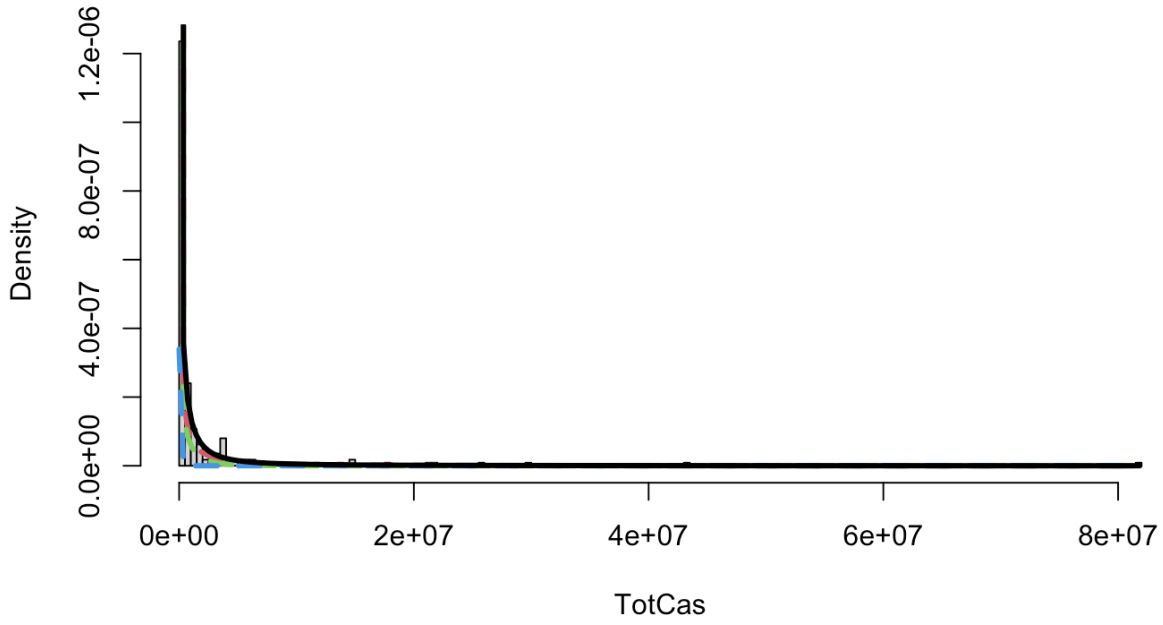


Figure 15: Mixture Weibull Distributions K=3

```
mu = mu.hat6, sigma = sigma.hat6),
lty = 2, lwd = 3, col = 4)
lines(seq(min(TotCas), max(TotCas),
length=length(TotCas)),
mix.GA[["prob"]][1]*dGA(seq(min(TotCas),
max(TotCas), length=length(TotCas)),
mu = mu.hat4, sigma = sigma.hat4)+
mix.GA[["prob"]][2]*dGA(seq(min(TotCas),
max(TotCas), length=length(TotCas)),
mu = mu.hat5, sigma = sigma.hat5) +
mix.GA[["prob"]][3]*dGA(seq(min(TotCas),
max(TotCas), length=length(TotCas)),
mu = mu.hat6, sigma = sigma.hat6),
lty = 1, lwd = 3, col = 1)
```

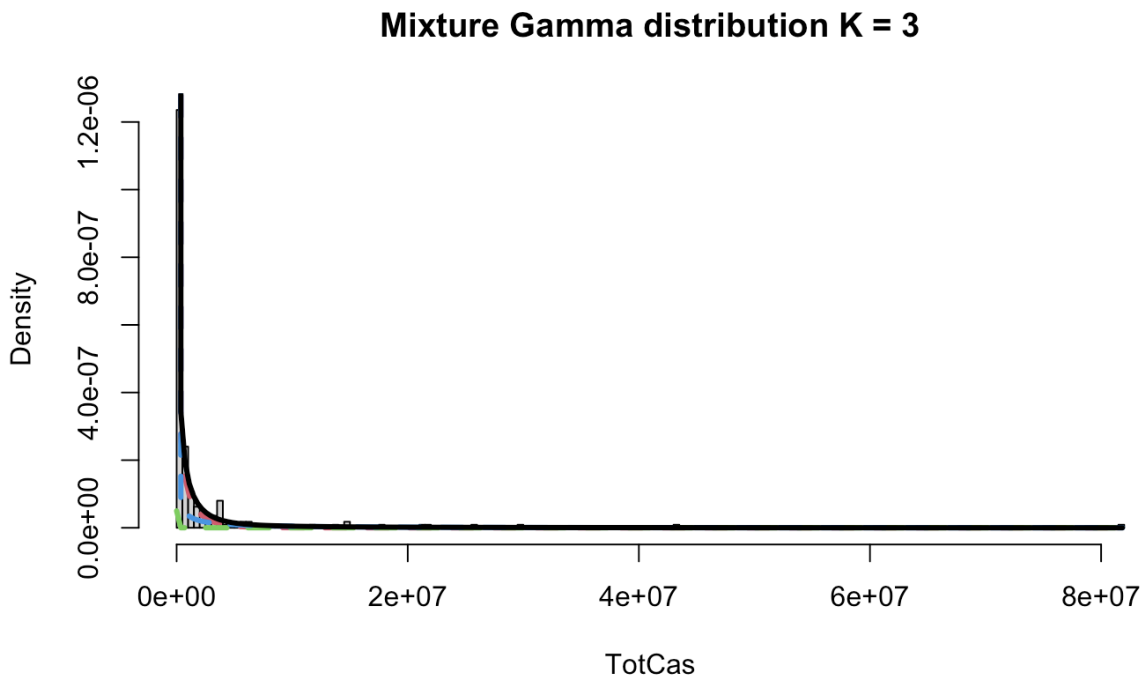


Figure 16: Mixture GAMMA Distributions K=3

3.4 Total Deaths

Description This is another continuous variable from 0 to infinite.

```
     summary(Total.Deaths)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0     189    1965   27448   13660 1008222
> boxplot(Total.Deaths, main = "Boxplot of total deaths")
> unique(Total.Deaths)
  [1]    7671    3492    6874     153    1900
  9     135  128065    8617     212    6384   15985    9697
 [14]     788    1471   29122     375    6844
 30826     656     163     128      12   21896   15719    2686
 [27]  660269      62     213   36568     382
 38     401    3054    1927   37690     113      33      24
 [40]     191     156   56750    4638  139660
 160     385       0    8308   15601    8514     267     947
 [53]   39720    5762     189      63    4375
 1337   35421   24417    4120     183     103    2468    1394
 [66]    7504      28     834    3178  142506
 394     646     303     365   16756  130563    1445     101
 [79]   27684      21     218     843   17325
 440     170    1226     833   10880    8172   45510  521388
 [92]  155288  140315   25173    6786      84
 10530  159784     796    2893   28248   14003   13660    5648
[105]      13    2554    2991     679    5643
10315     697     294    6419    8907    1037    1388    2626
```

```
[118]   35099     298     728     641     909
982     968     187  323212   11446      54    2177    2705
[131]       2   16060    2200   19433    4019
11951   22016     311     350     224     308    3142    9228
[144]    2518    4251   30361       6    5351
8170     640   18731  212328   59343  115345   21693     677
[157]     709   65090  369708    1458   17235
43        1      73    9048    1965   15825     164     125
[170]    1276      86   19417    6501     133
1348  100050     138  102541   16481     106    4907    1325
[183]   18331   13715    3144     853     124
800   25418     130     272    3756   28323   98157      36
[196]    2302    3595  165570  107980
7166 1008222    1637    5686   42600       7
2143    3967    5446
> length(unique(Total.Deaths))
[1] 208
> hist(Total.Deaths,breaks = 208, col = "blue",
main = "Histogram of total deaths",freq=FALSE)
> box()
> lines(density(Total.Deaths), col= "black")
> skewness(Total.Deaths)
[1] 6.900123
> kurtosis(Total.Deaths)
[1] 59.50997
```

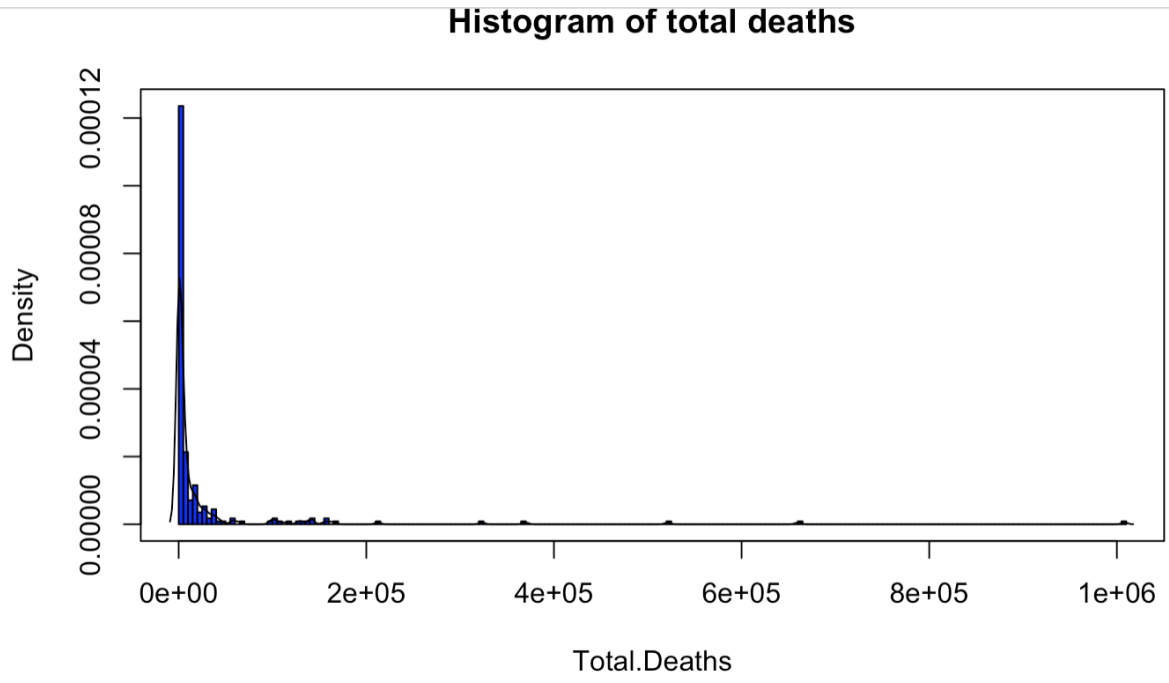


Figure 17: Histogram of Total Deaths

Now the next step is to fit some single models to this variable

```
    Deaths <- Total.Deaths[Total.Deaths!=0]

fit2.GA <- histDist(Deaths, family = GA, nbins = 208,
main = "Gamma distribution")
fit2.EXP <- histDist(Deaths, family = EXP, nbins = 208,
main = "Exponential distribution")
fit2.IG <- histDist(Deaths, family = IG, nbins = 208,
main = "Inversian Gaussian distribution")
fit2.LOGNO <- histDist(Deaths, family = LOGNO, nbins = 208,
main = "Log-Normal distribution")
fit2.WEI <- histDist(Deaths, family = WEI, nbins = 208,
main = "Weibull distribution")
fit2.GIG <- histDist(Deaths, family = GIG, nbins = 208,
main = "Generalized Inversed Gaussian")
```

Figure 18: BoxPlot of Total Deaths

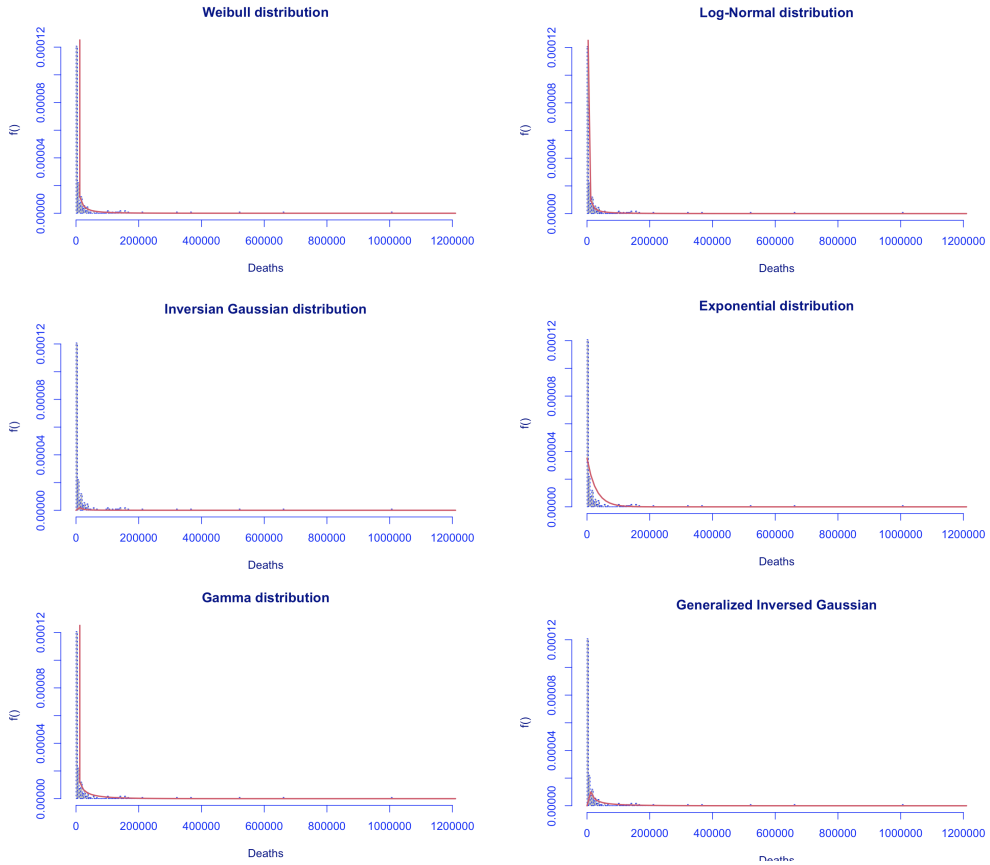


Figure 19: fitting models of Total Deaths

	num_of_par <dbl>	logLikelihood <dbl>	AIC <dbl>	BIC <dbl>
Gamma	2	-2187.089	4378.177	4384.937
Exponential	1	-2442.608	4887.215	4890.595
Inversian Gaussian	2	-2352.460	4708.921	4715.681
Log-Normal	2	-2157.372	4318.744	4325.504
Weibull	2	-2161.589	4327.179	4333.939
Generalized Inversian Gaussian	3	-2166.239	4338.477	4348.617

Figure 20: dataframe of Total Deaths

To fit this distribution, we are going to compare some models.

```
  data.frame(row.names = c("Gamma", "Exponential", "Inversian Gaussian","Log-Normal",
  "Weibull","Generalized Inversian Gaussian"),
         "num_of_par"=c(fit2.GA$df.fit,
         fit2.EXP$df.fit,fit2.IG$df.fit,
         fit2.LOGNO$df.fit,fit2.WEI$df.fit,
         fit2.GIG$df.fit),
         "logLikelihood"=c(logLik(fit2.GA),
         logLik(fit2.EXP), logLik(fit2.IG),
         logLik(fit2.LOGNO),
         logLik(fit2.WEI), logLik(fit2.GIG)),
         "AIC"= c(AIC(fit2.GA), AIC(fit2.EXP),
         AIC(fit2.IG), AIC(fit2.LOGNO),
         AIC(fit2.WEI), AIC(fit2.GIG)),
         "BIC"=c(fit2.GA$sbc, fit2.EXP$sbc,
         fit2.IG$sbc, fit2.LOGNO$sbc, fit2.WEI$sbc,
         fit2.GIG$sbc))
         "MU" =c(fitted(fit2.GA,"mu")[1],
         fitted(fit2.EXP,"mu")[1],                    fitted(fit2.IG,"mu")[1],
         fitted(fit2.LOGNO,"mu")[1],
         fitted(fit2.WEI,"mu")[1],
         fitted(fit2.GIG,"mu")[1])
         "SIGMA"= c(fitted(fit2.GA,"sigma")[1],
         NA ,
         fitted(fit2.IG,"sigma")[1],
         fitted(fit2.LOGNO,"sigma")[1],
         fitted(fit2.WEI,"sigma")[1],
                  fitted(fit2.GIG,"sigma")[1])
         "NU" =c(fitted(fit2.GIG,"nu")[1],
         NA,NA,NA,NA,NA)


   > mix2.LOGNO <- gamlssMXfits(n=5,Deaths~1,
   family = LOGNO, K=3, data = NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
```

```
> mix2.LOGNO$aic
[1] 4319.316
> mix2.LOGNO$sbc
[1] 4346.355
> mix2.LOGNO$prob
[1] 0.07048503 0.26456118 0.66495379

hist(Deaths, breaks = 208, freq = FALSE,
main = "Mixture Log-Normal distribution K = 3")
lines(seq(min(Deaths),
max(Deaths),
length=length(Deaths)),
mix2.LOGNO[["prob"]][1]*dLOGNO(seq(min(Deaths),
max(Deaths),
length=length(Deaths)), mu = mu.hat7,
sigma = sigma.hat7), lty = 2, lwd = 3, col = 2)
lines(seq(min(Deaths),max(Deaths),
length=length(Deaths)),
mix2.LOGNO[["prob"]][2]*dLOGNO(seq(min(Deaths),
max(Deaths),
length=length(Deaths)), mu = mu.hat8,
sigma = sigma.hat8),
lty = 2, lwd = 3, col = 3)
lines(seq(min(Deaths),max(Deaths),
length=length(Deaths)),
mix2.LOGNO[["prob"]][3]*dLOGNO(seq(min(Deaths),
max(Deaths), length=length(Deaths)), mu = mu.hat9,
sigma = sigma.hat9), lty = 2, lwd = 3, col = 4)
lines(seq(min(Deaths), max(Deaths),
length=length(Deaths)),
mix2.LOGNO[["prob"]][1]*dLOGNO(seq(min(Deaths),
max(Deaths),
length=length(Deaths)),
mu = mu.hat7, sigma = sigma.hat7)+ mix2.LOGNO[["prob"]][2]*dLOGNO(seq(min(Deaths),
max(Deaths), length=length(Deaths)),
mu = mu.hat8, sigma = sigma.hat8)
+ mix2.LOGNO[["prob"]][3]*dLOGNO(seq(min(Deaths),
max(Deaths),
length=length(Deaths)), mu = mu.hat9,
sigma = sigma.hat9), lty = 1, lwd = 3, col = 1)
```

3.5 Univariate Analysis of Total cases 1M pop

"Total cases per 1 million population" is a measure used to assess the spread of a particular condition, such as confirmed cases of a disease, in relation to the size of a given area's population. To calculate this measure, the total number of cases of interest (e.g., confirmed cases of a disease) is considered and divided by the population size, usually expressed in millions. This value is then multiplied by 1 million to make it more easily interpretable. This type of measure is useful because it provides a standardized way to compare the spread of a disease across areas with different population sizes. It allows for the relative level of disease spread to be evaluated more fairly, regardless of differences in population size among different regions or countries.

```
    > Cases <- Tot.Cases..1M.pop
    > summary(Cases)
    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      9   11384   88987  136900  223335  696044
      > length(unique(Cases))
[1] 225
> hist(Cases, breaks = 225, col = "blue", freq = FALSE,
main = "Histogram of Total cases of 1M pop")
> lines(density(Cases), col = "red")
> skewness(Cases)
[1] 1.083991
> kurtosis(Cases)
[1] 3.437767
> boxplot(Cases, main = "Boxplot of Total cases 1M Population")
```

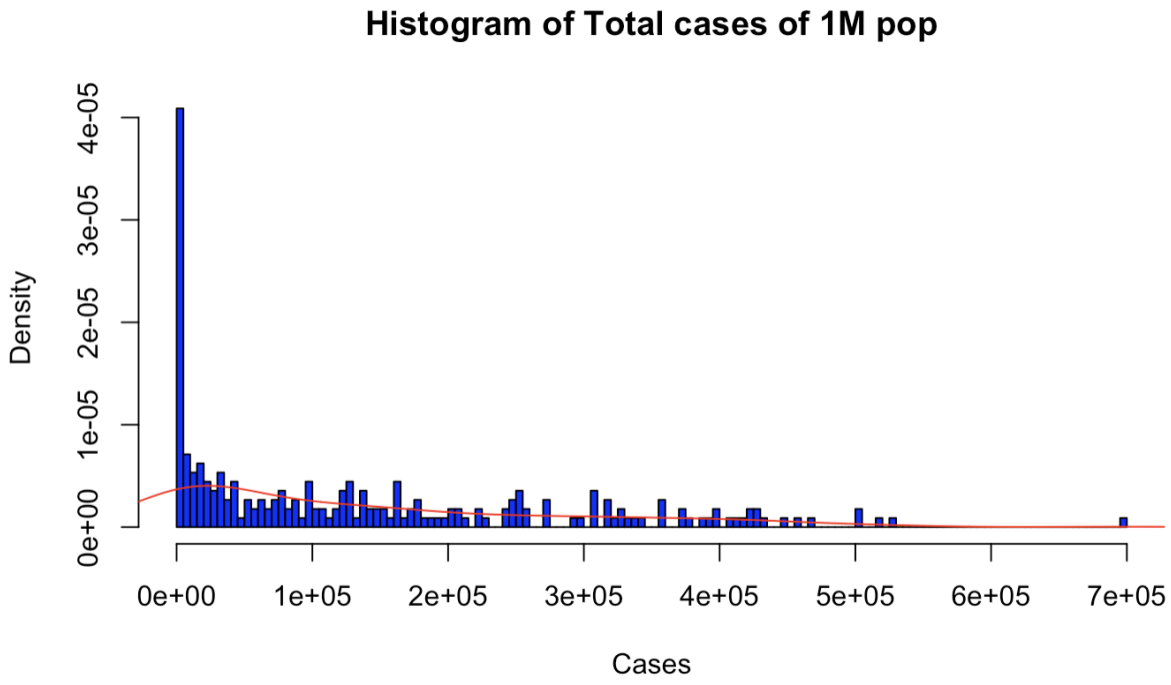


Figure 21: Histogram of total cases 1M pop.

```
    > sd(Cases) [1] 145060.3 > var(Cases) [1] 21042502325
```

Standard Deviation (SD): The standard deviation measures the amount of variation or dispersion of a set of values. In this case, the standard deviation of "Cases" is approximately 145,060.3.

Variance: The variance is another measure of the spread of a set of values. It is the square of the standard deviation. The variance of "Cases" is approximately 21,042,502,325.

Skewness: Skewness measures the asymmetry of the probability distribution of a real-valued random variable about its mean. A skewness value of 1.083991 suggests that the distribution of "Cases" is moderately skewed to the right.

Kurtosis: Kurtosis measures the "tailedness" of the probability distribution of a real-valued random variable. A kurtosis value of 3.437767 indicates that the distribution of "Cases" has heavier tails compared to a normal distribution (which has a kurtosis of 3 for its standard form).

- The standard deviation and variance indicate the extent of variability in the "Cases" data.
- The positive skewness suggests that there is a longer right tail in the distribution of "Cases," indicating that there may be some extremely high values.
- The kurtosis value indicates that the distribution of "Cases" is leptokurtic, meaning it has heavier tails and more extreme values than a normal distribution.

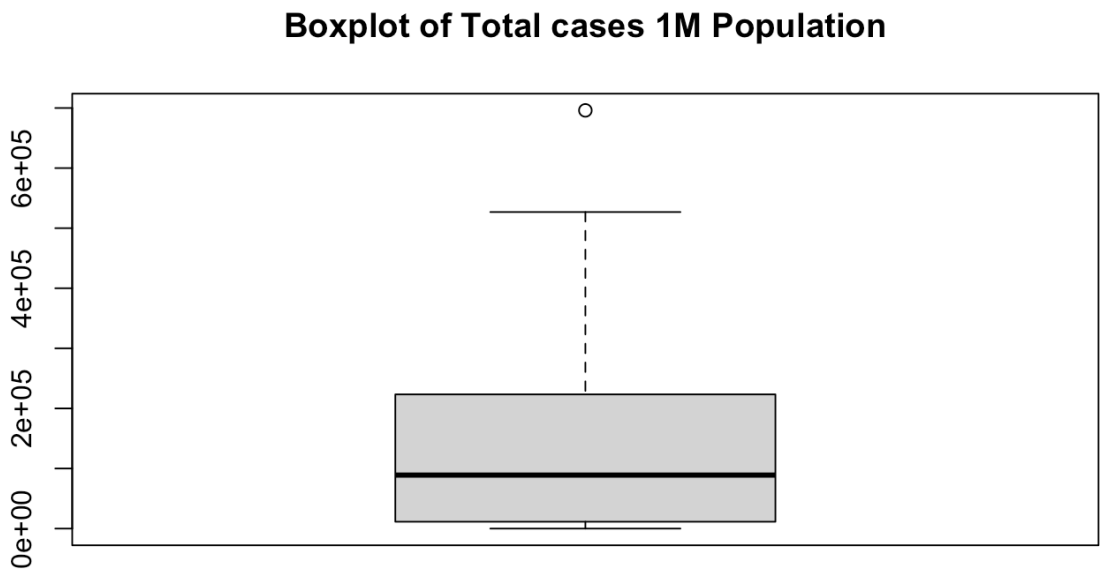


Figure 22: Boxplot of total cases 1M pop.

```
    > boxplot.stats(Cases)$out
[1] 696044
```

To fit this distribution, we are going to compare some models.

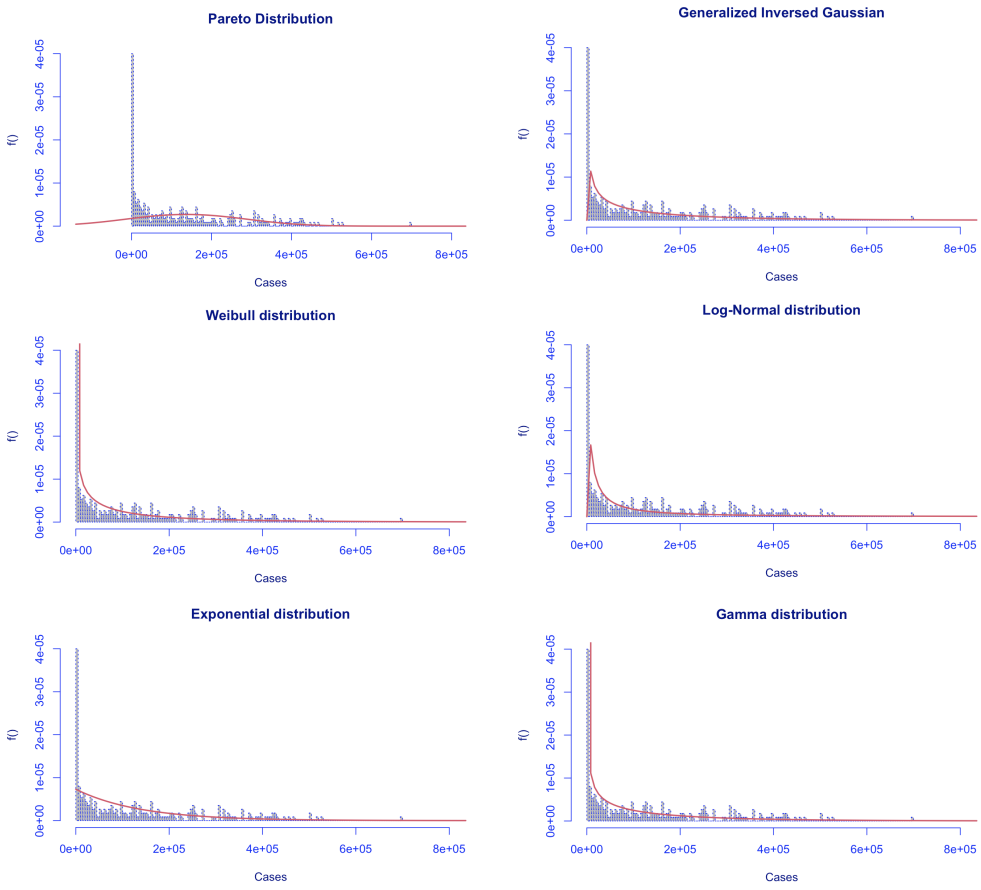


Figure 23: fitting models of Total cases 1M Pop.

```
fit3.GA <- histDist(Cases, family = GA, nbins = 225,
main = "Gamma distribution")
fit3.EXP <- histDist(Cases, family = EXP, nbins = 225,
main = "Exponential distribution")
fit3.IG <- histDist(Cases, family = IG, nbins = 225,
main = "Inversian Gaussian distribution")
fit3.LOGNO <- histDist(Cases, family = LOGNO, nbins = 225,
main = "Log-Normal distribution")
fit3.WEI <- histDist(Cases, family = WEI, nbins = 225,
main = "Weibull distribution")
fit3.GIG <- histDist(Cases, family = GIG, nbins = 225,
main = "Generalized Inversed Gaussian")
```

- **Log-Normal Distribution:** Useful for positive data with a right-skewed distribution.
- **Exponential Distribution:** Suitable for data describing times between events in a process with a constant rate.
- **Gamma Distribution:** Suitable for data that are the sum of exponentially distributed random variables, useful for asymmetric data.
- **Weibull Distribution:** Often used to describe the life of a product, it can adapt to various forms of data.

	num_of_par <dbl>	logLikelihood <dbl>	AIC <dbl>	BIC <dbl>	MU <dbl>	SIGMA <dbl>	NU <dbl>
Gamma	2	-2849.182	5702.363	5709.196	136899.41844	1.36037275	0.52384
Exponential	1	-2886.077	5774.154	5777.570	136900.37333	*NA*	*NA*
Inversian Gaussian	2	-3144.615	6293.231	6300.063	136899.82730	0.03203173	*NA*
Log-Normal	2	-2889.873	5783.746	5790.578	10.66544	2.13723309	*NA*
Weibull	2	-2856.219	5716.437	5723.269	109582.93148	0.67134362	*NA*
Generalized Inversian Gaussian	3	-2848.733	5703.467	5713.715	136900.45086	12.71316767	*NA*

Figure 24: dataframe of total cases 1M pop.

Now, is possible to compare AIC and BIC in order to try to find the best model for this variable. Looking at the values of BIC and AIC, the distribution that fits best would seem to be the Gamma distribution.

```
    mix3.GA <- gamlssMXfits(n=5, Cases~1, family = GA, K=3, data = NULL)
    mu.hat10 <- exp(mix3.GA[["models"]][[1]][["mu.coefficients"]])
sigma.hat10 <- exp(mix3.GA[["models"]][[1]][["sigma.coefficients"]])

mu.hat11 <-exp(mix3.GA[["models"]][[2]][["mu.coefficients"]])
sigma.hat11 <- exp(mix3.GA[["models"]][[2]][["sigma.coefficients"]])

mu.hat12 <- exp(mix3.GA[["models"]][[3]][["mu.coefficients"]])
sigma.hat12 <- exp(mix3.GA[["models"]][[3]][["sigma.coefficients"]])

mix3.GA
Mixing Family:  c("GA", "GA", "GA")
Fitting method: EM algorithm
Call:  gamlssMX(formula = Cases ~ 1, family = GA, K = 3, data = NULL)
Mu Coefficients for model: 1
(Intercept)
       8.27
Sigma Coefficients for model: 1
(Intercept)
     0.1029
Mu Coefficients for model: 2
(Intercept)
      12.49
Sigma Coefficients for model: 2
(Intercept)
    -0.6947
Mu Coefficients for model: 3
(Intercept)
      11.11
Sigma Coefficients for model: 3
(Intercept)
    -0.1164
Estimated probabilities: 0.2308561 0.4229741 0.3461697
Degrees of Freedom for the fit: 8 Residual Deg. of Freedom   217
Global Deviance:     5664.42
            AIC:     5680.42
            SBC:     5707.75
```

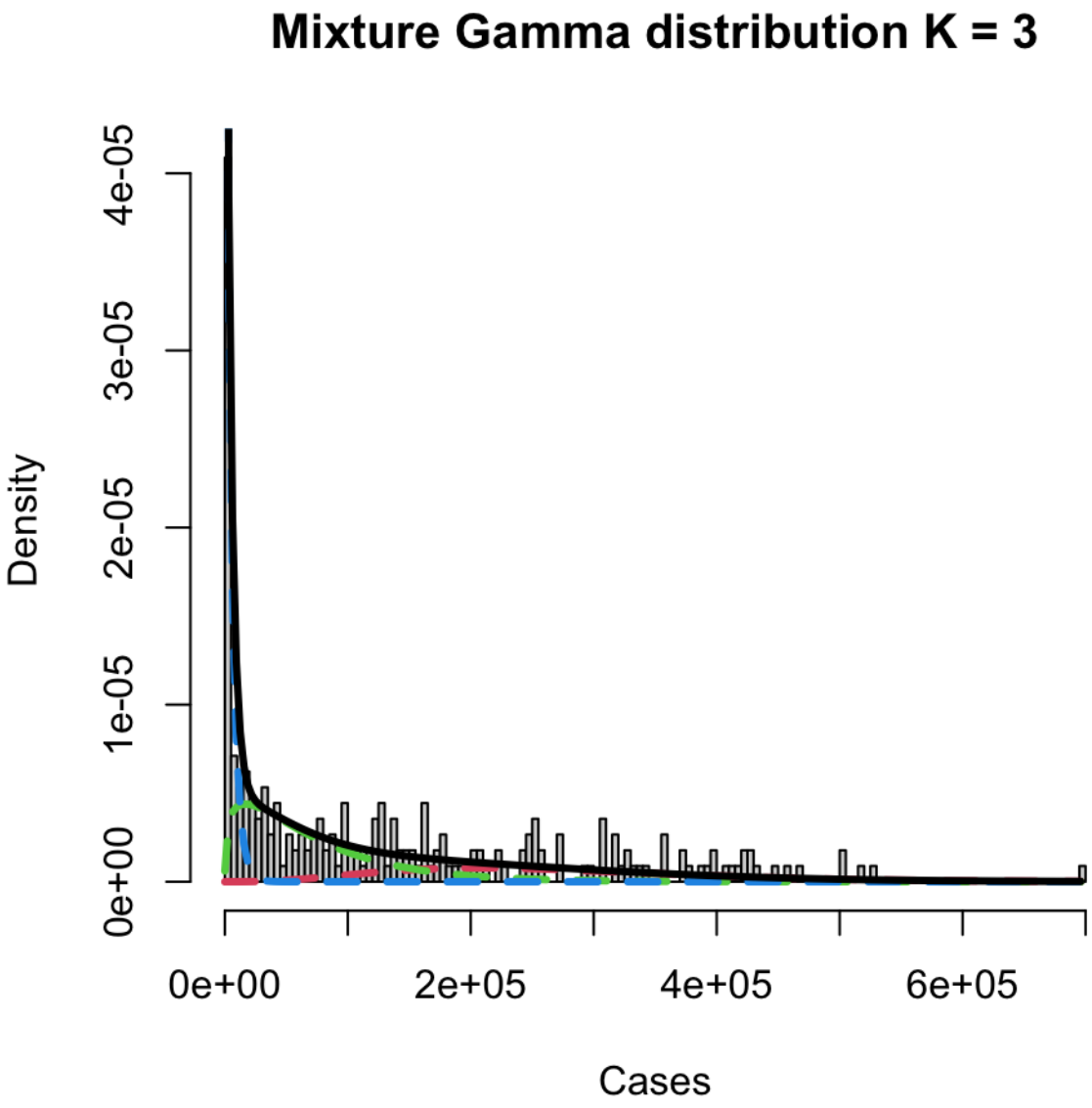


Figure 25: Mixture Gamma distribution K=3

```
hist(Cases, breaks = 225, freq = FALSE, main = "Mixture Gamma distribution K = 3")
lines(seq(min(Cases), max(Cases), length = length(Cases)),
      mix3.GA[["prob"]][1] * dGA(seq(min(Cases), max(Cases),
      length = length(Cases)), mu = mu.hat10, sigma = sigma.hat10),
      lty = 2, lwd = 3, col = 2)
lines(seq(min(Cases), max(Cases), length = length(Cases)),
      mix3.GA[["prob"]][2] * dGA(seq(min(Cases), max(Cases),
      length = length(Cases)), mu = mu.hat11, sigma = sigma.hat11),
      lty = 2, lwd = 3, col = 3)
lines(seq(min(Cases), max(Cases), length = length(Cases)),
      mix3.GA[["prob"]][3] * dGA(seq(min(Cases), max(Cases),
      length = length(Cases)), mu = mu.hat12, sigma = sigma.hat12),
      lty = 2, lwd = 3, col = 4)
lines(seq(min(Cases), max(Cases), length = length(Cases)),
      mix3.GA[["prob"]][1] * dGA(seq(min(Cases), max(Cases),
      length = length(Cases)), mu = mu.hat10, sigma = sigma.hat10) +
      mix3.GA[["prob"]][2] * dGA(seq(min(Cases), max(Cases),
      length = length(Cases)), mu = mu.hat11, sigma = sigma.hat11) +
      mix3.GA[["prob"]][3] * dGA(seq(min(Cases), max(Cases),
      length = length(Cases)), mu = mu.hat12, sigma = sigma.hat12),
      lty = 1, lwd = 3, col = 1)
```

3.6 Univariate Analysis of Total deaths of 1M Pop

This variable show us the number of deaths related to 1 million population as the previous variable.

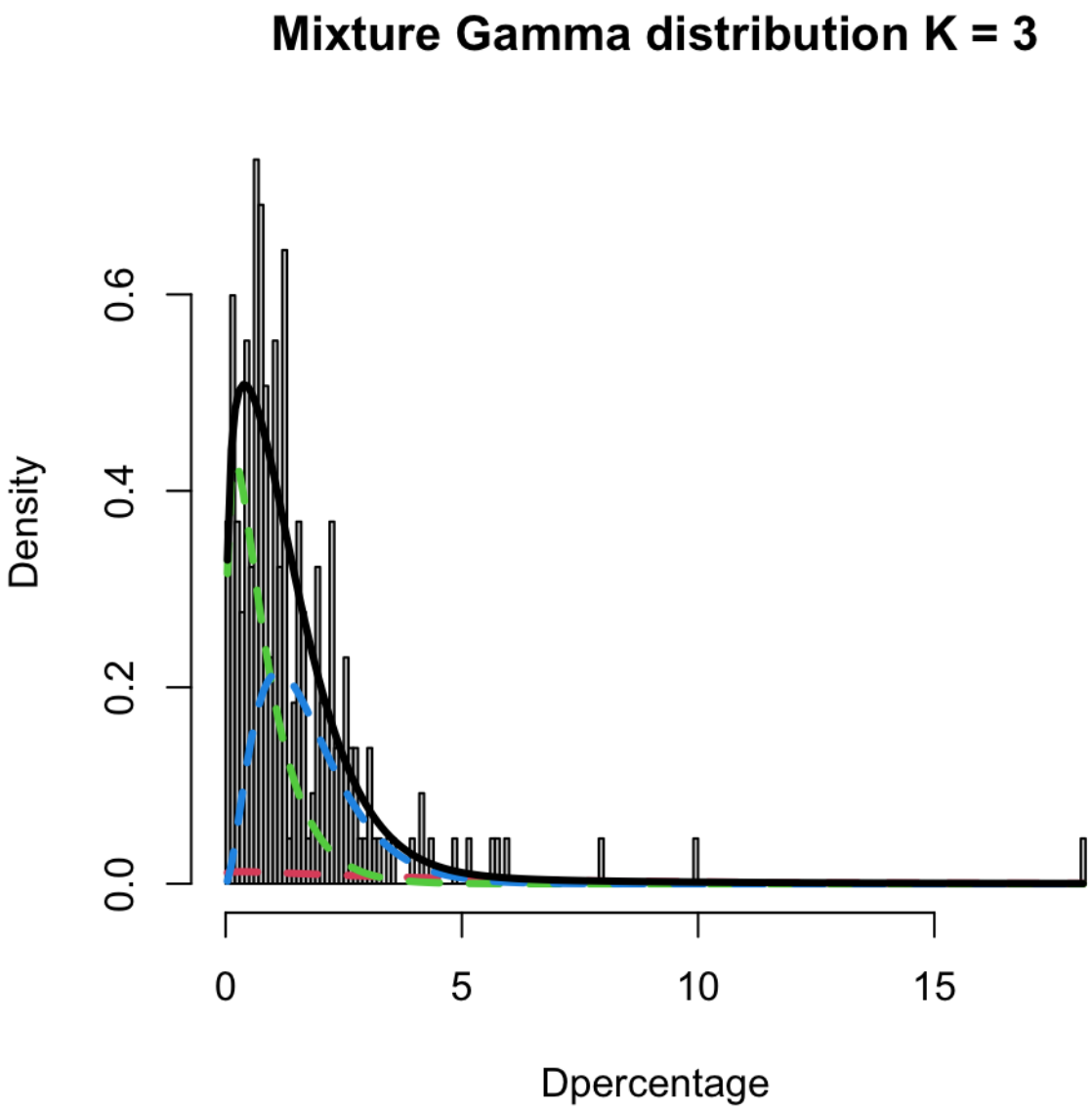


Figure 26: Histogram of 1M deaths

```
> skewness(Deaths2)
[1] 1.371802
> kurtosis(Deaths2)
[1] 4.866557
```

```
    > boxplot.stats(Deaths2)
$stats
[1]    2  172  759 1832 4306
$n
[1] 217
$conf
[1] 580.9527 937.0473
$out
[1] 4844 5333 4732 4430 6286
```

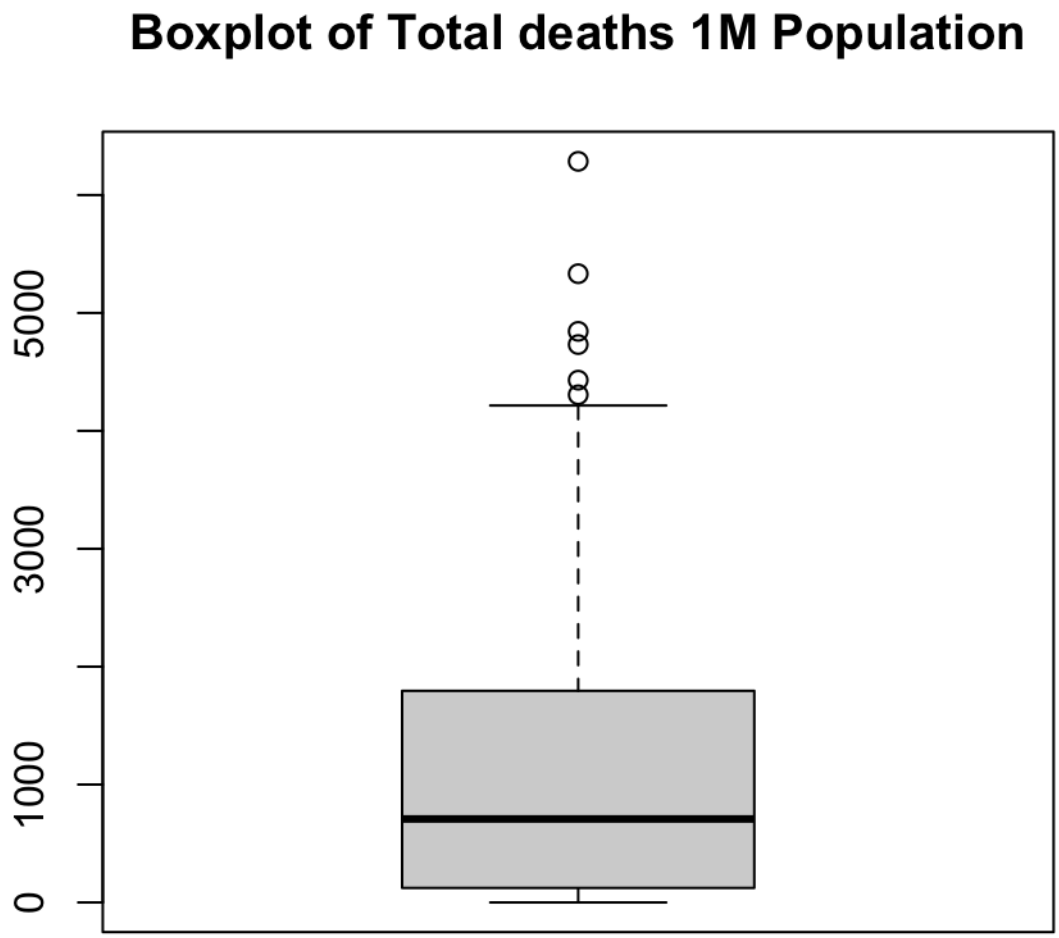


Figure 27: Boxplot of 1M deaths

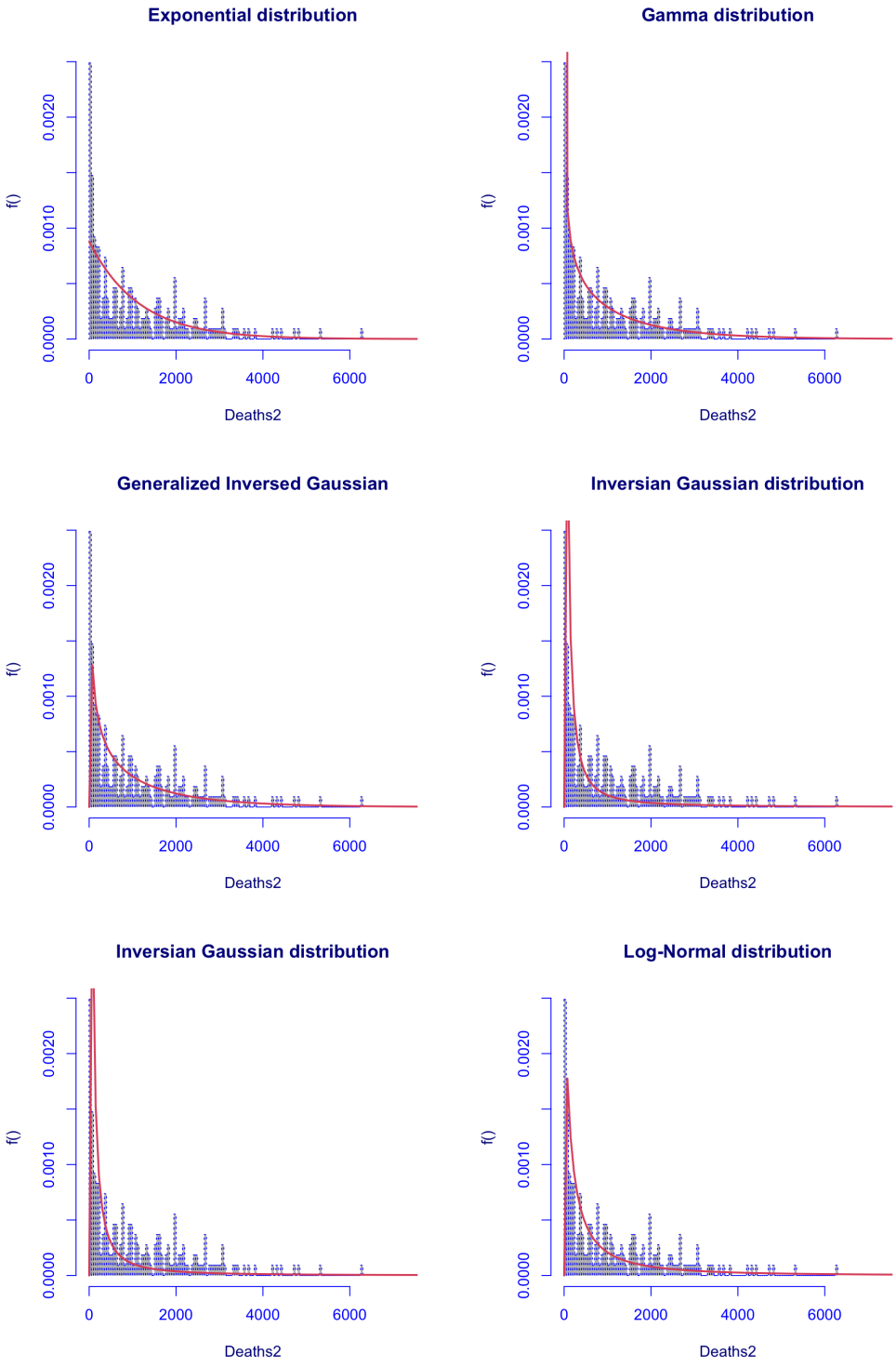


Figure 28: fitting models of Total deaths 1M Pop.

	num_of_par	logLikelihood	AIC	BIC
Gamma	2	-1730.153	3464.306	3471.066
Exponential	1	-1743.872	3489.745	3493.125

```
Inversian Gaussian                      2     -1838.952 3681.904 3688.664
Log-Normal                              2     -1756.416 3516.832 3523.592
Weibull                                 2     -1732.779 3469.559 3476.318
Generalized Inversian Gaussian          3     -1728.585 3463.170 3473.310
mix5.GA<-gamlssMXfits(n=5, Deaths2 ~1, family = GA, K=3, data = NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
> mix5.GA$aic
[1] 3458.512
> mix5.GA$sbc
[1] 3485.551
> mix5.GA$prob
[1] 0.2143384 0.3587304 0.4269312
```

```
hist(Deaths2, breaks = 200, freq = FALSE, main = "Mixture Gamma distribution K = 3")
lines(seq(min(Deaths2),max(Deaths2),
length=length(Deaths2)), mix5.GA[["prob"]][1]*dGA(seq(min(Deaths2),max(Deaths2),
length=length(Deaths2)), mu = mu.hat10, sigma = sigma.hat10),
lty = 2, lwd = 3, col = 2)
lines(seq(min(Deaths2),max(Deaths2),
length=length(Deaths2)), mix5.GA[["prob"]][2]*dGA(seq(min(Deaths2),max(Deaths2),
length=length(Deaths2)), mu = mu.hat11, sigma = sigma.hat11),
lty = 2, lwd = 3, col = 3)
lines(seq(min(Deaths2),max(Deaths2),
length=length(Deaths2)), mix5.GA[["prob"]][3]*dGA(seq(min(Deaths2),max(Deaths2),
length=length(Deaths2)), mu = mu.hat12, sigma = sigma.hat12),
lty = 2, lwd = 3, col = 4)
lines(seq(min(Deaths2), max(Deaths2),
length=length(Deaths2)),
mix5.GA[["prob"]][1]*dGA(seq(min(Deaths2),
max(Deaths2), length=length(Deaths2)), mu = mu.hat10,
sigma = sigma.hat10)+
mix5.GA[["prob"]][2]*dGA(seq(min(Deaths2),
max(Deaths2), length=length(Deaths2)), mu = mu.hat11,
sigma = sigma.hat11)
+ mix5.GA[["prob"]][3]*dGA(seq(min(Deaths2),
max(Deaths2), length=length(Deaths2)),
mu = mu.hat12, sigma = sigma.hat12),
lty = 1, lwd = 3, col = 1)
```

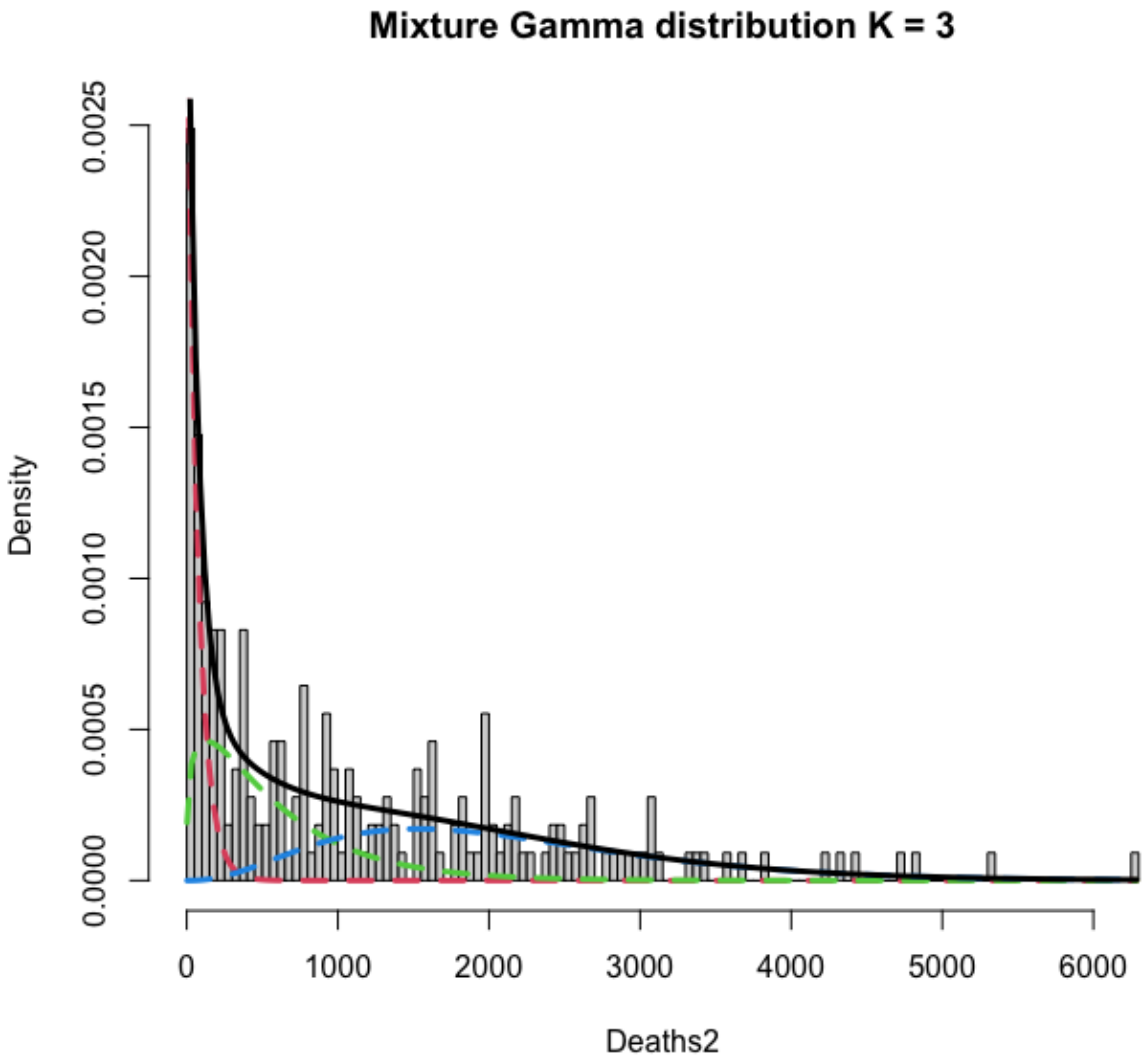


Figure 29: Mixture gamma distribution

3.7 Univariate Analysis of Death percentage

Total Death per Total Covid 19 cases reported. It is a number between 0 and 100.

```
    summary(Dpercentage)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.5113  1.0369  1.4441  1.9770 18.1518
 > length(unique(Dpercentage))
[1] 218
> sd(Dpercentage)
[1] 1.750967
> var(Dpercentage)
[1] 3.065885
```

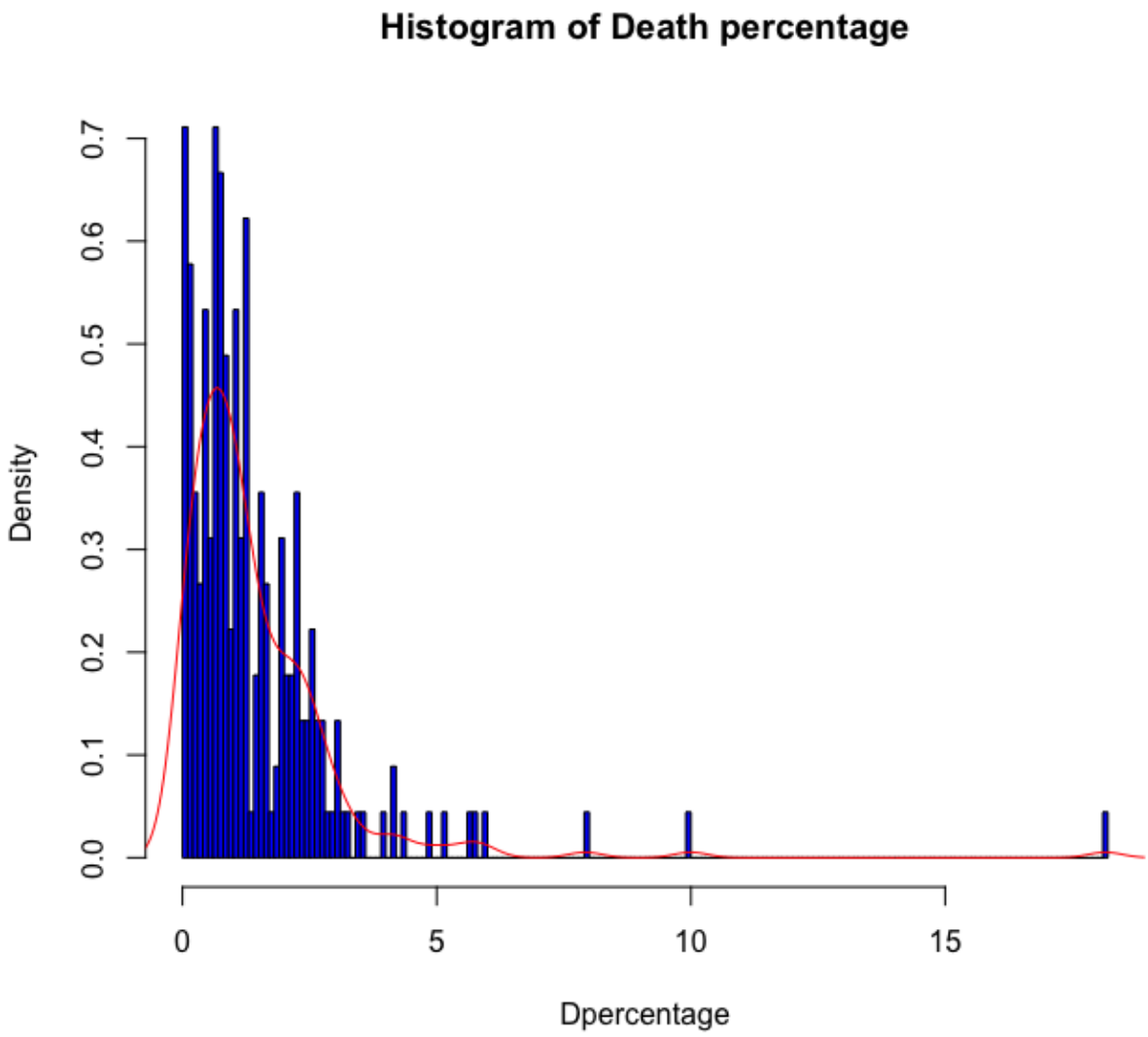


Figure 30: Histogram of deaths percentage

```
    > skewness(Dpercentage)
    [1] 5.016905
    > kurtosis(Dpercentage)
    [1] 42.34555
    Dpercentage <- Death.percentage[Death.percentage!=0]
fit6.GA <- histDist(Dpercentage, family = GA, nbins = 218,
main = "Gamma distribution")
fit6.EXP <- histDist(Dpercentage, family = EXP, nbins = 218,
main = "Exponential distribution")
fit6.IG <- histDist(Dpercentage, family = IG, nbins = 218,
main = "Inversian Gaussian distribution")
fit6.LOGNO <- histDist(Dpercentage, family = LOGNO, nbins = 218,
main = "Log-Normal distribution")
fit6.WEI <- histDist(Dpercentage, family = WEI, nbins = 218,
```

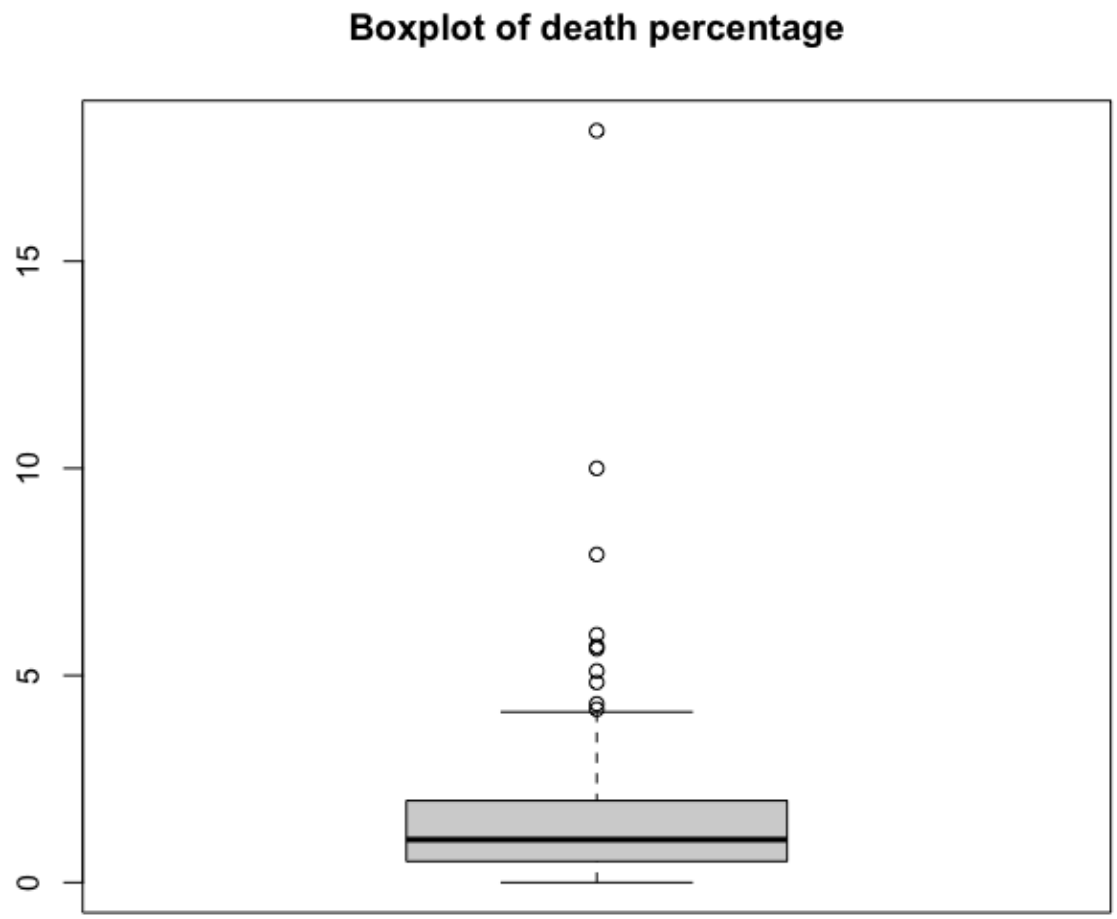


Figure 31: boxplot of deaths percentage

```
main = "Weibull distribution")
fit6.GIG <- histDist(Dpercentage, family = GIG, nbins = 218,
main = "Generalized Inversed Gaussian")
```

```
                                num_of_par logLikelihood  AIC  BIC
Gamma                                2     -302.0272 608.0544 614.8142
Exponential                          1     -304.6043 611.2087 614.5886
Inversian Gaussian                   2     -327.4961 658.9922 665.7520
Log-Normal                           2     -304.6083 613.2167 619.9765
Weibull                              2     -303.7599 611.5199 618.2796
Generalized Inversian Gaussian       3     -300.5573 607.1145 617.2542

> mix6.GA <- gamlssMXfits(n=5, Dpercentage~1, K=3,
```

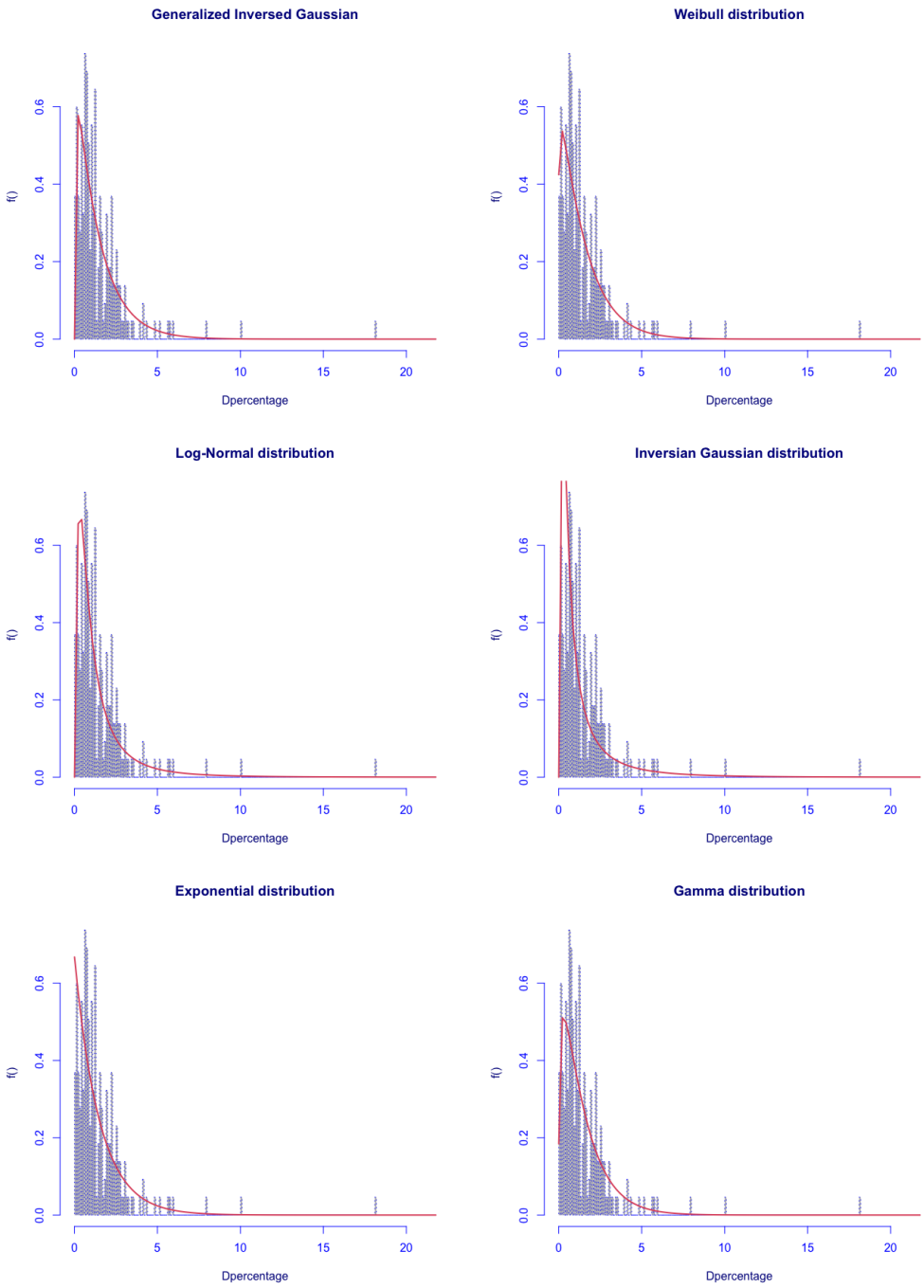


Figure 32: fitting models of deaths percentage

```
family = GA, data = NULL )
model= 1
model= 2
model= 3
model= 4
model= 5
> mix6.GA$aic
```

```
[1] 604.0289
> mix6.GA$sbc
[1] 631.0681
> mix6.GA$prob
[1] 0.56239035 0.36850889 0.06910076
hist(Dpercentage, breaks = 218, freq = FALSE, main = "Mixture Gamma distribution K = 3")
lines(seq(min(Dpercentage),max(Dpercentage),
length=length(Dpercentage)),
mix6.GA[["prob"]][1]*dGA(seq(min(Dpercentage),
max(Dpercentage), length=length(Dpercentage)),
mu = mu.hat10,
sigma = sigma.hat10), lty = 2, lwd = 3, col = 2)
lines(seq(min(Dpercentage),max(Dpercentage),
length=length(Dpercentage)),
mix6.GA[["prob"]][2]*dGA(seq(min(Dpercentage),
max(Dpercentage), length=length(Dpercentage)),
mu = mu.hat11,
sigma = sigma.hat11), lty = 2, lwd = 3, col = 3)
lines(seq(min(Dpercentage),max(Dpercentage),
length=length(Dpercentage)),
mix6.GA[["prob"]][3]*dGA(seq(min(Dpercentage),
max(Dpercentage), length=length(Dpercentage)), mu = mu.hat12,
sigma = sigma.hat12), lty = 2, lwd = 3, col = 4)
lines(seq(min(Dpercentage),
max(Dpercentage),
length=length(Dpercentage)),
mix6.GA[["prob"]][1]*dGA(seq(min(Dpercentage),
max(Dpercentage), length=length(Dpercentage)), mu = mu.hat10,
sigma = sigma.hat10)+
mix6.GA[["prob"]][2]*dGA(seq(min(Dpercentage),
max(Dpercentage), length=length(Deaths2)),
mu = mu.hat11, sigma = sigma.hat11) +
mix6.GA[["prob"]][3]*dGA(seq(min(Dpercentage),
max(Dpercentage), length=length(Dpercentage)),
mu = mu.hat12, sigma = sigma.hat12),
lty = 1, lwd = 3, col = 1)
```

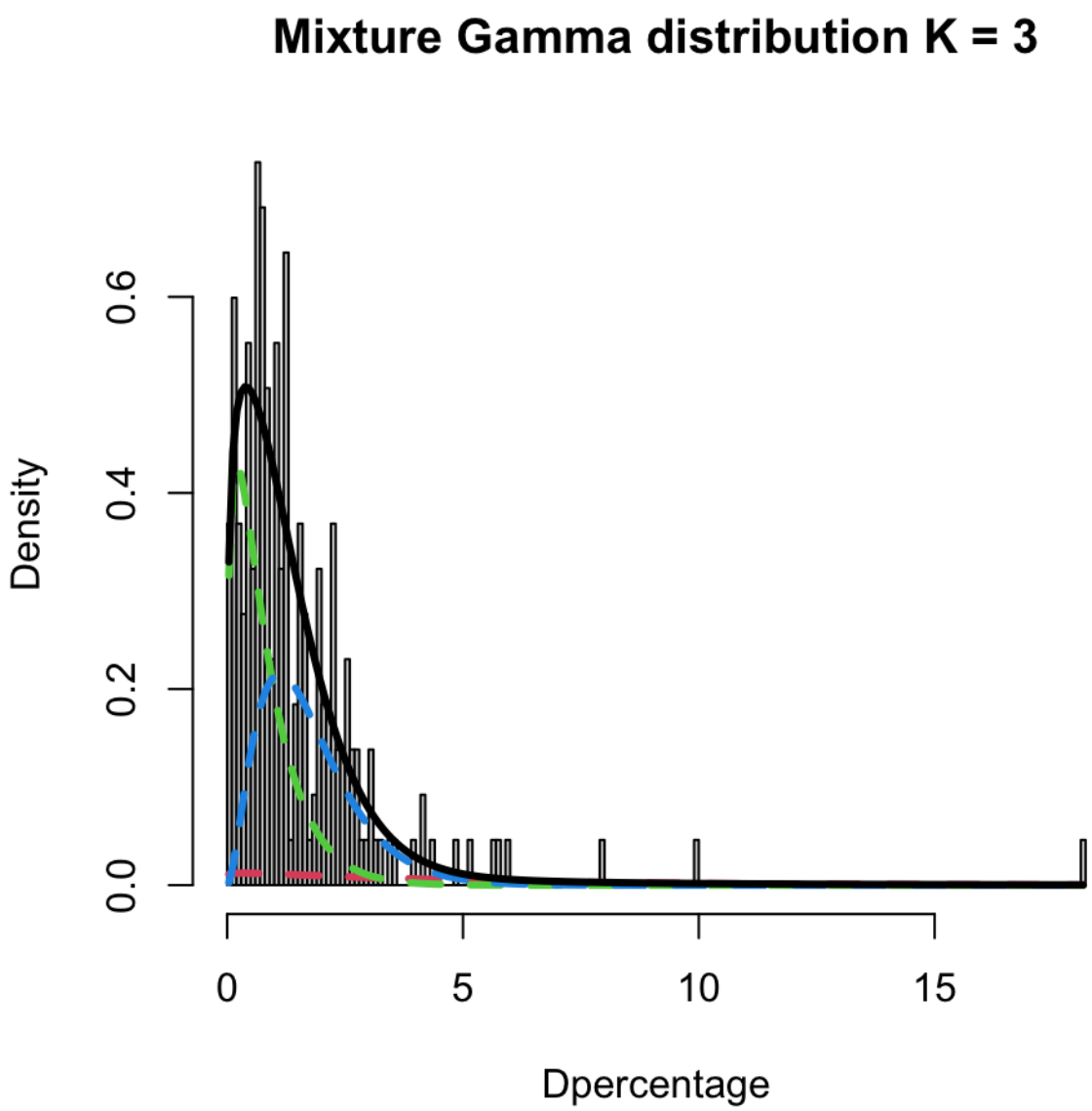


Figure 33: Mixture gamma distribution K=3

4 Multivariate Analysis

In this chapter i will use 2 types of analysis:

- **Principal Component Analysis**
- **Cluster Analysis**

Principal Component Analysis PCA allows us to summarize and to visualize the information in a data set containing statistical units described by multiple correlated quantitative variables. Each variable could be considered as a different dimension. The goal of PCA is to explain most of the variability in the data with

a smaller number of q of variables than the original data set. The information in a given data set corresponds to the total variation (variability) it contains. The goal of PCA is to identify directions (or PCs) along which the variation in the data is maximal. In other words, PCA reduces the dimensionality of multivariate data to two or three PCs, that can be visualized graphically, with minimal loss of information.

```
str(dataset)
numerical_data <- dataset[, -1]
scaled_df<-apply(numerical_data, 2,scale)
scaled<-data.frame(scaled_df)
scal<-gt(head(scaled,10))%>%
tab_header(title=md("**First 10 observation of the dataset scaled**"))
scal
```

First 10 observation of the dataset scaled

Population	Total.Cases	Total.Deaths	Tot.Cases..1M.pop	Tot.Deaths.1M.pop	Death.percentage
0.03870229	-0.2758345	-0.2041157	-0.91345004	-0.75830373	1.64756958
-0.23125886	-0.2626344	-0.2472463	-0.28644200	0.09975988	-0.09706876
0.07299165	-0.2637585	-0.2123413	-0.90326117	-0.79008386	0.65629709
-0.25133052	-0.2947740	-0.2817074	2.61728758	0.73452624	-0.60965558
-0.00300912	-0.2866417	-0.2636770	-0.92401805	-0.87120684	0.27060110
-0.25177754	-0.2999038	-0.2831936	0.27781285	-0.42293969	-0.63775256
-0.25117348	-0.2992451	-0.2818932	-0.42381242	0.21935355	0.20528846
0.07791159	0.9423311	1.0384460	0.41348743	1.41529016	-0.01587664
-0.23053607	-0.2421966	-0.1943522	0.03612033	1.50644897	0.34164034
-0.25111450	-0.2955949	-0.2810985	1.23863370	0.73118096	-0.47167487

Figure 34: first 10 observations

	Population	Total.Cases	Total.Deaths	Tot.Cases..1M.pop	Tot.Deaths.1M.pop	Death.percentage
Population	1.00000000	0.43068686	0.42818884	-0.13013025	-0.06648471	0.08338714
Total.Cases	0.43068686	1.00000000	0.91526266	0.12788974	0.21627212	-0.03251997
Total.Deaths	0.42818884	0.91526266	1.00000000	0.03656118	0.28920072	0.08261611
Tot.Cases..1M.pop	-0.13013025	0.12788974	0.03656118	1.00000000	0.50862967	-0.34998574
Tot.Deaths.1M.pop	-0.06648471	0.21627212	0.28920072	0.50862967	1.00000000	0.06108835
Death.percentage	0.08338714	-0.03251997	0.08261611	-0.34998574	0.06108835	1.00000000

Figure 35: dataset correlation

```
cor(dataset[2:7])
```

Now I can compute the covariance Matrix from which I can extract the information. **EIGEN DECOMPOSITION**

```
dataset.cov <- cov(scaled)
dataset.eigen <- eigen(dataset.cov)
```

As we can see from the eigen value the variability of each component is: **VARIABILITY OF COMPONENTS**

```
PVE <- dataset.eigen$values/sum(dataset.eigen$values)
pct <- round(PVE,3)*100
df.loadings <- data.frame(
PCs= c("PC1","PC2","PC3","PC4","PC5","PC6"), percentage=pct)
gt(df.loadings)%>%
tab_header(title=md("**Relative Variability**"))
```

Relative Variability	
PCs	percentage
PC1	38.5
PC2	27.0
PC3	17.3
PC4	10.2
PC5	5.8
PC6	1.1

Figure 36: relative variability

So, we have obtained the loading vectors and we can compute the component. The percentage of variance explained (PVE) by each principal component is a key output of PCA. It indicates how much of the total variance in the dataset is captured by each principal component. This is crucial for understanding the importance of each principal component in representing the dataset's variability. PC1: 38.5% of the variance

in the data is captured by the first principal component. This indicates that PC1 alone accounts for a significant portion of the total variance in the dataset, making it the most important component. PC2: 27.0% is explained by the second principal component. Together with PC1, they capture over 65% of the dataset's variance, indicating that these two components are significant in representing the dataset's structure. PC3: 17.3%, PC4: 10.2%, PC5: 5.8%, and PC6: 1.1% show progressively smaller portions of the variance explained by each subsequent principal component. The first few principal components (e.g., PC1 and PC2) might be sufficient to capture most of the variability in the data, allowing you to reduce the number of dimensions without losing critical information. This simplification is beneficial for visualization, further analysis, and computational efficiency. The "Relative Variability" table illustrates the importance of each prncipal component in explaining the variance of the dataset.

```
phi <- data.frame(names(dataset[,-1]),
round(dataset.eigen$vectors, 3))
colnames(phi) <- c("Variables", "phi_1", "phi_2", "phi_3",
"phi_4", "phi_5", "phi_6")

gt(phi) %>%
  tab_header(
    title = md("**Loading Vectors**"))
```

Loading Vectors

Variables	phi_1	phi_2	phi_3	phi_4	phi_5	phi_6
Population	-0.378	0.349	0.235	0.819	0.098	0.002
Total.Cases	-0.615	0.050	0.126	-0.318	-0.185	-0.684
Total.Deaths	-0.619	0.096	-0.016	-0.327	0.019	0.707
Tot.Cases..1M.pop	-0.140	-0.683	0.042	0.291	-0.642	0.122
Tot.Deaths.1M.pop	-0.274	-0.495	-0.523	0.163	0.603	-0.125
Death.percentage	-0.023	0.393	-0.808	0.104	-0.424	-0.033

Figure 37: Loading vectors

The table show us lists the loadings (coefficients) of six principal components (PC1 to PC6) for six variables related to a dataset, presumably concerning some aspects of population and COVID-19 statistics. These loadings are derived from a Principal Component Analysis (PCA), a statistical technique used to reduce the dimensionality of a dataset while retaining as much of the variation present in the dataset as possible.

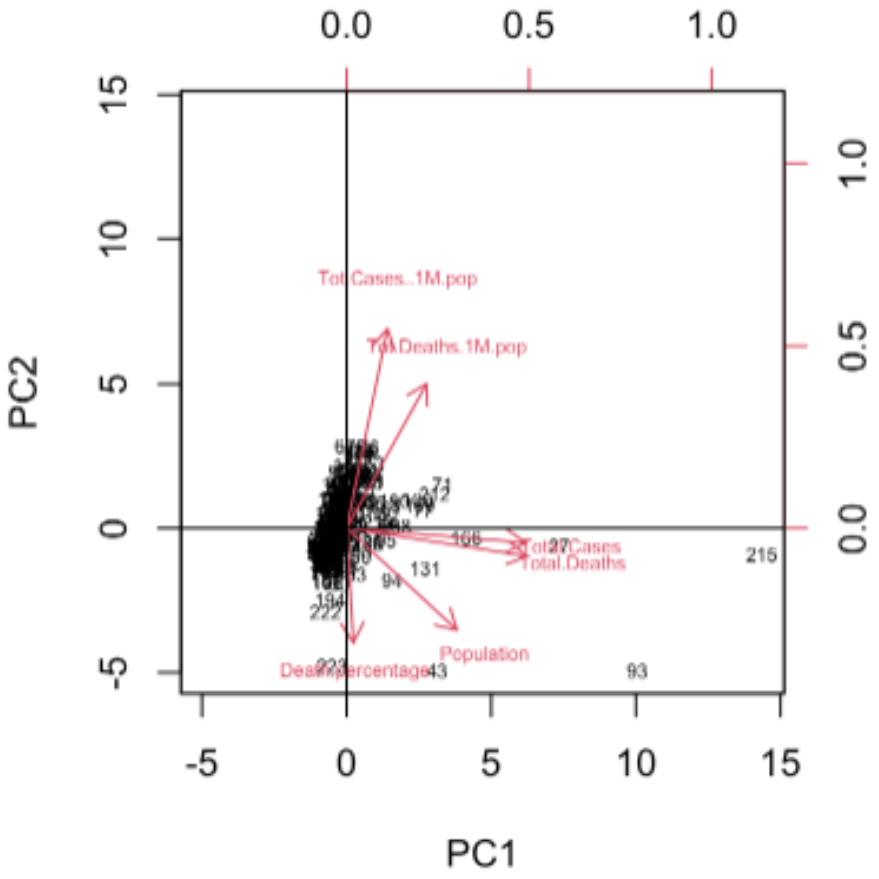


Figure 38: biplot

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

4.1 Choosing the optimal number of PCs

At this point we have to choose the optimal number of principal components and to do this we can use 3 different heuristic methods:

- **CPVE**
- **Kaiser's rule**
- **Scree Plot**

4.1.1 Cumulative proportion of variance explained

CPVE allows us to choice the number of Components that represent a variability greater than 80%. Now we can see the graph.

```
    cumPVE <-qplot(c(1:6),cumsum(PVE))+
  geom_line()+
  xlab("Principal Component")+
  ylab(NULL)+
  ggtitle("Cumulative Scree Plot")+
  ylim(0,1)
print(cumPVE)
```

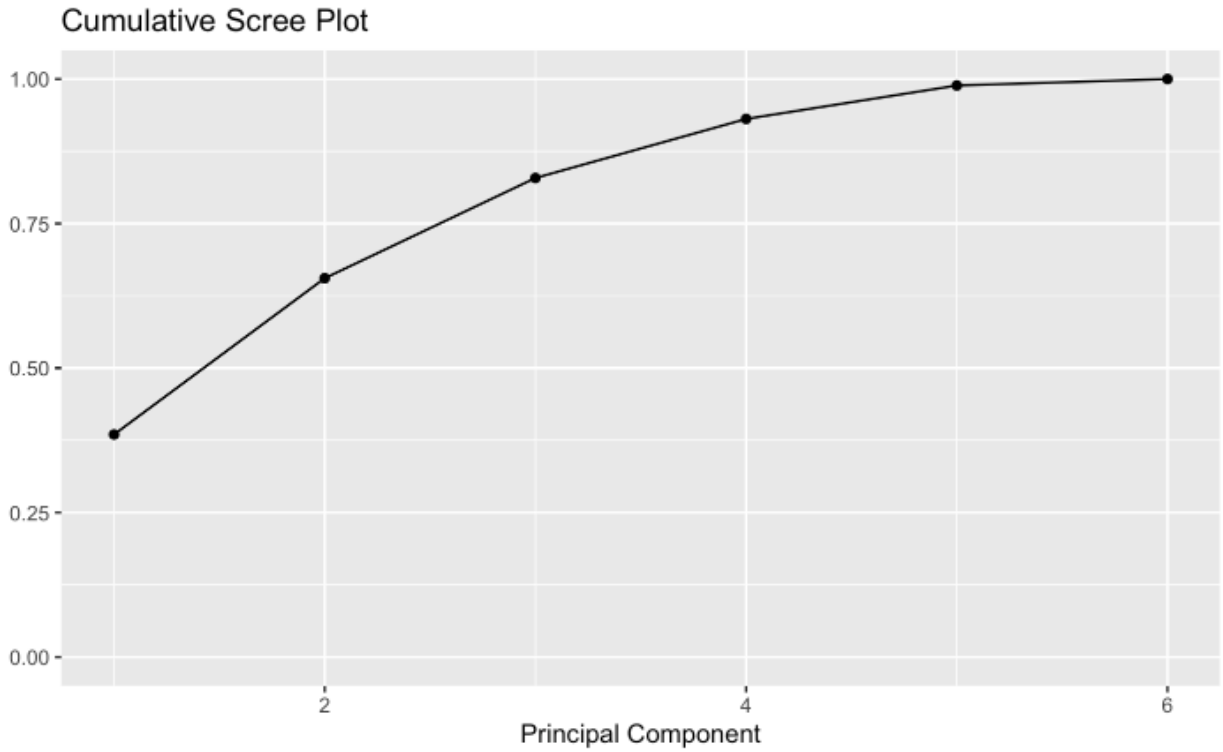


Figure 39: Cumulative proportion of variance explained

As we can see from the chart i should choose 3 principal components because the 80% of the variability is represented by 3 components.

4.1.2 Kaiser's Rule

According to Kaiser 's rule instead we should take the eigen values that are greater than 1 related to PCs.

```
    df.eigen_Value <- data.frame(
  PCs=c("PC1","PC2","PC3","PC4","PC5","PC6"),
  "Eigen Value"=round(dataset.eigen$values,2))
gt(df.eigen_Value)%>%
  tab_header(
    title = md("**Eigen Value**"))
```

As we can see i should choose 3 Principal Components because the eigen values that are grater than 1 are the first three.

Eigen Value	
PCs	Eigen.Value
PC1	2.31
PC2	1.62
PC3	1.04
PC4	0.61
PC5	0.35
PC6	0.07

Figure 40: Eigen values

4.1.3 Scree Plot

In this method the choice of the PCs is graphical,in fact we should see the elbow to determine the number of PCs

```
    PVEplot <-qplot(c(1:6),PVE)+
  geom_line()+
  xlab("Principal Component")+
  ylab(NULL)+
  ggtitle(" Scree Plot")+
  ylim(0,1)
print(PVEplot)
```

As we can see from the scree plot the elbow says us that the number of principal component that we have to choice is three. Putting into a table the result.

```
    res <- data.frame(Criteria=c("PVCE","Kraiser's rule","Scree plot"),Result=c(3,3,3))
gt(res)%>%
tab_header(
  title = md("**Results**"))
```

From the above results we can say that the number of Principal Components is 3 and after this we can compute the principal component scores.

```
    eigen_vec <- dataset.eigen$vectors
rownames(eigen_vec)<- colnames(scaled_df)
colnames(eigen_vec)<- c("PC1","PC2","PC3","PC4","PC5","PC6")
eg<-data.frame(eigen_vec)
PC1 <- scaled_df %*% eigen_vec[,1]
```

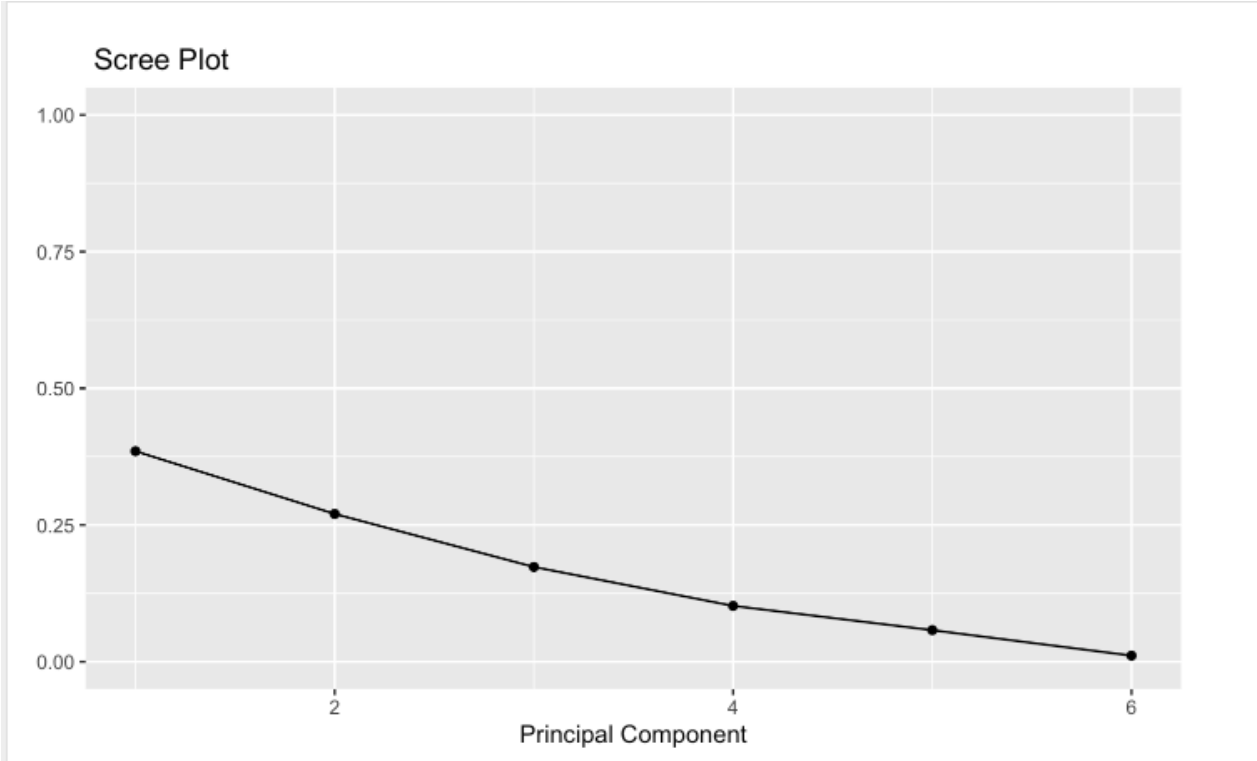


Figure 41: Scree plot

Results	
Criteria	Result
PVCE	3
Kraiser's rule	3
Scree plot	3

Figure 42: Results

```
PC2 <- scaled_df %*% eigen_vec[,2]
PC3 <- scaled_df %*% eigen_vec[,3]
PC_m <- c(PC1,PC2,PC3)
# Create the data frame with Principal Component scores
PC <- data.frame(PC1, PC2,PC3)
head(PC)%>%
gt()%>%
  tab_header(
    title = md("**Principal Component Scores**"))
```

To give a sense to the PC Plot we should observe the loadings and we should understand that the PCs represent.

Principal Component Scores		
PC1	PC2	PC3
0.5791847	1.626651657	-0.99542885
0.4170435	-0.009601072	-0.06923418
0.5941862	1.257930528	-0.16754494
-0.1032803	-2.520792577	0.12628296
0.7028987	1.128176707	0.16582844
0.5469295	-0.361014042	0.65593352

Figure 43: Principal component Scores

4.2 PCA

Therefore we can plot the chart of PCA.

Loadings Vector			
Variables	phi_1	phi_2	phi_3
Population	-0.378	0.349	0.235
Total.Cases	-0.615	0.050	0.126
Total.Deaths	-0.619	0.096	-0.016
Tot.Cases..1M.pop	-0.140	-0.683	0.042
Tot.Deaths.1M.pop	-0.274	-0.495	-0.523
Death.percentage	-0.023	0.393	-0.808

Figure 44: loading vectors

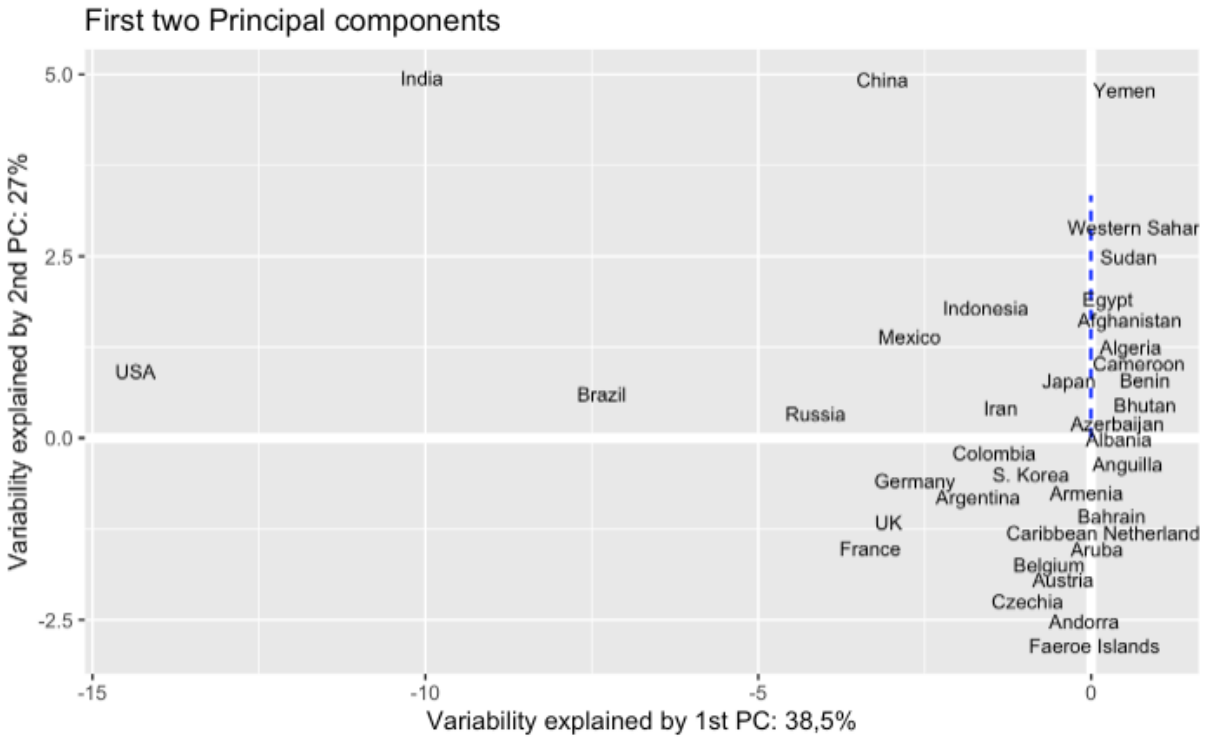


Figure 45: First two PCs

Horizontal Axis (X-axis): It represents the scores of each observation on the first principal component (PC1), which explains 38.5% of the variance in the dataset. The variability on this axis is the greatest, which is why it's plotted on the x-axis as the first principal component typically captures the most variance. **Vertical Axis (Y-axis)**: This axis represents the scores on the second principal component (PC2), explaining 27% of the variance. While less variability is explained compared to PC1, PC2 captures additional information that is orthogonal (uncorrelated) to what is captured by PC1. **Data Points**: Each point represents an observation (or a data point) in the original dataset projected into this two-dimensional principal component space. The proximity of points to each other suggests how similar they are with respect to the variables that significantly contribute to these components. Dense Region (Vertical Line): There seems to be a high density of points around the value 0 on PC1, as indicated by a vertical blue dashed line. This might suggest a cutoff or a natural division in the dataset where a significant change in the data structure occurs. This pattern could imply that the data have an inherent grouping or clustering along the PC1 dimension. This kind of plot is particularly useful for identifying patterns, clusters, or outliers in the data.

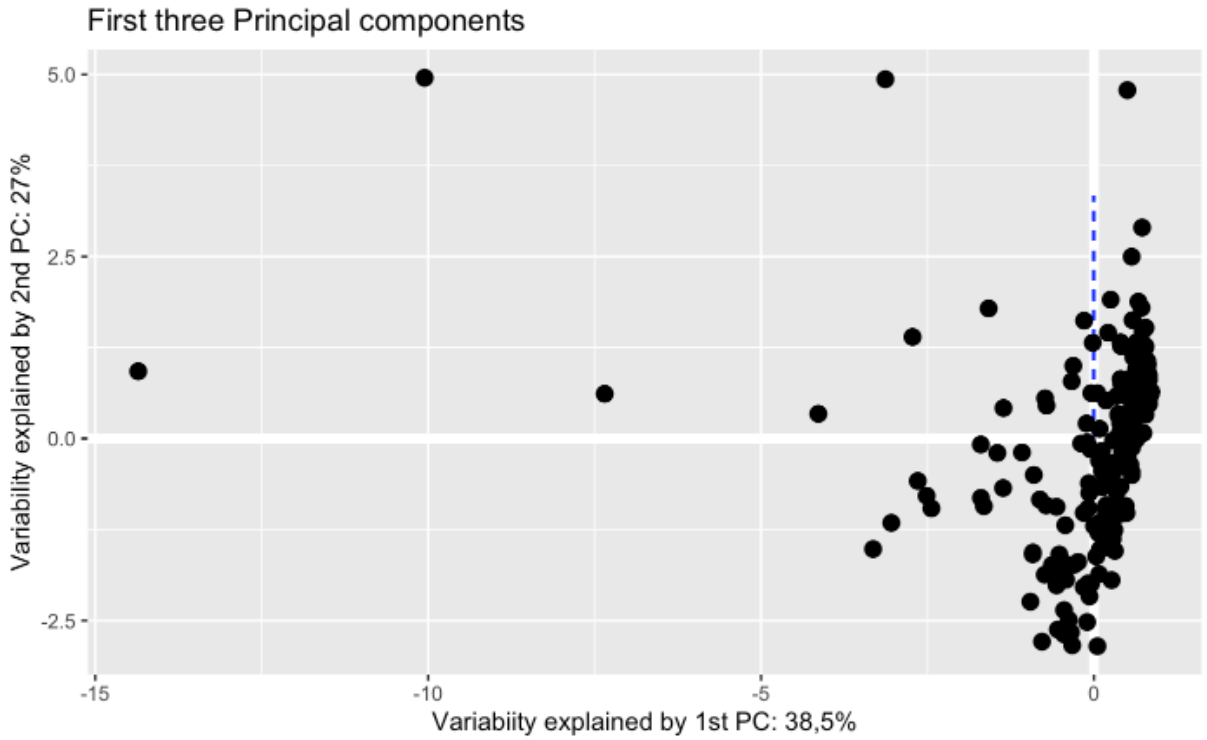


Figure 46: First two PCs

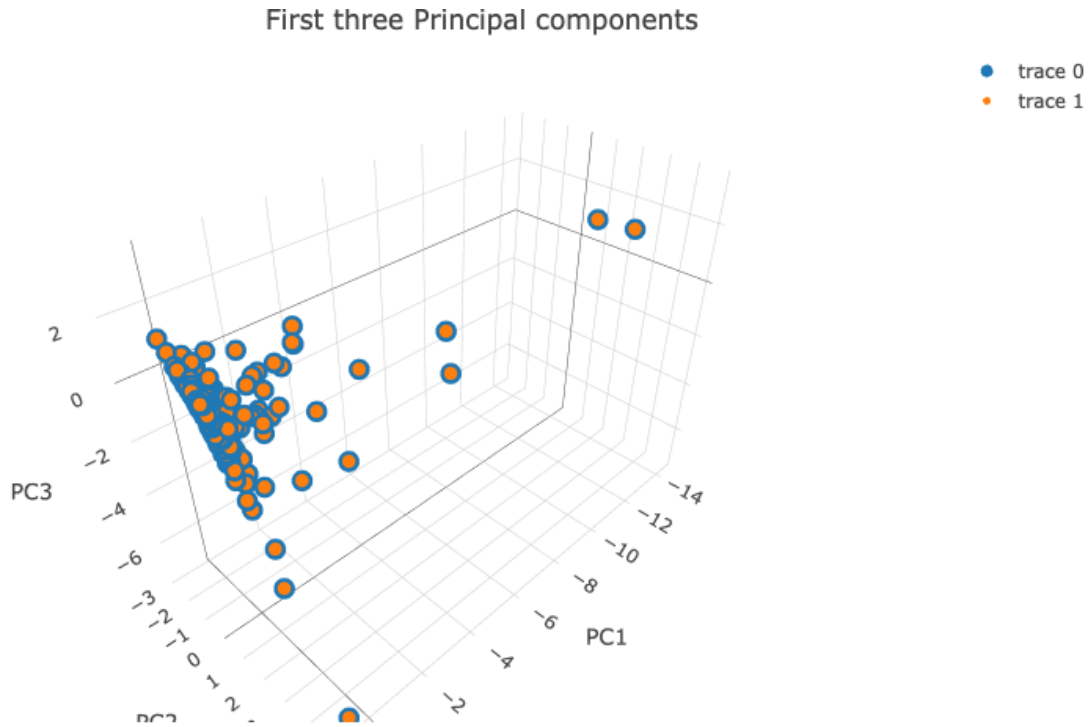


Figure 47: Three Dimensional PCs

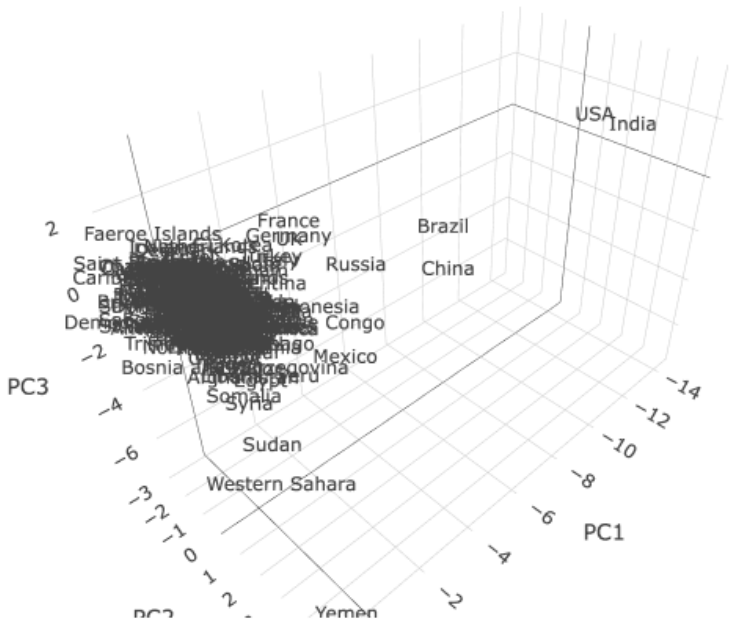


Figure 48: Three Dimensional PCs

In this plot (Figure 47), each dot represents an observation in the dataset projected onto the space defined by the first three principal components. The colors (blue and orange) indicate different clusters or categories within the data, suggesting that the PCA may also be revealing some natural grouping within the dataset. The fact that the orange dots are spread out along the PC1 axis but not much along PC2 and PC3 suggests that the variance within this group is largely explained by the first principal component, such as I said before in the first two PCs. The blue dots, more clustered in a different region of the space, indicate a different variance-covariance structure for this group of observations. However, the true meaning and the implications of these clusters As we can see the first PC gives more weight to Total.Cases, Total.Deaths. The second PC gives more weight to Tot.Cases..1M.pop. The third PC gives more weight to Tot.Deaths.1M.pop and Death.Percentage while the variable Population has the same weight in PC1 and PC2. In summary, these data suggest that variations in total cases and total deaths are strongly associated with the first principal component (PC1), while variations in cases and deaths per million population are more strongly associated with the second principal component (PC2). The death percentage, on the other hand, appears to be primarily influenced by the second and third principal components (PC2 and PC3).

```
    library(plotly)

plot_ly(data = PC, x = ~PC1, y = ~PC2, z = ~PC3, type =
"scatter3d", mode = "markers") %>%
  add_markers(marker = list(size = 5)) %>%
  layout(scene = list(xaxis = list(title = "PC1"),
                      yaxis = list(title = "PC2"),
                      zaxis = list(title = "PC3")),
         title = "First three Principal components")
```

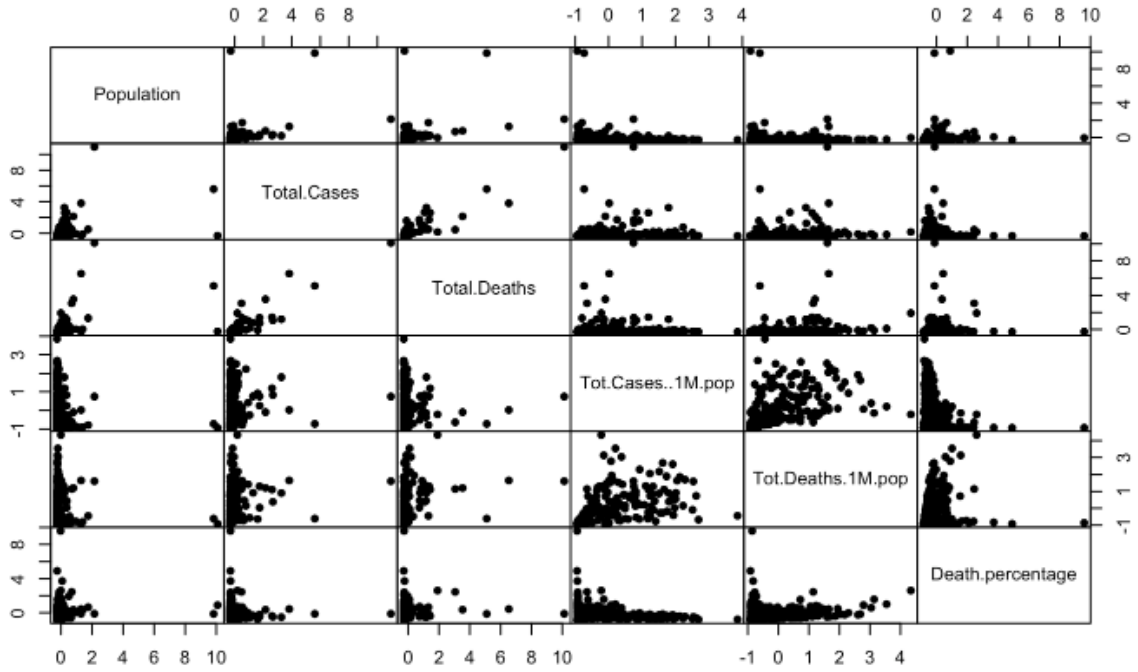


Figure 49: scatterplot

Now I will visualize the data.

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

5 Cluster Analysis

5.1 Assesing Cluster Tendency

Cluster Analysis (CA), simply said clustering, is one of the most important statistical methods for discovering knowledge in multidimensional data. The goal of CA is to identify patterns (or groups, or clusters) of similar units within a data set. There are different methods. The first step, in our case, is to calculate the distance between pair of units and build the distance matrix. Important! We are analysing only a subset of data. First distance to be computed is Euclidean.

```
    df<-scaled_df
dist.eucl <- dist(df, method="euclidean")
round(as.matrix(dist.eucl)[1:10, 1:10],2)
      1    2    3    4    5    6    7    8    9   10
1  0.00 2.06 0.99 4.46 1.38 2.62 1.83 3.50 2.79 3.38
2  2.06 0.00 1.35 3.02 1.24 0.94 0.36 2.33 1.51 1.69
3  0.99 1.35 0.00 4.05 0.41 1.82 1.25 3.17 2.52 2.88
4  4.46 3.02 4.05 0.00 3.99 2.61 3.19 3.01 2.86 1.39
5  1.38 1.24 0.41 3.99 0.00 1.59 1.23 3.21 2.58 2.80
6  2.62 0.94 1.82 2.61 1.59 0.00 1.27 2.68 2.18 1.51
7  1.83 0.36 1.25 3.19 1.23 1.27 0.00 2.36 1.38 1.87
8  3.50 2.33 3.17 3.01 3.21 2.68 2.36 0.00 1.82 2.18
9  2.79 1.51 2.52 2.86 2.58 2.18 1.38 1.82 0.00 1.65
10 3.38 1.69 2.88 1.39 2.80 1.51 1.87 2.18 1.65 0.00
```

The matrix represents a distance matrix calculated using the Euclidean distance between pairs of observations in a dataset. It shows the distances for the first 10 observations. This matrix shows the pairwise Euclidean distances between observations. A distance of 0 (as seen on the diagonal) means it's the same point, as distance is calculated between an observation and itself. Observations closer to each other have smaller distances. For example, observations 1 and 3 are relatively close, with a distance of 0.99, indicating they might be similar or belong to the same cluster. Larger distances between observations might indicate outliers. For instance, observation 4 has relatively larger distances to others, with the smallest being 1.39 to observation 10, suggesting it could be an outlier or belong to a different cluster. By examining the distance matrix, we can assess the clustering tendency of the dataset. Observations with smaller distances to each other compared to other pairs might suggest a natural grouping or cluster. However, determining the exact number of clusters or the best way to cluster would require further analysis.

```
dist.man <- dist(df, method="manhattan")
round(as.matrix(dist.man)[1:10, 1:10],2)
      1    2    3    4    5    6    7    8    9   10
1  0.00 3.56 1.09 7.67 1.61 4.21 3.30 7.66 4.83 6.15
2  3.56 0.00 2.60 4.14 2.24 1.72 0.65 4.90 2.24 2.62
3  1.09 2.60 0.00 6.74 0.64 3.27 2.37 6.66 3.89 5.22
4  7.67 4.14 6.74 0.00 6.30 3.53 4.38 6.36 4.47 1.52
5  1.61 2.24 0.64 6.30 0.00 2.84 1.94 6.52 3.75 4.78
6  4.21 1.72 3.27 3.53 2.84 0.00 2.19 5.49 3.32 2.29
7  3.30 0.65 2.37 4.38 1.94 2.19 0.00 5.15 2.05 2.86
8  7.66 4.90 6.66 6.36 6.52 5.49 5.15 0.00 3.55 4.85
9  4.83 2.24 3.89 4.47 3.75 3.32 2.05 3.55 0.00 2.95
10 6.15 2.62 5.22 1.52 4.78 2.29 2.86 4.85 2.95 0.00
```

This Matrix represents distances calculated using the Manhattan (or city block) distance metric between pairs of observations in your dataset, for the first 10 observations. The Manhattan distance measures the distance between two points by summing the absolute values of the differences in their coordinates. Smaller distances indicate closer or more similar observations. For example, observations 1 and 3 have a small distance (1.09), suggesting they're quite similar based on the Manhattan distance metric. This could indicate they belong to the same cluster or have similar properties. Larger distances can hint at outliers or observations that significantly differ from others. Observation 4, for instance, shows relatively larger distances to most other observations (with the smallest distance being 1.52 to observation 10), which could indicate it's distinct or an outlier within this subset of the data. The Manhattan distance can yield different clustering results compared to the Euclidean distance, especially in higher-dimensional spaces or when the data has a grid-like structure. Analyzing this matrix can help identify natural groupings or the clustering tendency, but deciding on the number of clusters or the best method for clustering would require further analysis using specific clustering algorithms and validation methods.

```{r}
fviz_dist(dist.eucl, show_labels = FALSE)
```

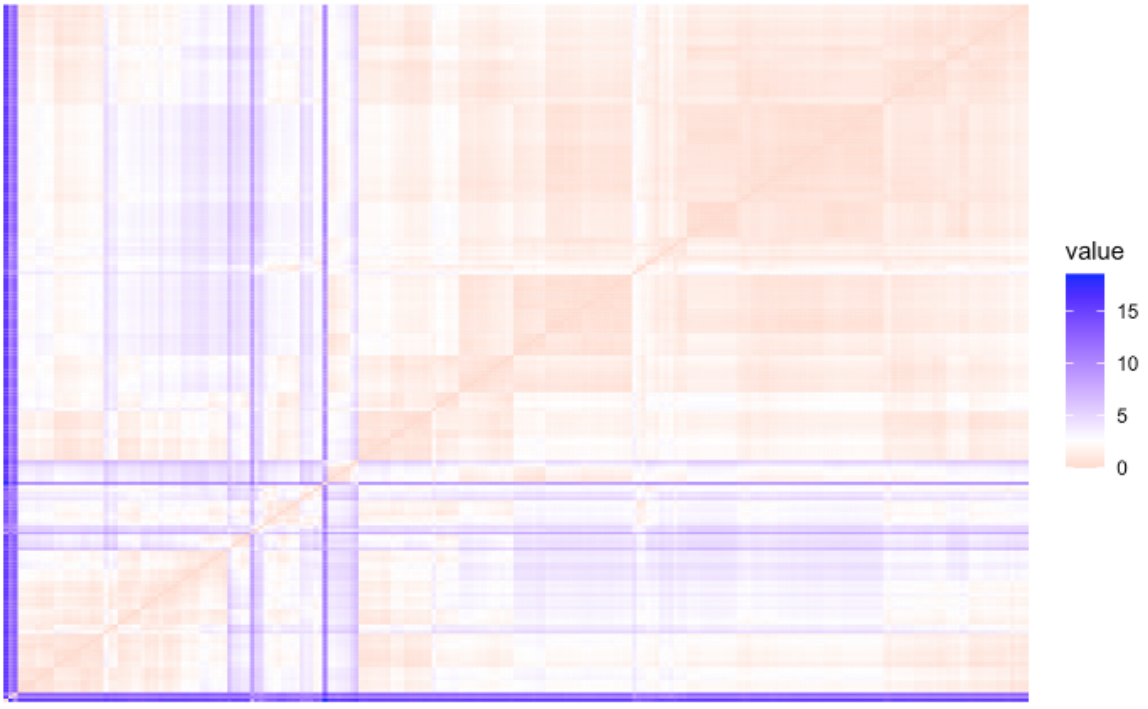


Figure 50: Euclidean Distance

According with the images, a level of colour red indicates a higher similarity between the observations, and a level of colour blue indicates a lower similarity between observations. Another approach that gives us important information about clustering is the **Hopkins method**, that is a measure of clustering tendency that include the values in the interval [0:1].

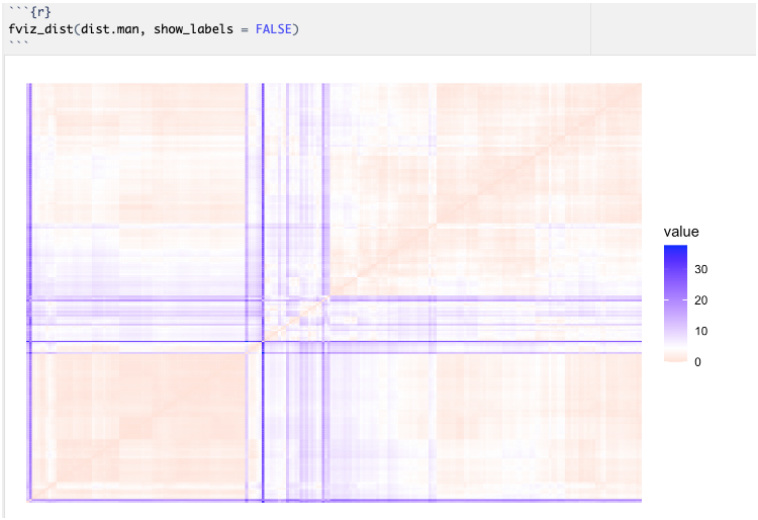


Figure 51: Manhattan Distance

```
random_df <- apply(df,2, function(x){runif(length(x),min(x),(max(x)))})
random_df <- as.data.frame(random_df)
scaled.random_df=scale(random_df)
pairs(scaled.random_df, gap=0, pch=16)
```

```
    random.data <- matrix(rnorm(nrow(df) * ncol(df)), nrow = nrow(df))
```

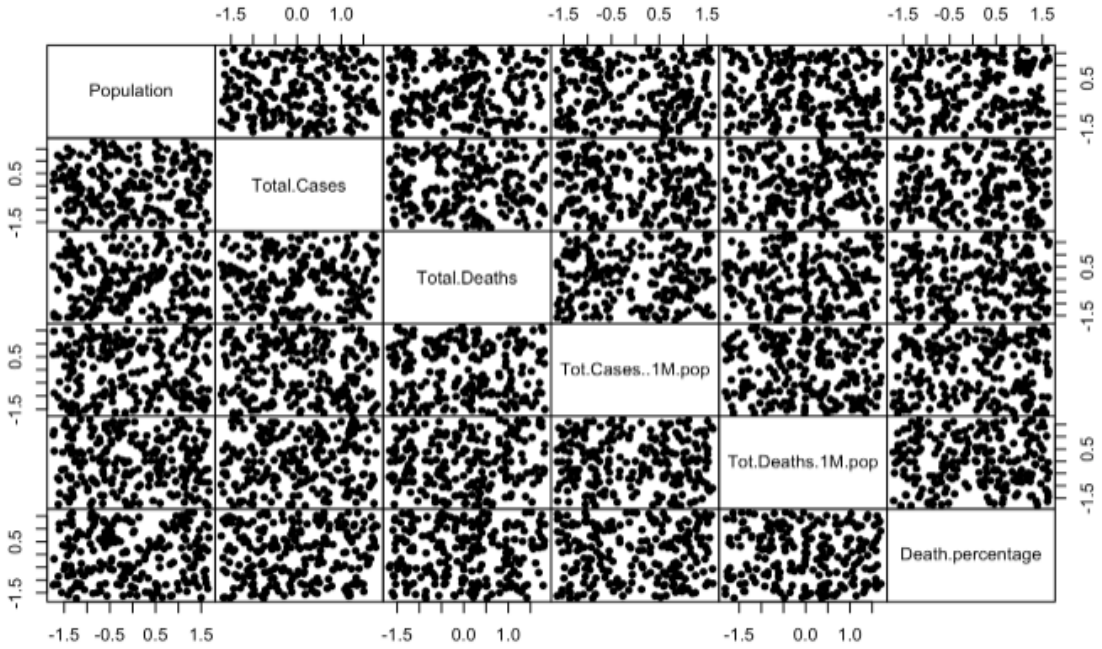


Figure 52: Scatterplot of random data

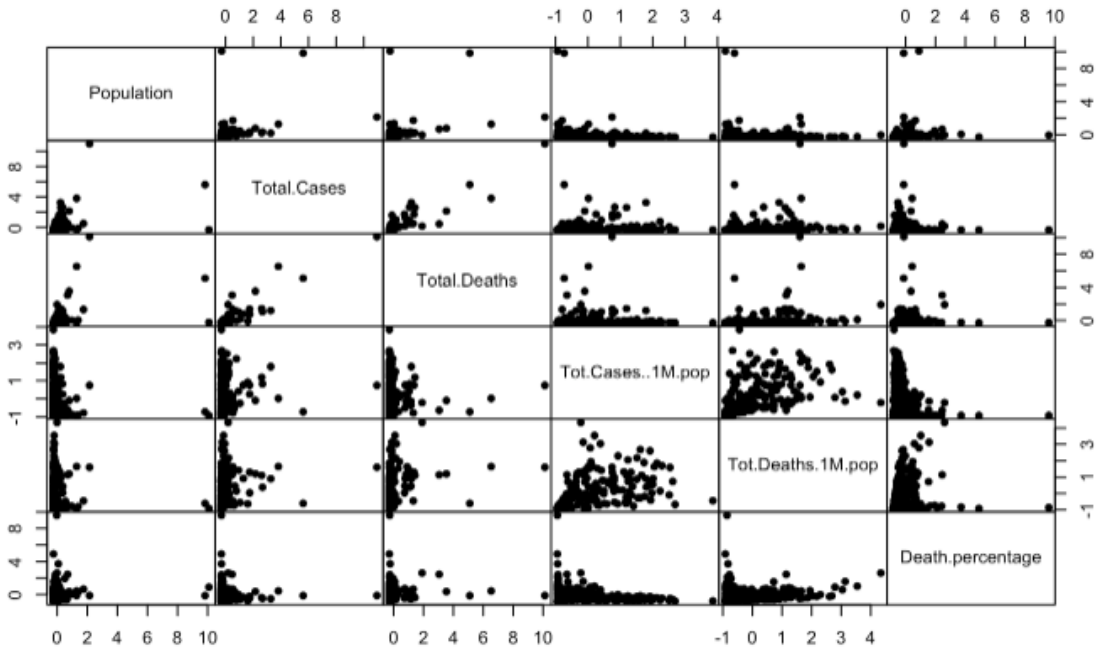


Figure 53: Scatterplot of dataset

```
pca_covid <- prcomp(df, scale. = TRUE)
```

```
pca_covid <- prcomp(df, scale. = TRUE)
pca_random <- prcomp(random.data, scale. = TRUE)

pca_data_plot <- fviz_pca_ind(pca_covid, title = "PCA - COVID-19 Data",
                              palette = "jco", geom = "point",
                              ggtheme = theme_classic(), legend = "bottom")

pca_random_plot <- fviz_pca_ind(pca_random, title = "PCA - Random Data",
                                palette = "jco", geom = "point",
                                ggtheme = theme_classic(), legend = "bottom")


ggarrange(pca_data_plot, pca_random_plot, ncol = 1, nrow = 2)
```

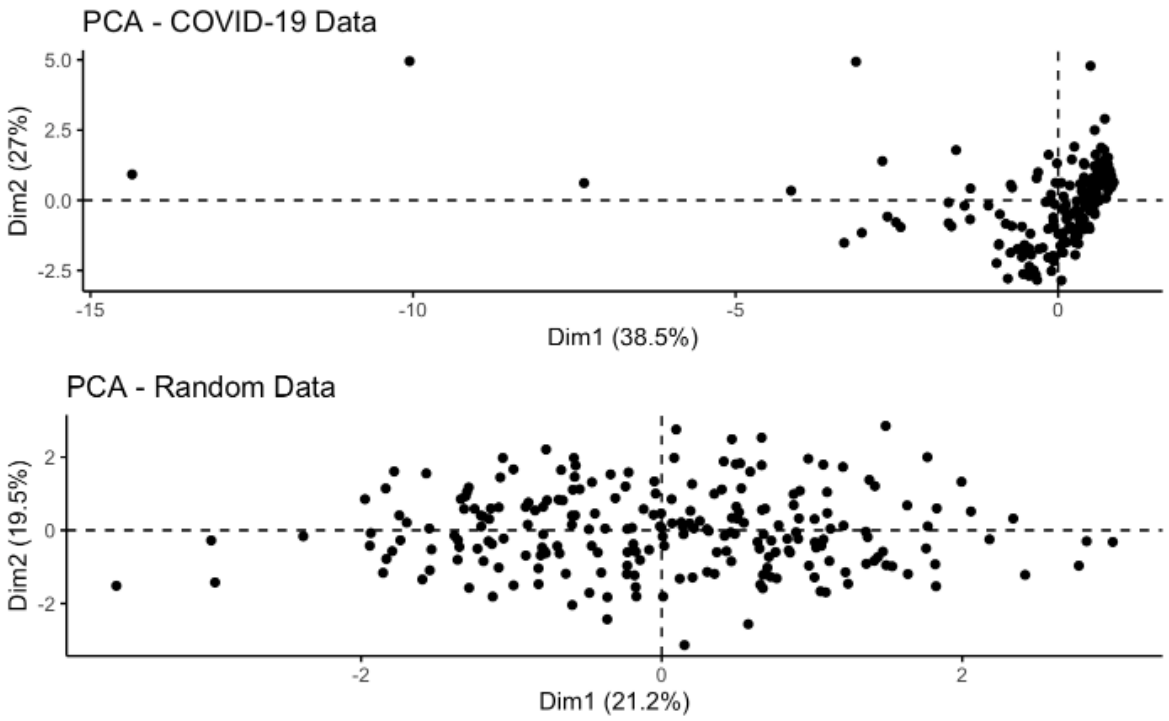


Figure 54: PCA

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

- H close to 0 indicates clustered data;
- indicates uniformly distributed data (no meaningful clusters).

```
covid_hop <-hopkins(df, n=nrow(df)-1)
covid_hop
[1] 0.05921581
random_hop <- hopkins(random_df, n = nrow(random_df)-1)
random_hop
[1] 0.506032
```

From the provided results:

- **random_hop** has a Hopkins value of around 0.514, which is close to 0.5, suggesting that the random_df dataset doesn't have a natural clustering tendency and appears to consist of random data without distinct groups.
- **covid_hop** has a Hopkins value of around 0.06, which is very close to 0, indicating that the dataset exhibits a strong clustering tendency, meaning it likely contains significant clusters.

```
p1<-fviz_dist(dist(df), show_labels=FALSE)+
labs(title="Covid Data")
```

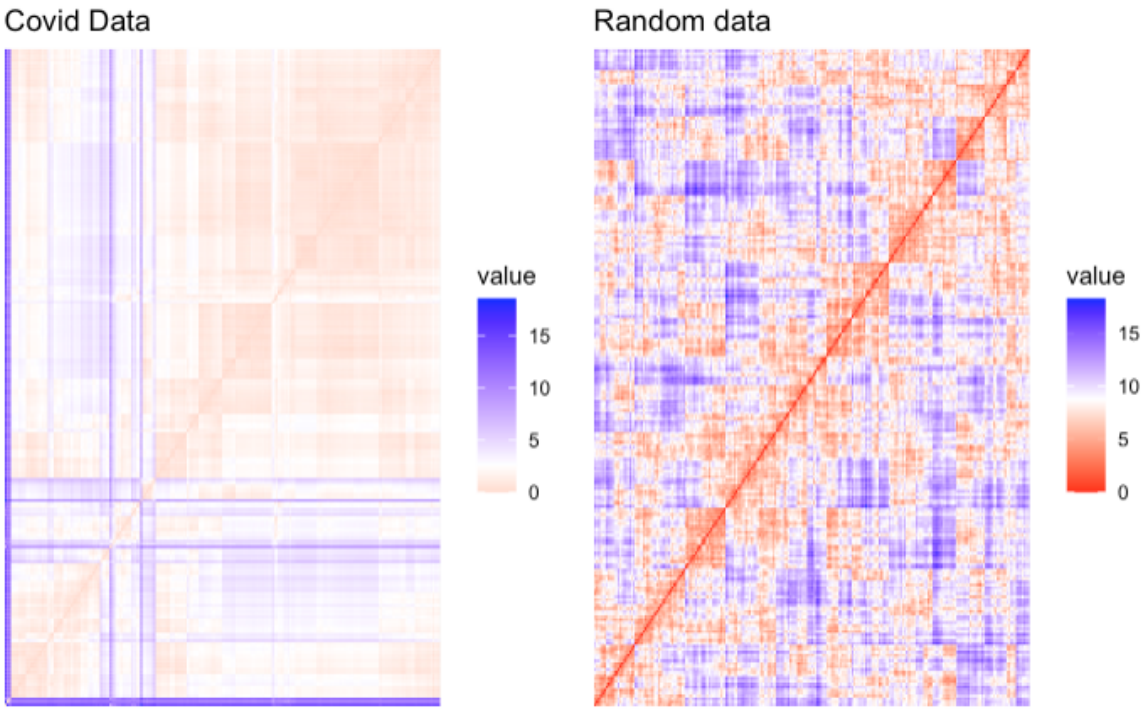


Figure 55: VAT

```
p2<-fviz_dist(dist(random_df), show_labels =FALSE)+
  labs(title="Random data")

ggpubr::ggarrange(p1,p2)
```

Now i want to try also the VAT method that is a graphical method to assess the presence of cluster (figure 55).

5.2 CLUSTER ALGORITHMS

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

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

5.2.1 Cluster Algorithm based Ward's linkage method and Euclidean distance

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

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

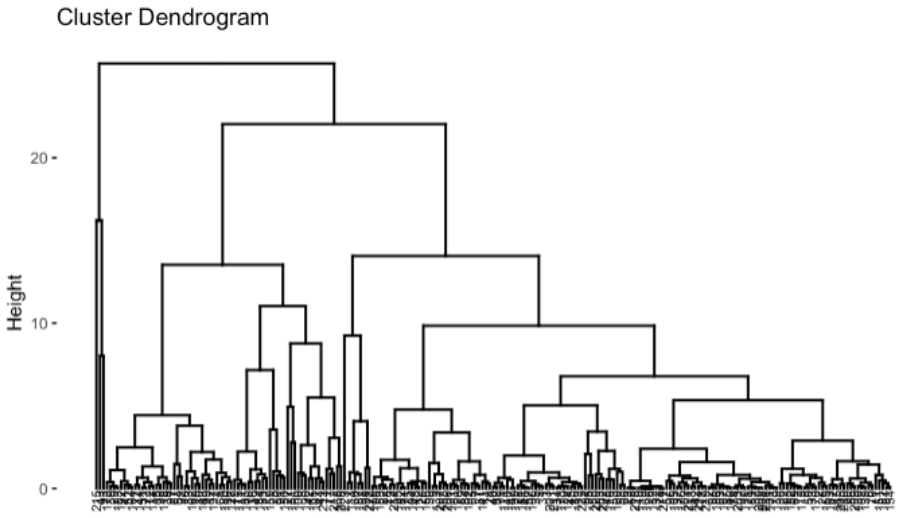
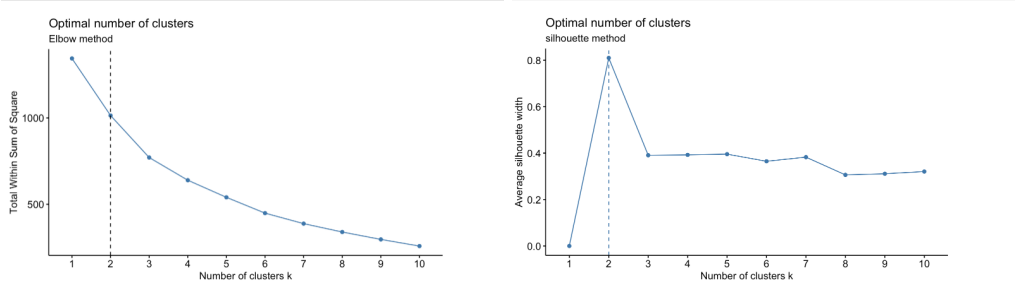


Figure 56: Dendrogram

```
> res.coph <- cophenetic(res.hc)
> cor(dist.eucl, res.coph)
[1]  0.5485074
```

The result is 0.54, so isn't good, because the value above 0.75 are felt to be good. This means that the clustering approach that we used does not preserves the original distances between units. Next, the computation of the optimal number of cluster K.

```
fviz_nbclust(df, hcut, method = "wss", distance="euclidean") +
labs(subtitle = "Elbow method") +
geom_vline(xintercept = 2, linetype = 2)
```



```
fviz_nbclust(df, hcut, method = "silhouette", distance
="euclidean")+ labs(subtitle = "silhouette method")
fviz_nbclust(df, hcut, method = "gap_stat", distance ="euclidean",
nboot = 500)+ labs(subtitle = "Gap statistic method")
```

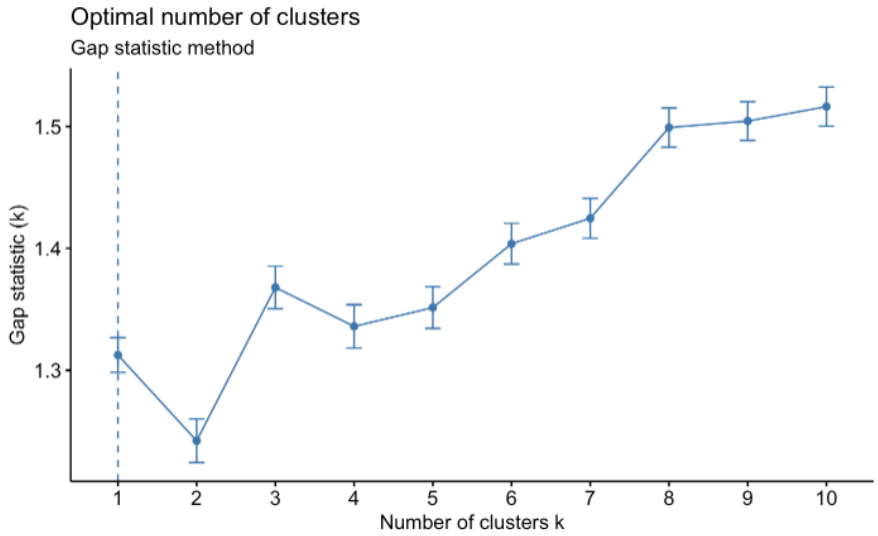


Figure 58: Gap Statistic Method

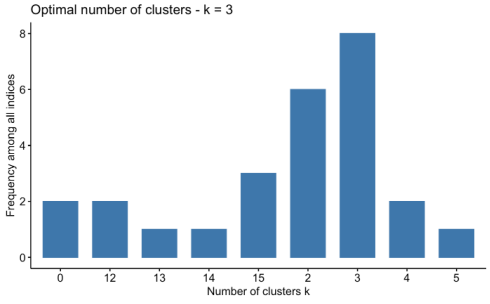
```
y <- NbClust(df, diss=NULL, distance = "euclidean", method = "ward.D2")
fviz_nbclust(y)
```

```
group <- cutree(res.hc, k=3)
fviz_dend(res.hc, k=3, cex=0.5, k_colors = c("red","blue","black"), color_labels_by_k = TRUE, rect = TRU
```

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

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

```
*** : 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
* 8 proposed 3 as the best number of clusters
* 2 proposed 4 as the best number of clusters
* 1 proposed 5 as the best number of clusters
* 2 proposed 12 as the best number of clusters
* 1 proposed 13 as the best number of clusters
* 1 proposed 14 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  3

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



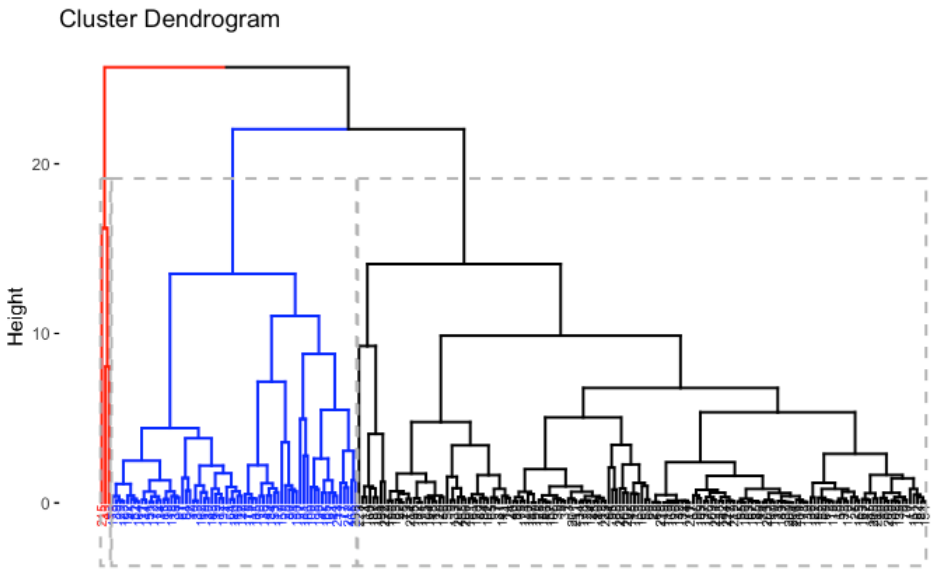


Figure 60: Cluster Dendrogram

Blue cluster and red cluster are not well separeted.

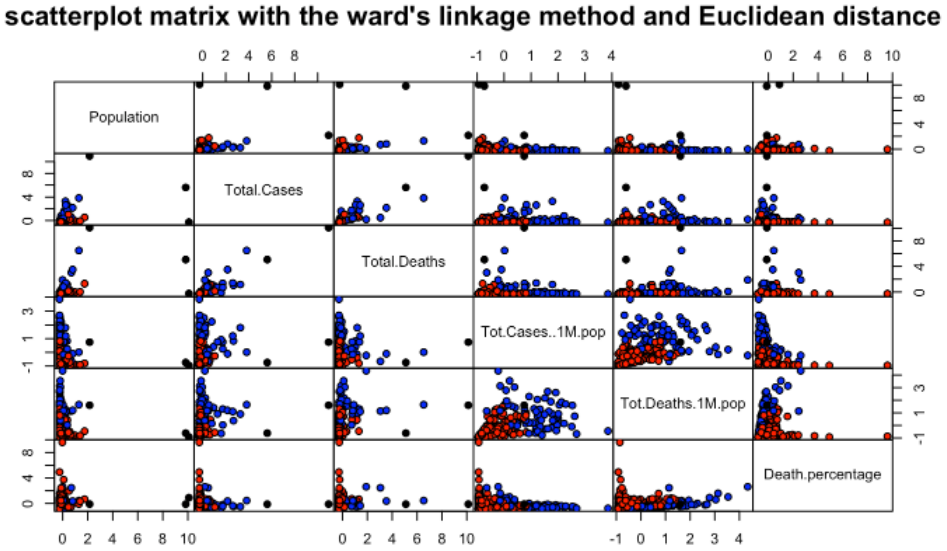


Figure 61: ScatterPlot

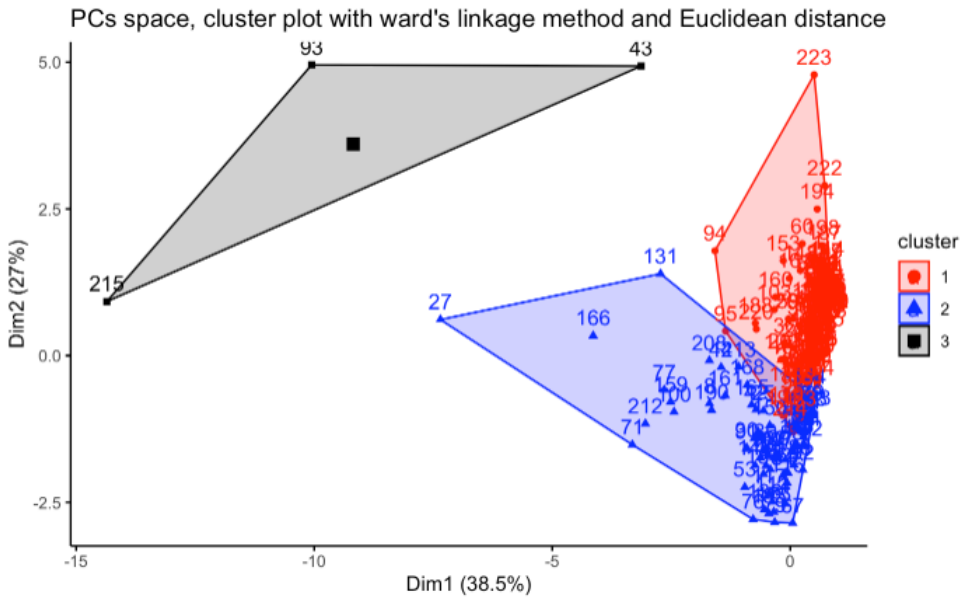


Figure 62: Cluster

5.2.2 Cluster Algorithm based on single linkage method and euclidean distance

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

```
> res.coph <- cophenetic(hc.single)
> cor(dist.eucl, res.coph)
[1] 0.9006669
```

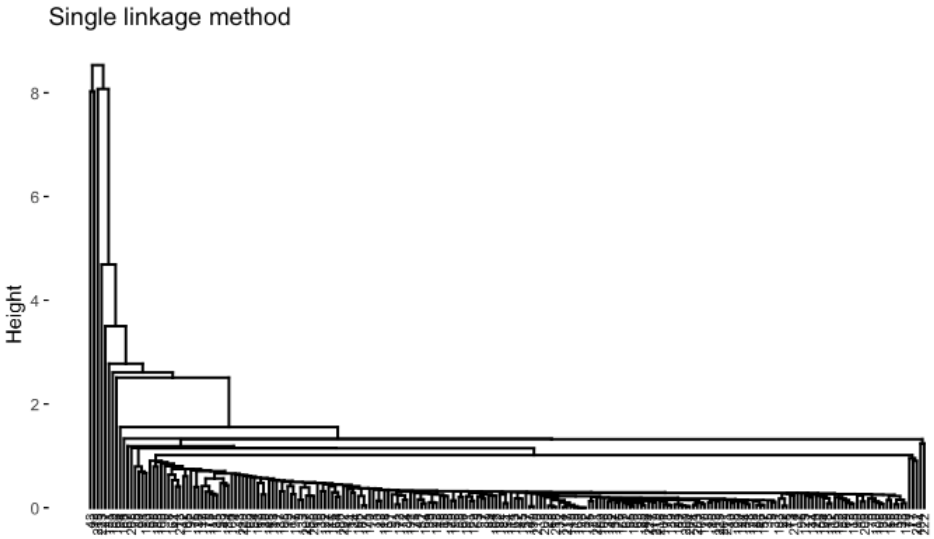
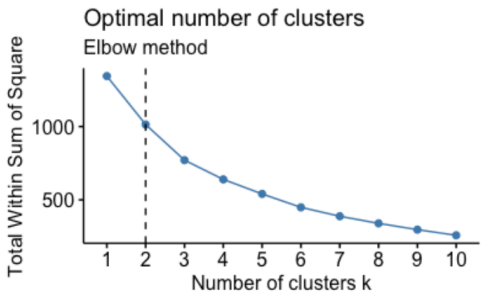
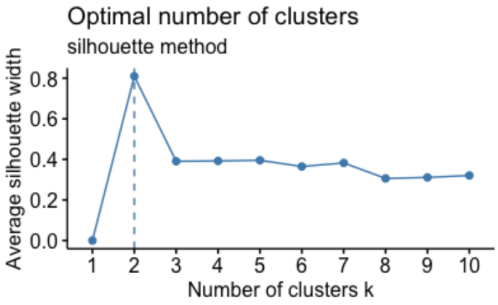


Figure 63: Dendrogram

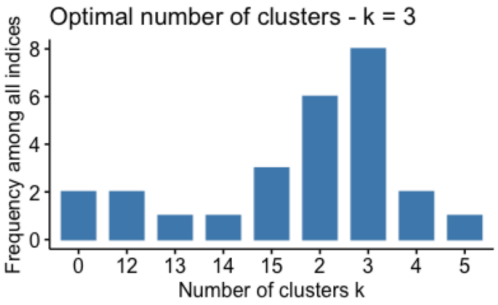
the result is very good because the value should be above 0.75.

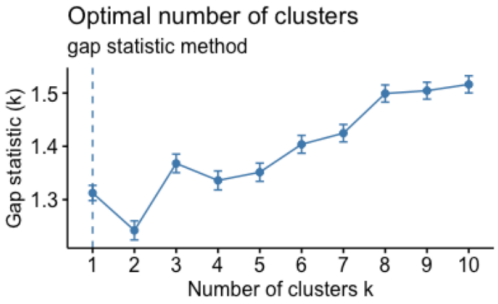
```
fviz_nbclust(df, hcut, method =
"wss", distance ="euclidean")+
labs(subtitle = "Elbow method")+
geom_vline(xintercept = 2,
linetype=2)
fviz_nbclust(df,hcut, method =
"silhouette", distance =
"euclidean")+ labs(subtitle =
"silhouette method")
```






```
* 1 proposed 14 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  3


*******************************************************************
Among all indices:
==================
* 2 proposed  0 as the best number of clusters
* 6 proposed  2 as the best number of clusters
* 8 proposed  3 as the best number of clusters
* 2 proposed  4 as the best number of clusters
* 1 proposed  5 as the best number of clusters
* 2 proposed  12 as the best number of clusters
* 1 proposed  13 as the best number of clusters
* 1 proposed  14 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  3 .
```






```
group1 <- cutree(hc.single, k= 2)
fviz_dend(hc.single, k=2, cex=0.5,
k_colors = c("red", "blue"), color_labels_by_k = TRUE,
rect = TRUE)
```

```
pairs(df, gap=0, main = "scatter plot matrix with the ward's
linkage method and Euclidean distance", pch=21,
bg=c("red","blue")[group1])
```

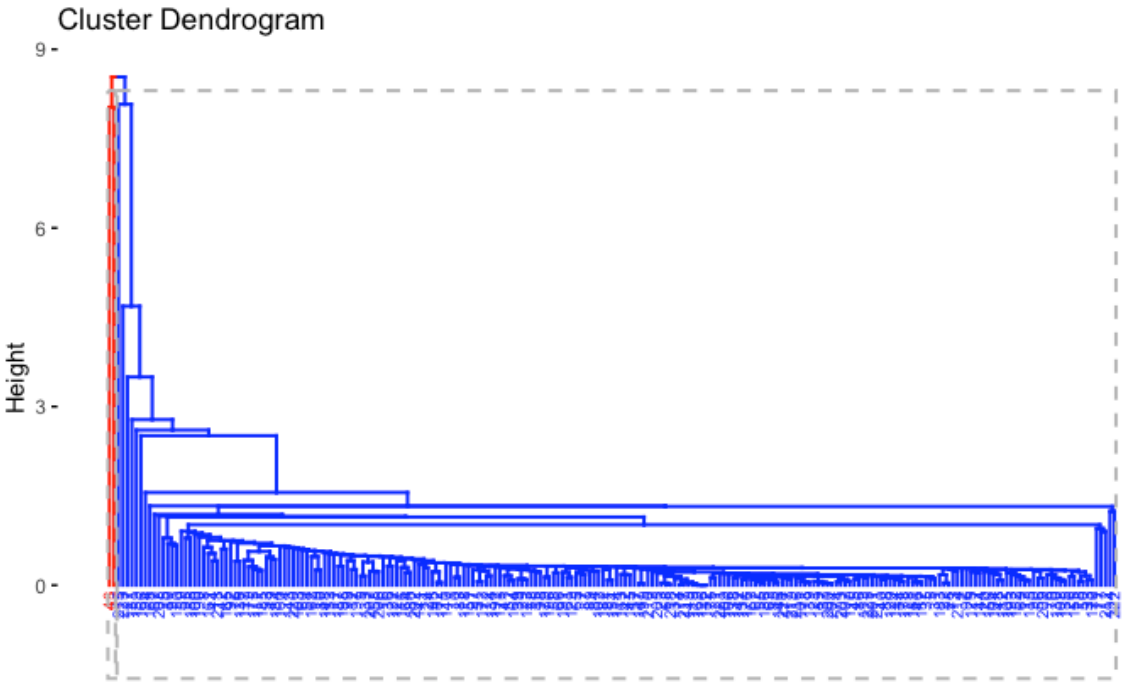


Figure 65: Dendrogram with group

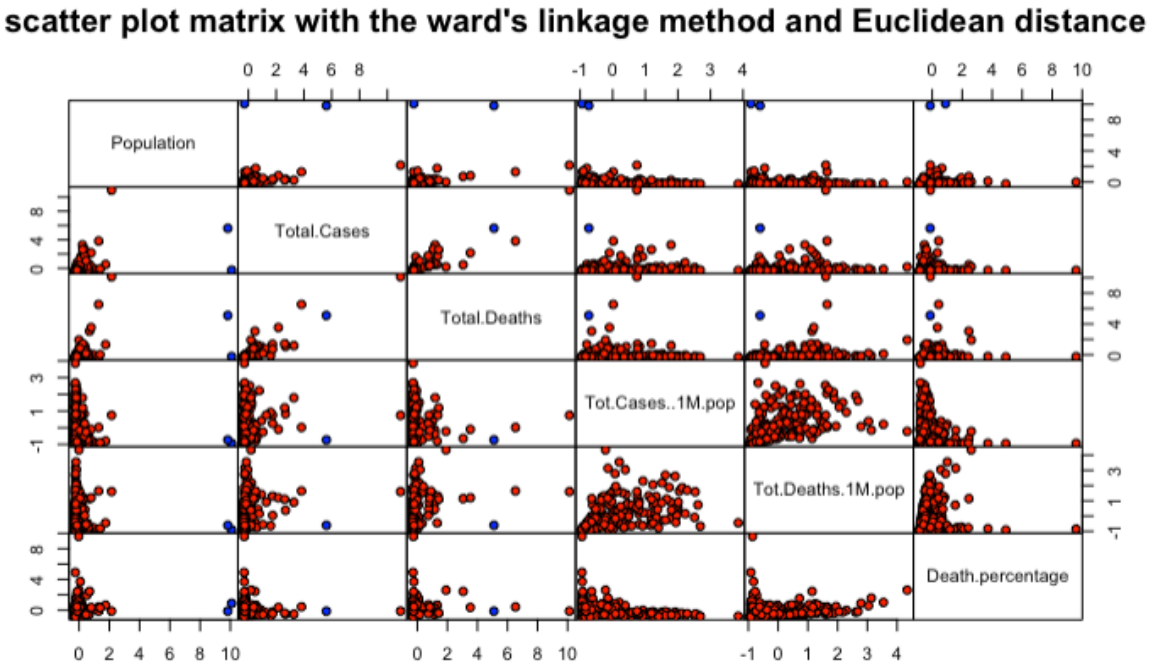


Figure 66: ScatterPlot

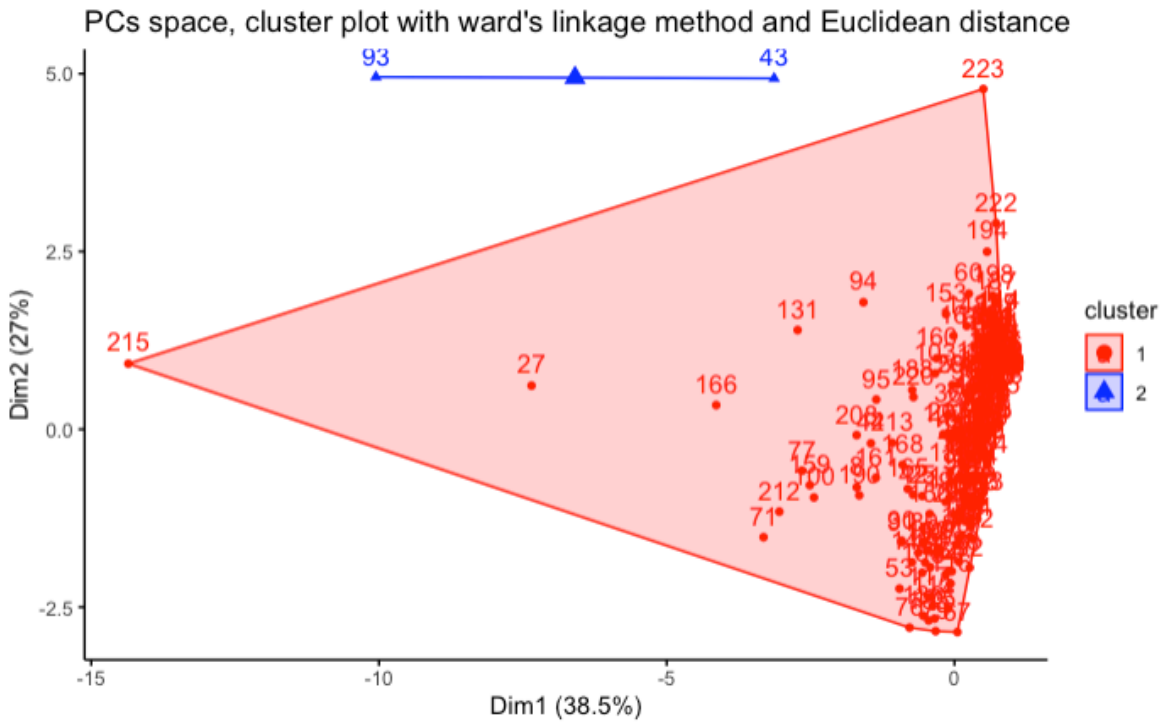


Figure 67: PCs

```
fviz_cluster(list(data=df, cluster=group1),
palette=c("red","blue"),
ellipse.type = "convex",
main = "PCs space, cluster plot
with ward's linkage method
and Euclidean distance",
repel = FALSE, ggtheme=theme_classic())
```

5.2.3 Cluster Algorithm based on complete linkage method and Euclidean distance

```
hc.complete <- hclust(d=dist.eucl, method="complete")
fviz_dend(hc.complete, cex=0.5,
main= "Complete linkage method")
res.coph <- cophenetic(hc.complete)
cor(dist.eucl, res.coph)
fviz_nbclust(df,hcut, method="wss",
distance ="Euclidean ")+labs(subtitle = "Elbow Method")+
geom_vline(xintercept = 2,
linetype=2)
fviz_nbclust(df, hcut, method = "silhouette",
distance="euclidean") +
labs(subtitle = "silhouette method")
fviz_nbclust(df, hcut, method = "gap_stat", distance= "euclidean", nboot = 500) +
labs(subtitle = "Gap statistic method")
```

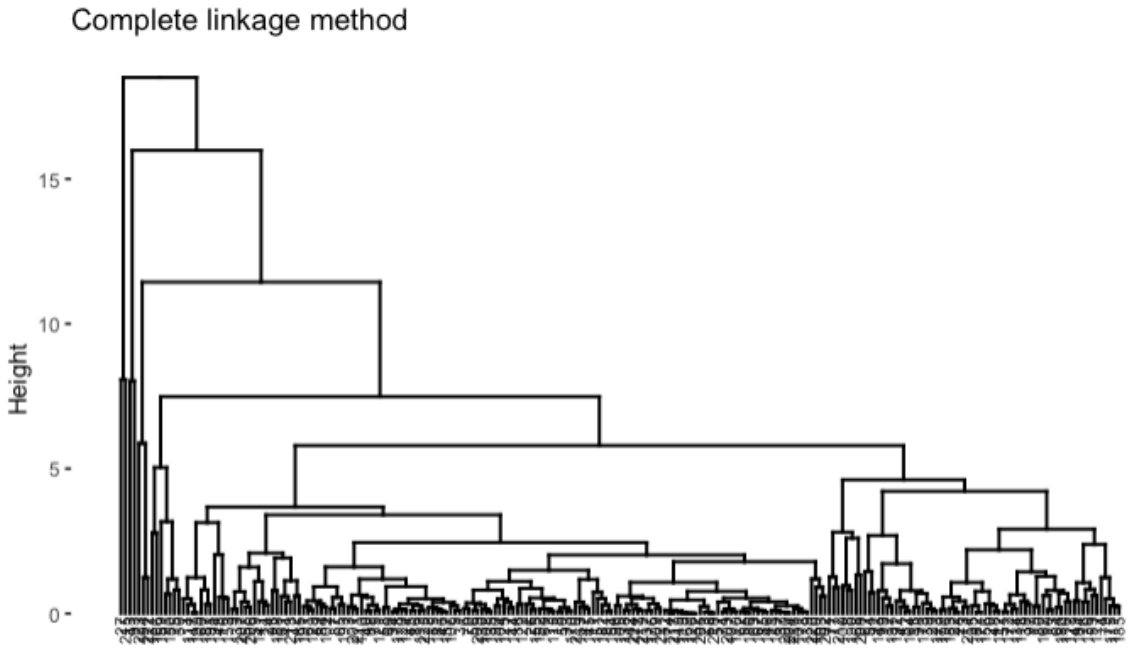
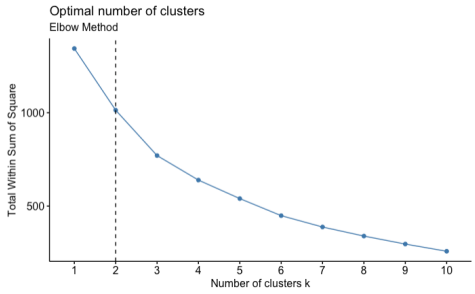
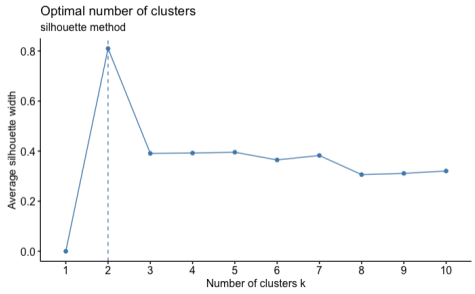


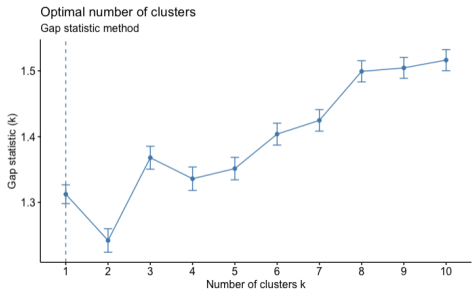
Figure 68: Dendrogram

```
> cor(dist.eucl, res.coph)
[1] 0.8789759
```

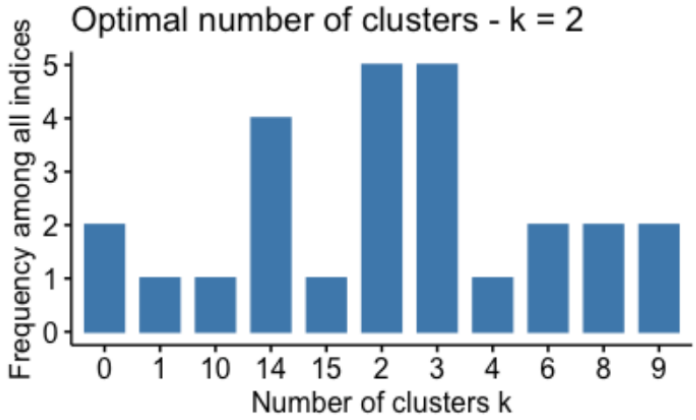
```
fviz_nbclust(v)
group2 <- cutree(hc.complete, k=2)
fviz_dend(hc.complete, k=2, cex = 0.5, k_colors =
c("red","blue"), color_labels_by_k = TRUE, rect = TRUE)


pairs(df, gap=0,
main = "scatter plot matrix with the complete linkage
method and euclidean distance", pch = 21,
bg = c("red","blue")[group2])
fviz_cluster(list(data=df, cluster=group2), palette=c("red","blue"),
ellipse.type = "convex",
main =" PCs space, cluster plot
with the complete
linkage method and euclidean distance",
repel = FALSE, ggtheme = theme_classic())
```








```
*** : 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
* 5 proposed 3 as the best number of clusters
* 1 proposed 4 as the best number of clusters
* 2 proposed 6 as the best number of clusters
* 2 proposed 8 as the best number of clusters
* 2 proposed 9 as the best number of clusters
* 1 proposed 10 as the best number of clusters
* 4 proposed 14 as the best number of clusters
* 1 proposed 15 as the best number of clusters

                ***** Conclusion *****

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



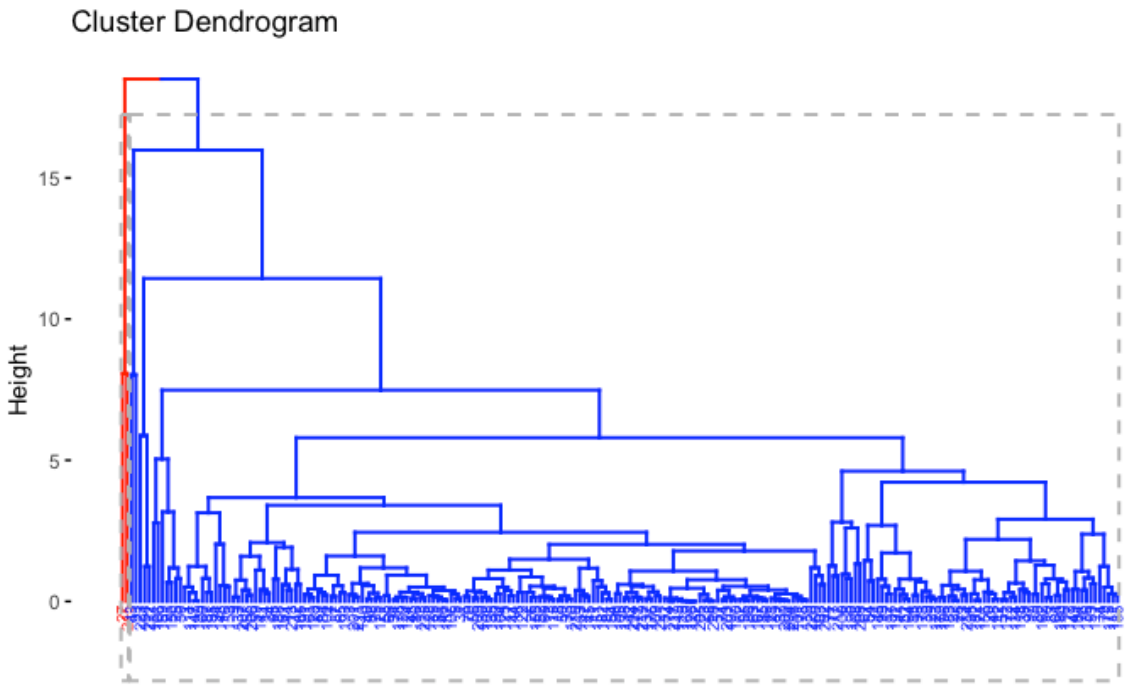


Figure 70: Dendrogram with groups

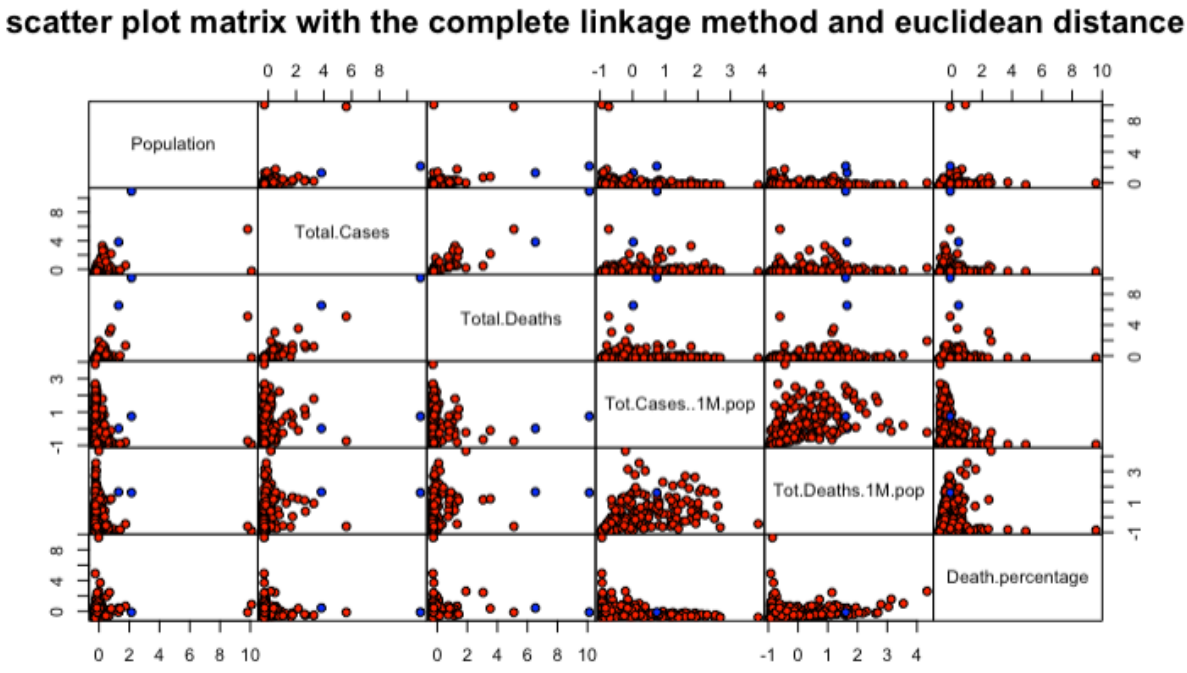


Figure 71: scatterplot

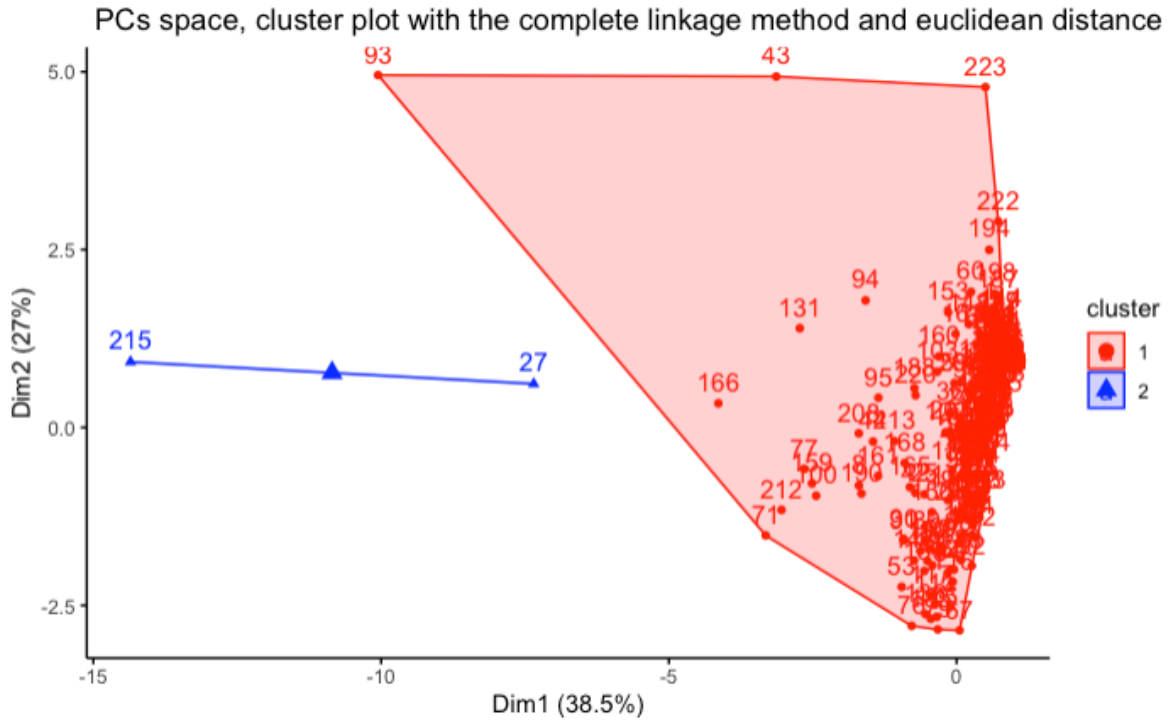


Figure 72: Cluster with k=2

5.2.4 Cluster Algorithm based on average linkage method

```
res.hc <- hclust(d=dist.eucl, method = "average")
fviz_dend(res.hc, cex=0.5, main = "average linkage method")
res.coph <-cophenetic(res.hc)
cor(dist.eucl, res.coph)
fviz_nbclust(df,hcut, method="wss", distance ="Euclidean ")
+labs(subtitle = "Elbow Method")+geom_vline(xintercept = 2, linetype=2)
fviz_nbclust(df, hcut, method = "silhouette", distance="euclidean") +
labs(subtitle = "silhouette method")
fviz_nbclust(df, hcut,
method = "gap_stat", distance= "euclidean", nboot = 500)
+ labs(subtitle = "Gap statistic method")
```

```
group3
  1   2
224   1
> table(group2)
group2
  1   2
223   2
> table(group1)
group1
  1   2
223   2
```

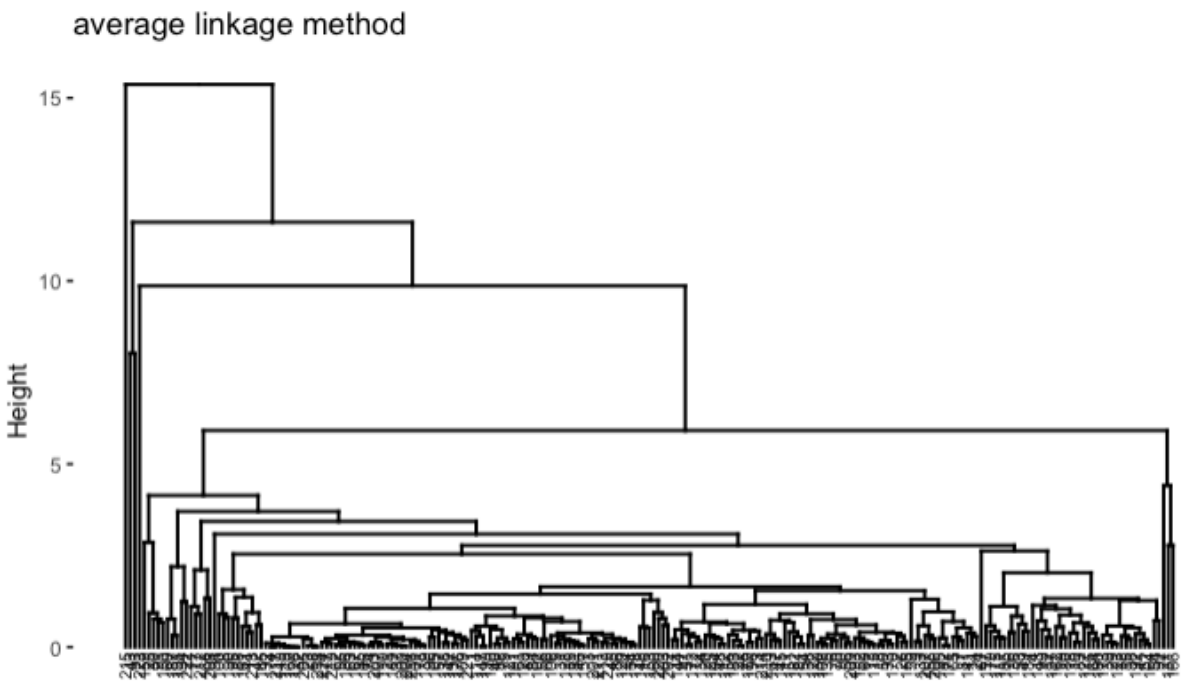
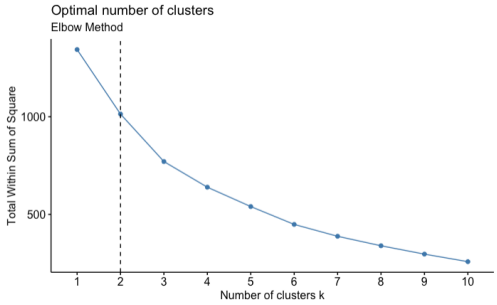
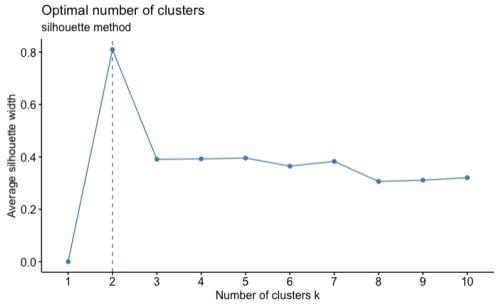
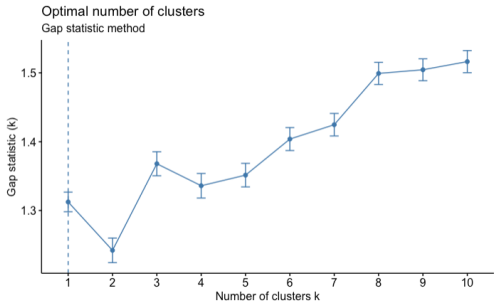


Figure 73: Dendrogram

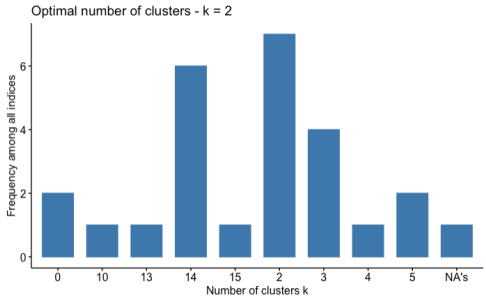
```
nb <- NbClust(df ,diss = NULL,
distance = "euclidean", method = "average")
fviz_nbclust(nb)
group3 <- cutree(res.hc, k=2)
fviz_dend(hc.complete, k=2, cex = 0.5, k_colors = c("red","blue"), color_labels_by_k = TRUE, rect = TRUE
table(group3)
table(group2)
table(group1)
pairs(df, gap=0,
main = "scatter plot matrix with the complete
linkage method and euclidean distance",
pch = 21, bg = c("red","blue")[group3])
fviz_cluster(list(data=df, cluster=group3),
palette=c("red","blue"),
ellipse.type = "convex",
main =" PCs space, cluster plot with the
complete linkage method
and euclidean distance",
repel = FALSE,
ggtheme = theme_classic())
```








```
*****************
* 2 proposed  0 as the best number of clusters
* 7 proposed  2 as the best number of clusters
* 4 proposed  3 as the best number of clusters
* 1 proposed  4 as the best number of clusters
* 2 proposed  5 as the best number of clusters
* 1 proposed  10 as the best number of clusters
* 1 proposed  13 as the best number of clusters
* 6 proposed  14 as the best number of clusters
* 1 proposed  15 as the best number of clusters
* 1 proposed  NA's as the best number of clusters

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



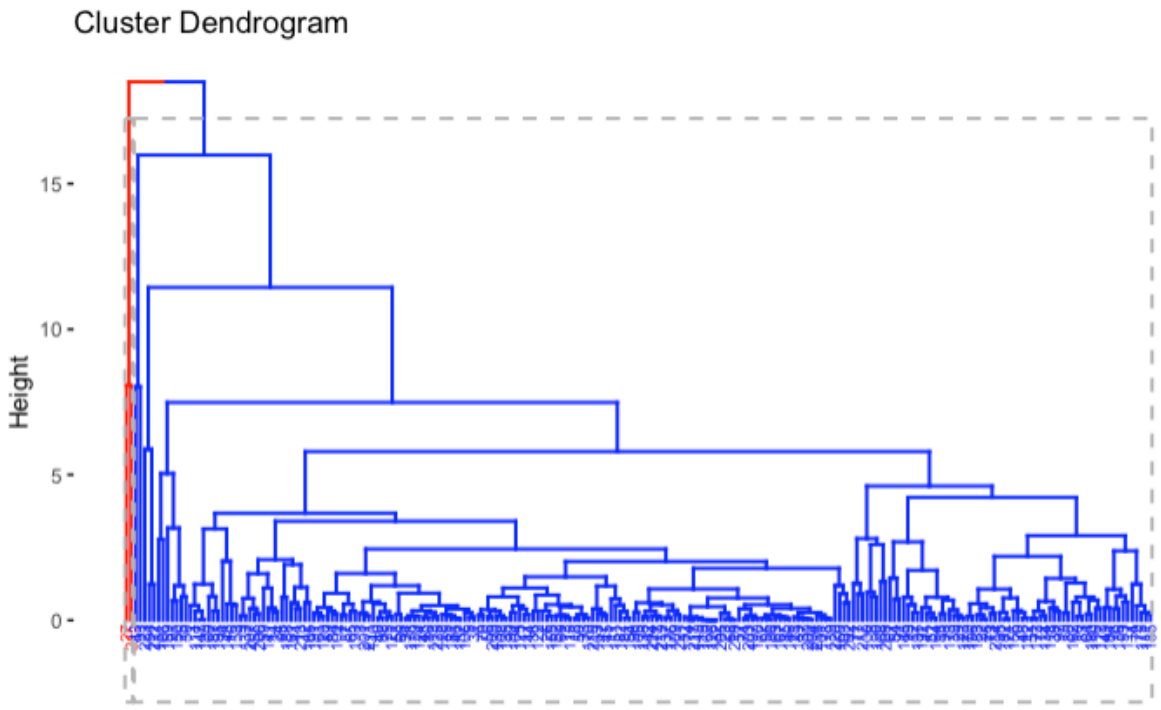


Figure 75: Dendrogram with cluster

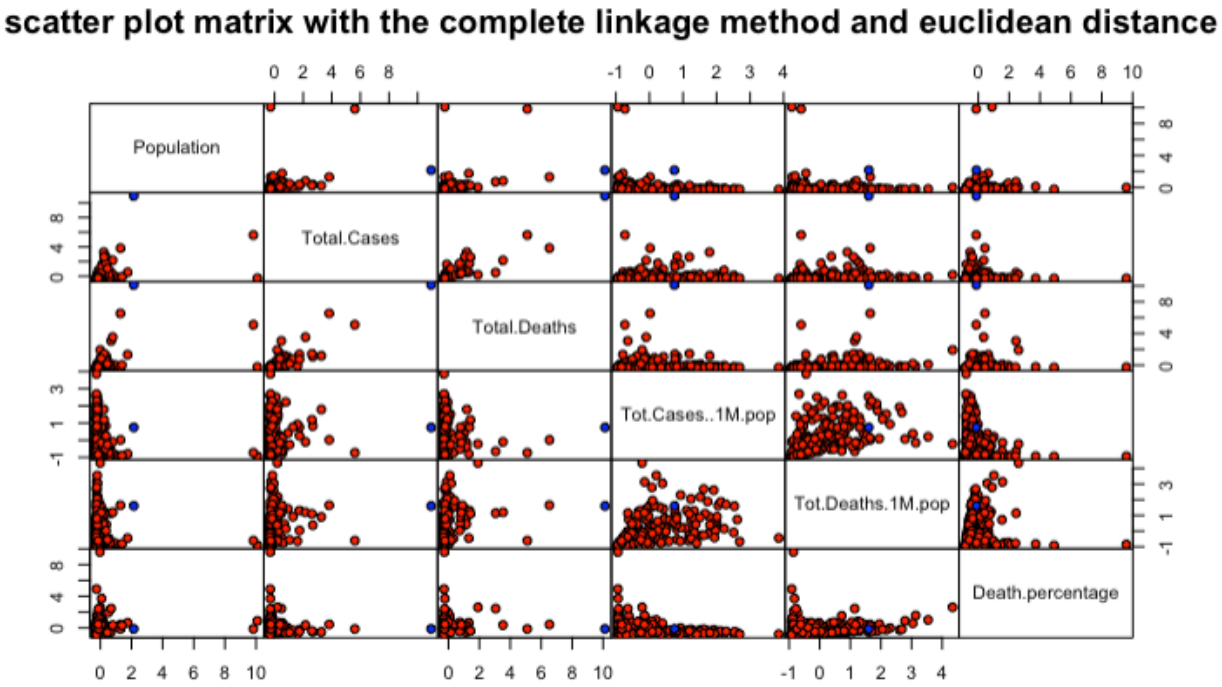


Figure 76: Scatter plot with cluster

Figure 77: Scatter plot with cluster

5.2.5 Cluster Algorithm base on ward's linkage method and manhattan distance

```
res.hc1 <- hclust(d=dist.man, method = "ward.D2")
fviz_dend(res.hc1, cex=0.5
> res.coph <- cophenetic(res.hc1)
> cor(dist.man, res.coph)
[1] 0.3749748
fviz_nbclust(df,hcut, method="wss",
distance ="manhattan ")+labs(subtitle = "Elbow Method")+
geom_vline(xintercept = 2,
linetype=2)
fviz_nbclust(df, hcut,
method = "silhouette", distance="manhattan")
+ labs(subtitle = "silhouette method")
fviz_nbclust(df, hcut,
method = "gap_stat",
distance= "manhattan", nboot = 500)+
labs(subtitle = "Gap statistic method")
```

```
group4 <- cutree(res.hc1, k=4)
table(group4)
fviz_dend(res.hc1, k=4, cex=0.5, k_colors = c("red","blue","yellow","green"),
color_labels_by_k = TRUE, rect = TRUE)
```

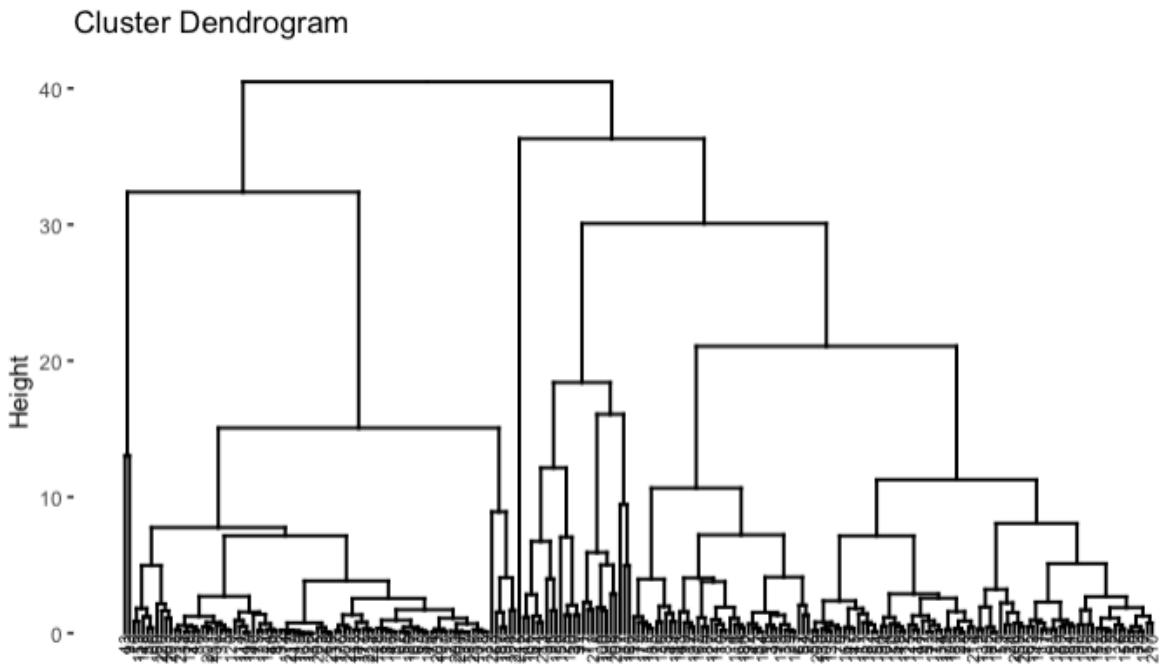
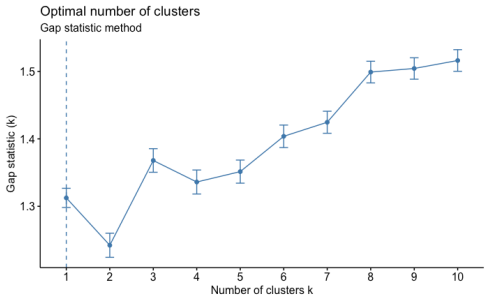
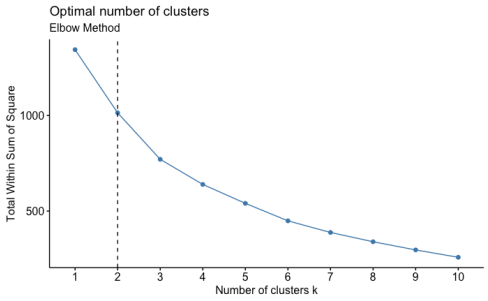
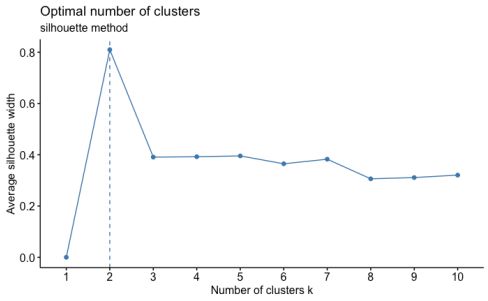


Figure 78: Dendrogram

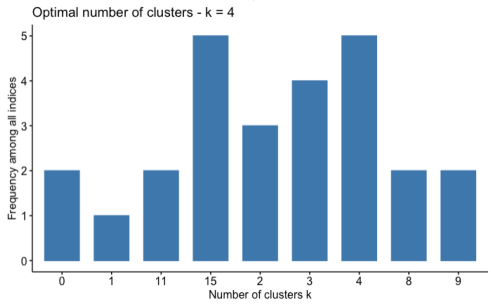
```
pairs(df, gap=0, main = "scatter plot matrix
with the ward's linkage method and euclidean distance",
pch = 21, bg = c("red","blue","yellow","green")[group4])
fviz_cluster(list(data=df, cluster=group4), palette=c("red","blue","yellow","green"),
ellipse.type = "convex", main =" PCs space,
cluster plot with the ward's linkage
method and manhattan distance", repel = FALSE, ggtheme = theme_classic())
```








```
                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
* 4 proposed 3 as the best number of clusters
* 5 proposed 4 as the best number of clusters
* 2 proposed 8 as the best number of clusters
* 2 proposed 9 as the best number of clusters
* 2 proposed 11 as the best number of clusters
* 5 proposed 15 as the best number of clusters

                  ***** Conclusion *****

* According to the majority rule, the best number of clusters is  4
```



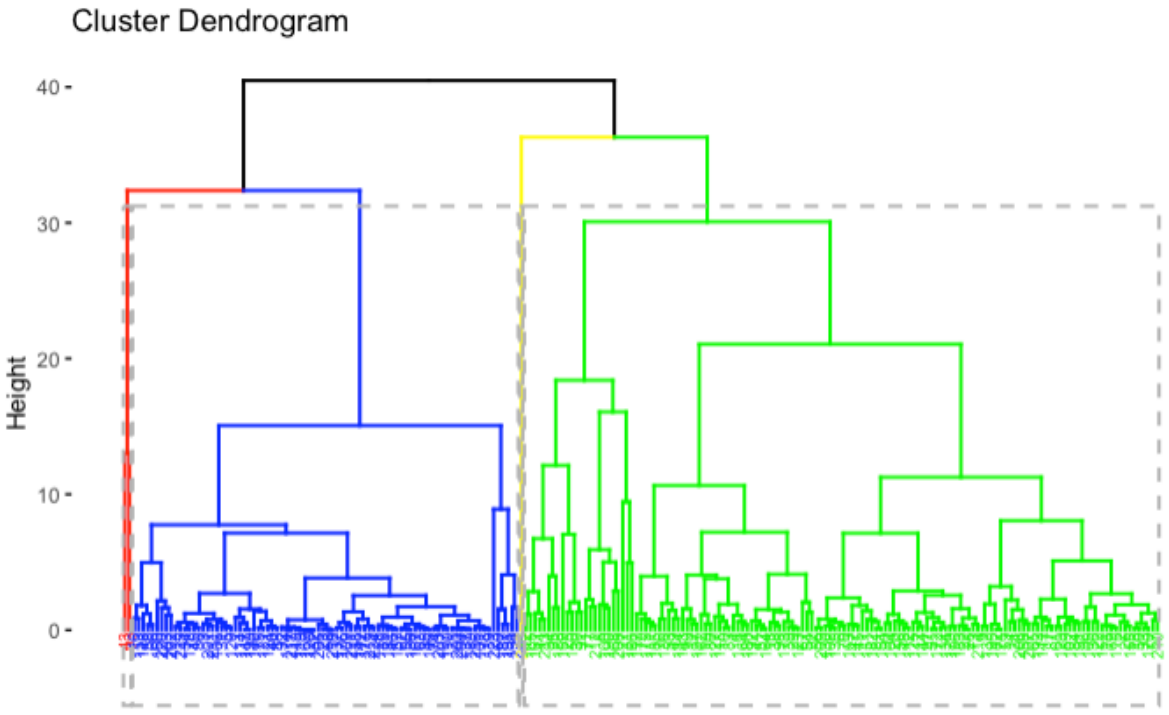


Figure 80: Dendrogram with k=4

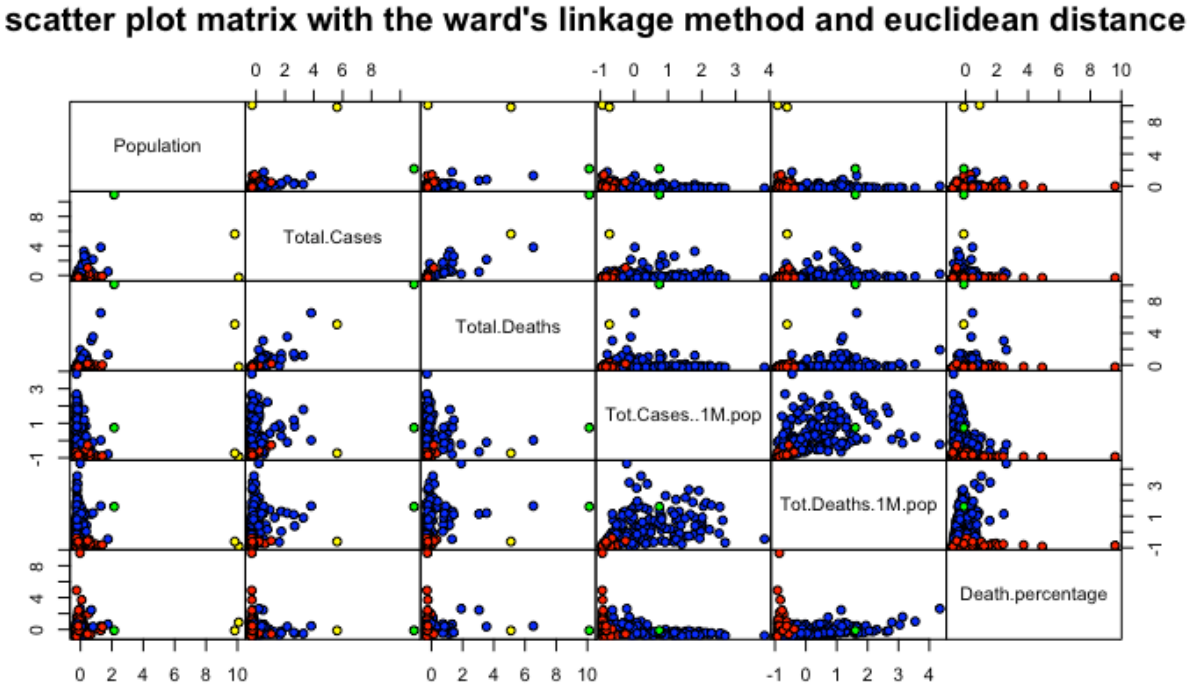


Figure 81: scatterplot with k=4

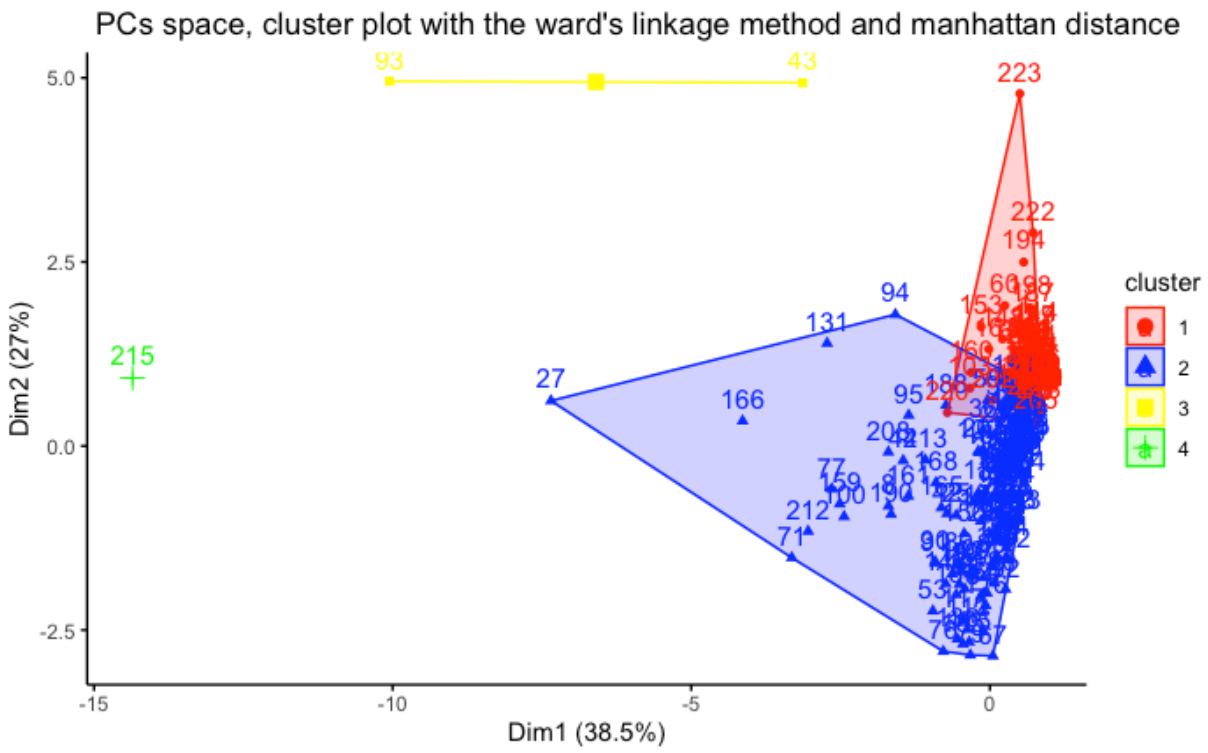


Figure 82: PCs space with k=4

5.2.6 Cluster Algorithm base on single linkage method and manhattan distance

```
hc.single1 = hclust(d=dist.man, method="single")
fviz_dend(hc.single1, cex=0.5, main = "single linkage method")
> res.coph <- cophenetic(res.hc1)
> cor(dist.man, res.coph)
[1] 0.3749748
fviz_nbclust(df,hcut, method="wss",
distance ="manhattan ")+labs(subtitle = "Elbow Method")
+geom_vline(xintercept = 2, linetype=2)
fviz_nbclust(df, hcut,
method = "silhouette", distance="manhattan")
+ labs(subtitle = "silhouette method")
fviz_nbclust(df, hcut,
method = "gap_stat", distance= "manhattan", nboot = 500)
+ labs(subtitle = "Gap statistic method")


group5<-cutree(hc.single1, k=3)
fviz_dend(hc.single1, k=3, cex=0.5, k_colors = c("green","blue","black"),
           color_labels_by_k = TRUE,
          rect = TRUE)
table(group5)
group
  1   2   3
```

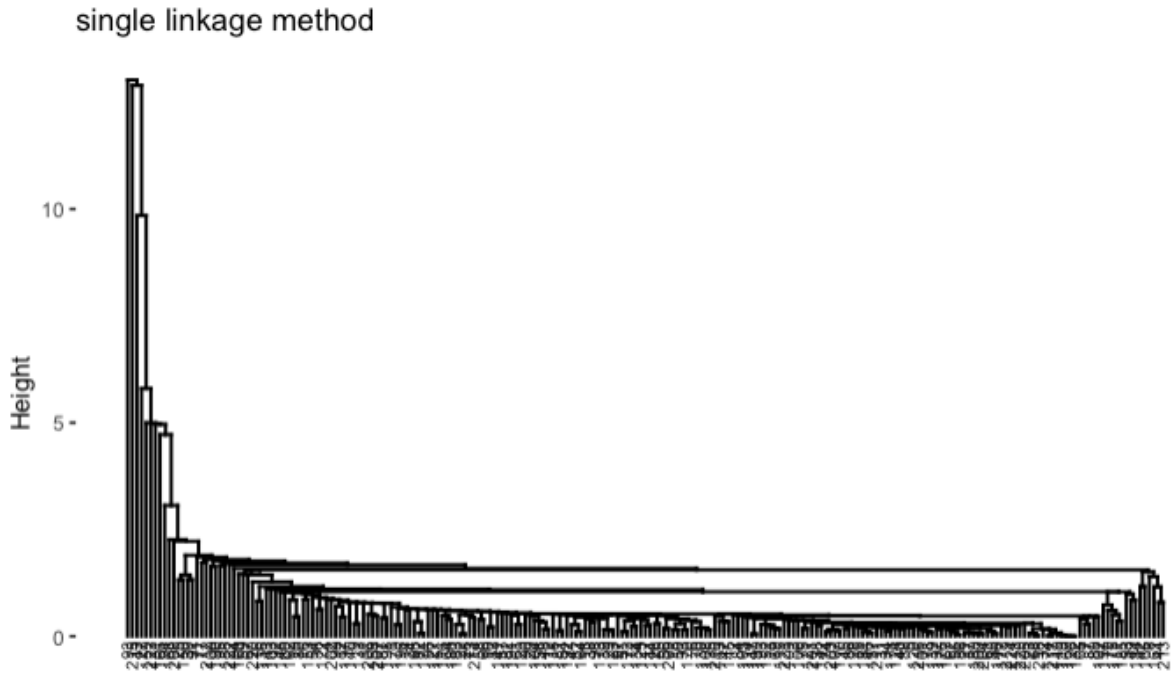
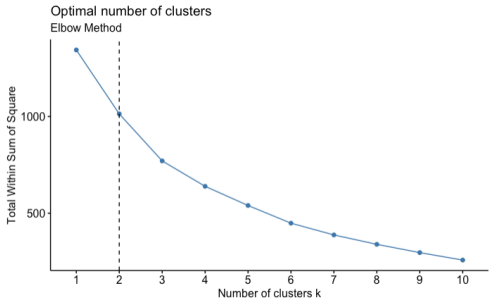
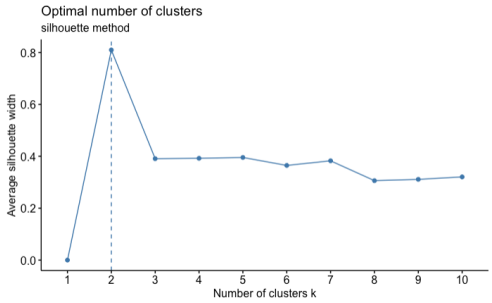
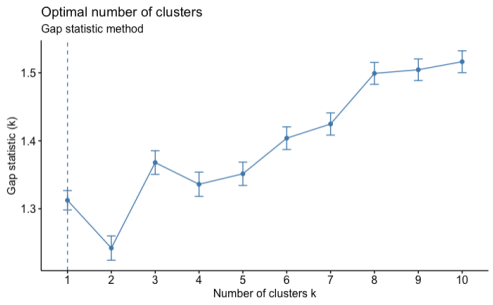


Figure 83: Dendrogram

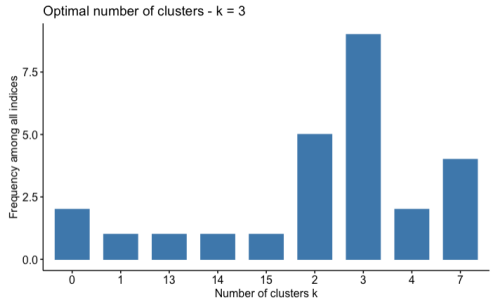
```
155  67   3
pairs(df, gap=0,
main="scatterplot matrix single method
and distance k=3", pch=21, bg=c("green","blue","black")
[group5])
fviz_cluster(list(data=df, cluster=group), palette=c("green","blue","black"),
ellipse.type = "convex", main = "PCs space,
cluser plot with single linkage method and
manhattan distance",
repeal=FALSE, ggtheme=theme_classic())
```








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

                 ***** Conclusion *****

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

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



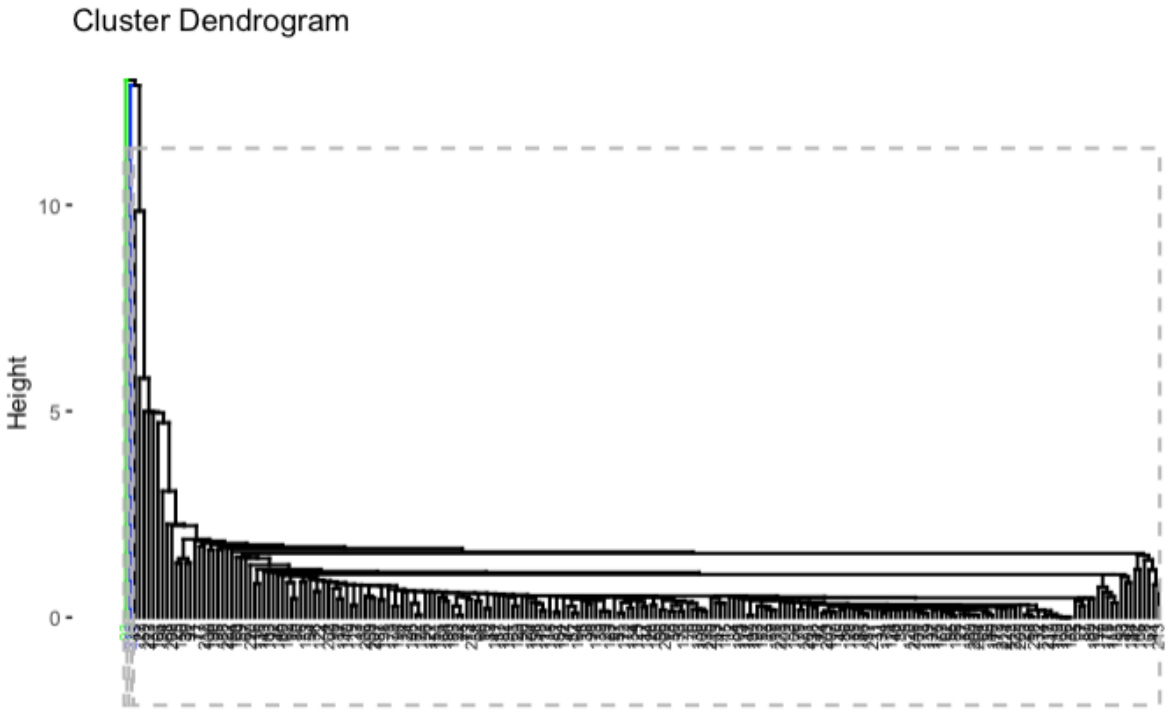


Figure 85: Dendrogram with k=3

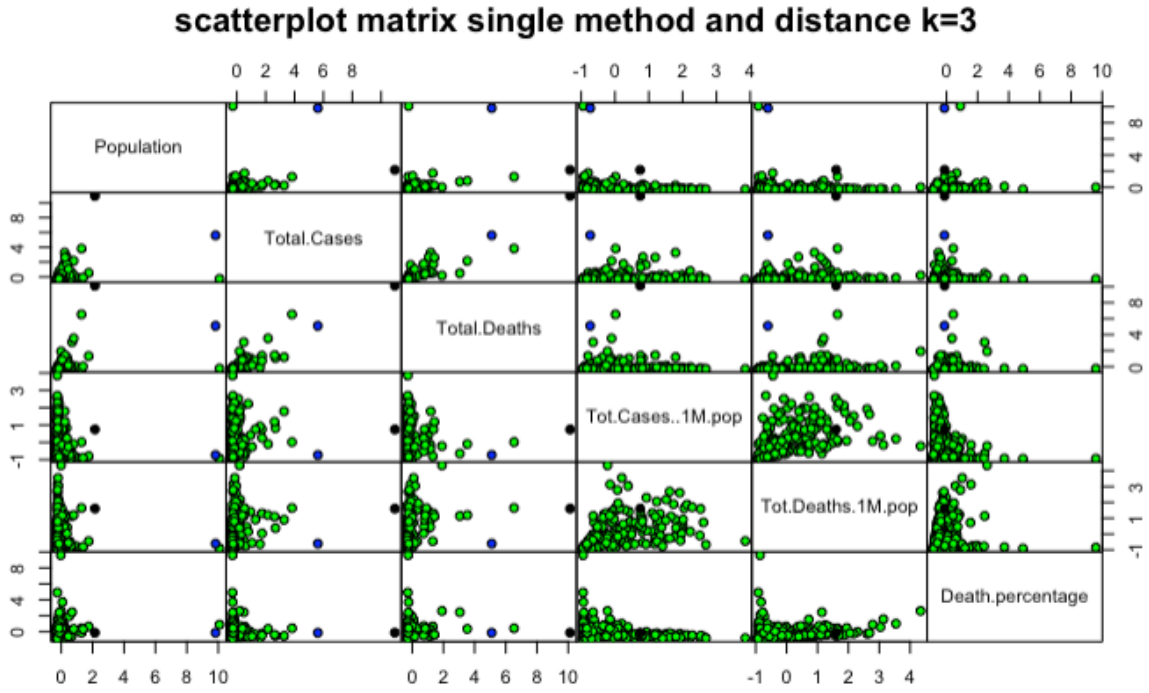


Figure 86: scatterplot with k=3

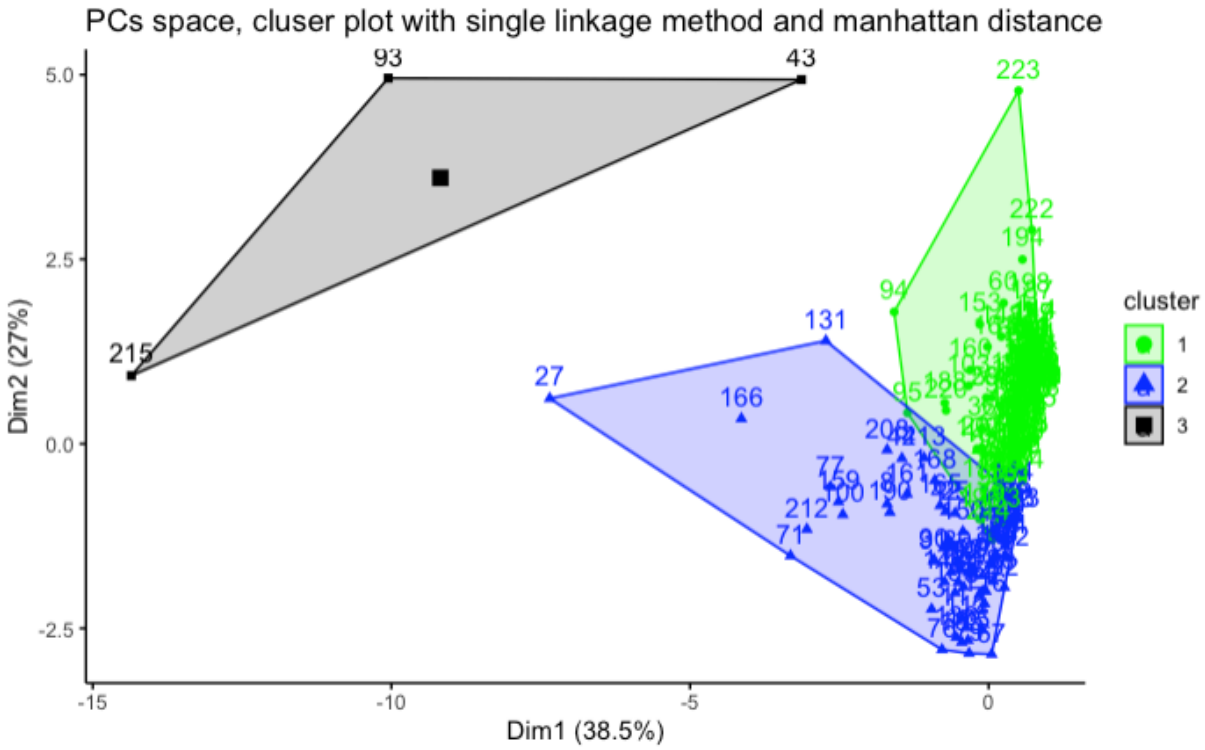


Figure 87: PCs space with k=3

5.2.7 Cluster Algorithm based on complete linkage method and Manhattan distance

```
hc.complete1<-hclust(d=dist.man, method="complete")
fviz_dend(hc.complete1, cex = 0.5, main = "Complete linkage method ")

res.coph<- cophenetic(hc.complete1)
cor(dist.man, res.coph)
[1] 0.9005529

nc3<-NbClust(df, diss=NULL, method = "complete", distance = "manhattan")


group6<-cutree(hc.complete1, k=2)
fviz_dend(hc.complete1, k=2, cex=0.5, k_colors = c("yellow","blue"),
          color_labels_by_k = TRUE,
          rect=TRUE)

table(group6)
group6
  1   2
223   2
pairs(df, gap=0,main = "scatterplot
matrix with complete linkage method and
manhattan distance K = 2",pch=21,
bg=c("yellow","blue")[group6])
```

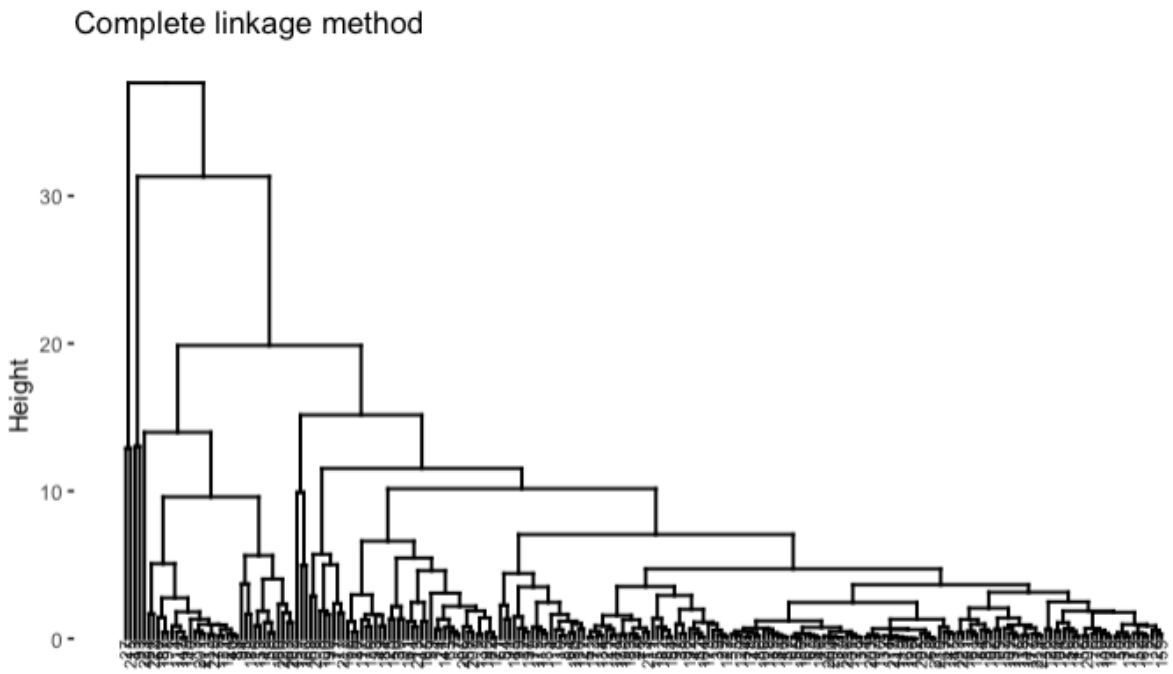
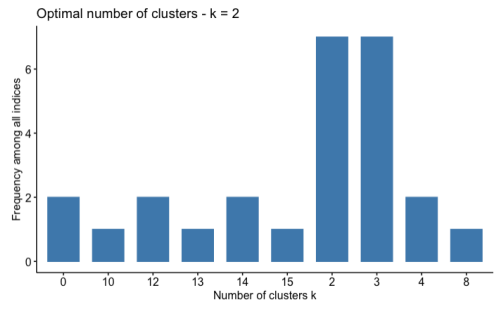


Figure 88: Dendrogram

```
*** : 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
* 7 proposed 3 as the best number of clusters
* 2 proposed 4 as the best number of clusters
* 1 proposed 8 as the best number of clusters
* 1 proposed 10 as the best number of clusters
* 2 proposed 12 as the best number of clusters
* 1 proposed 13 as the best number of clusters
* 2 proposed 14 as the best number of clusters
* 1 proposed 15 as the best number of clusters

                  ***** Conclusion *****

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


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



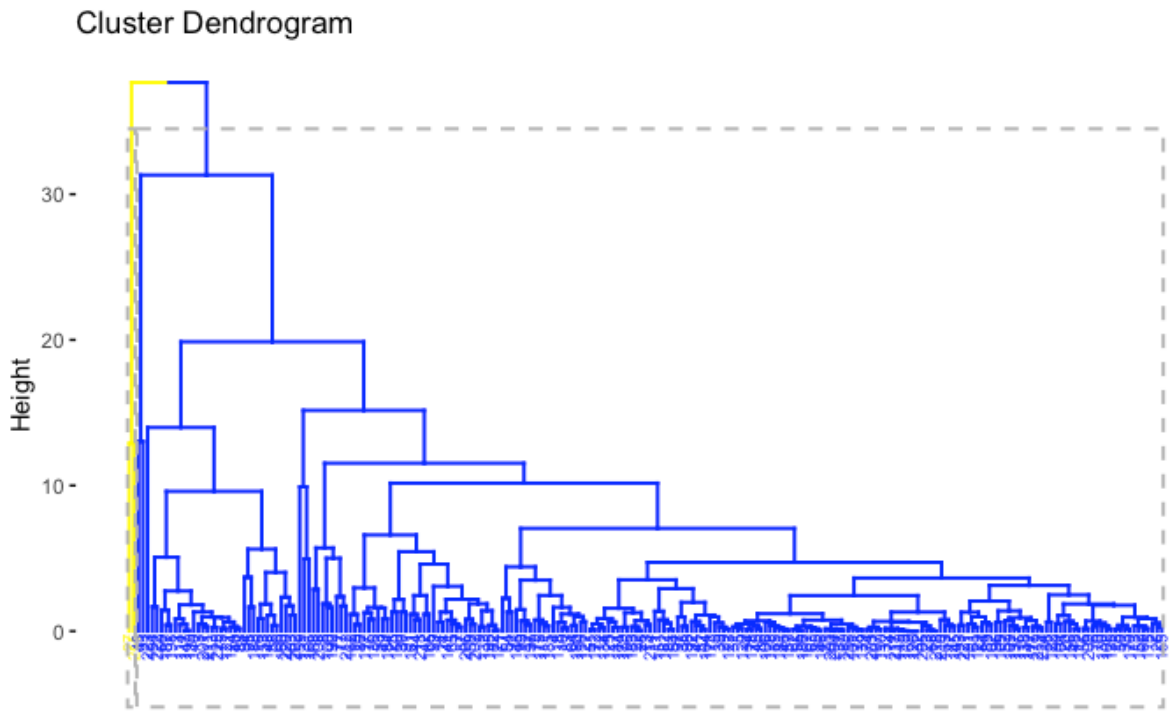


Figure 90: PCs space with k=2

```
fviz_cluster(list(data=df, cluster=group6), palette=c("yellow","blue"),
ellipse.type = "convex",
main = "PCs space, cluster plot with Complete
linkage method and manhattan distance",
repeal= FALSE, ggtheme = theme_classic())
```

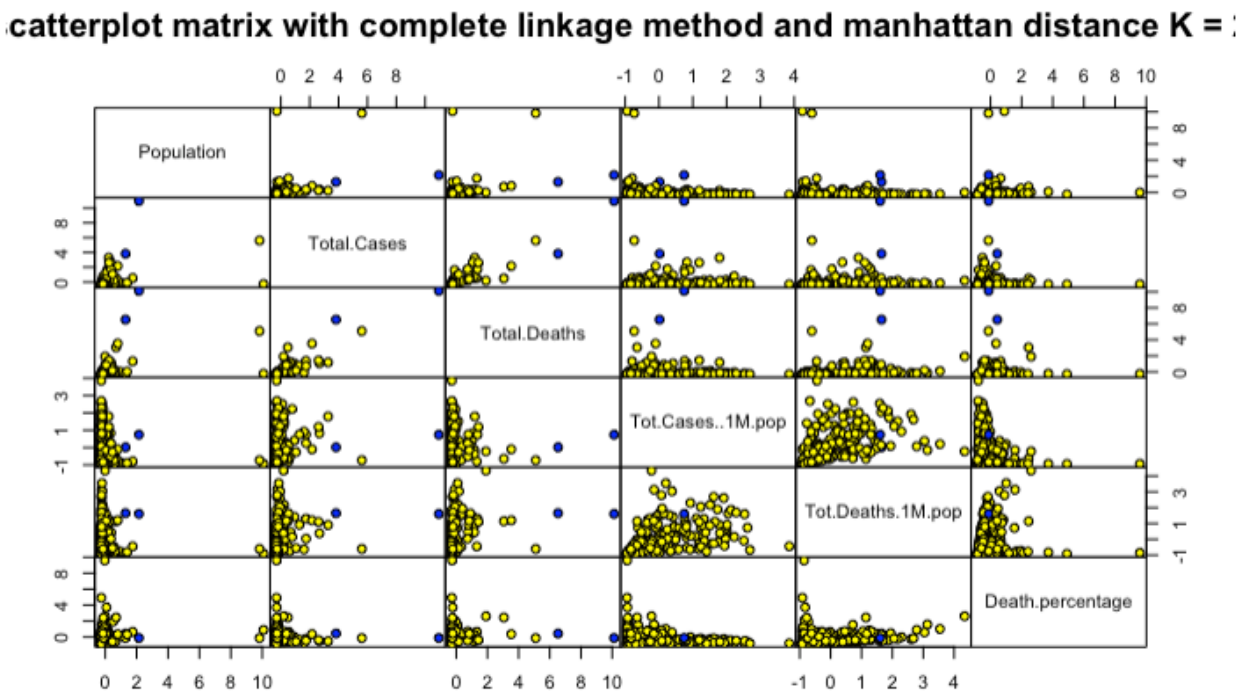


Figure 91: PCs space with k=2

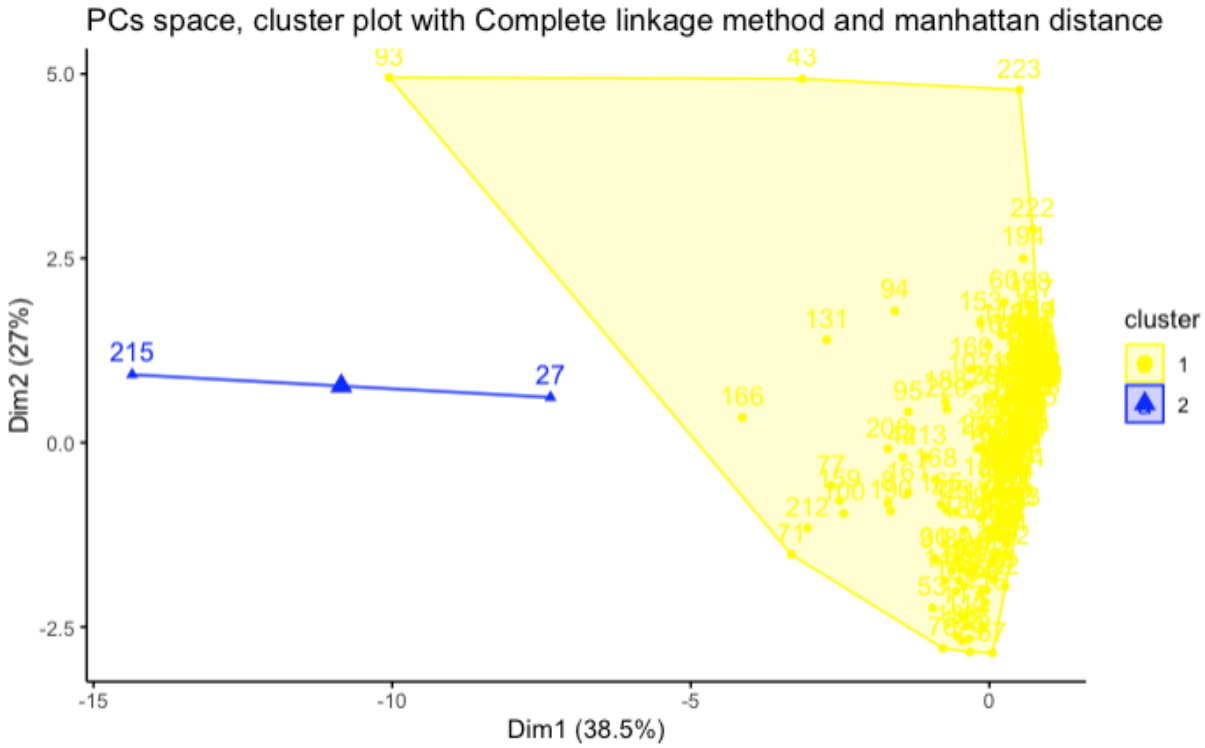


Figure 92: PCs space with k=2

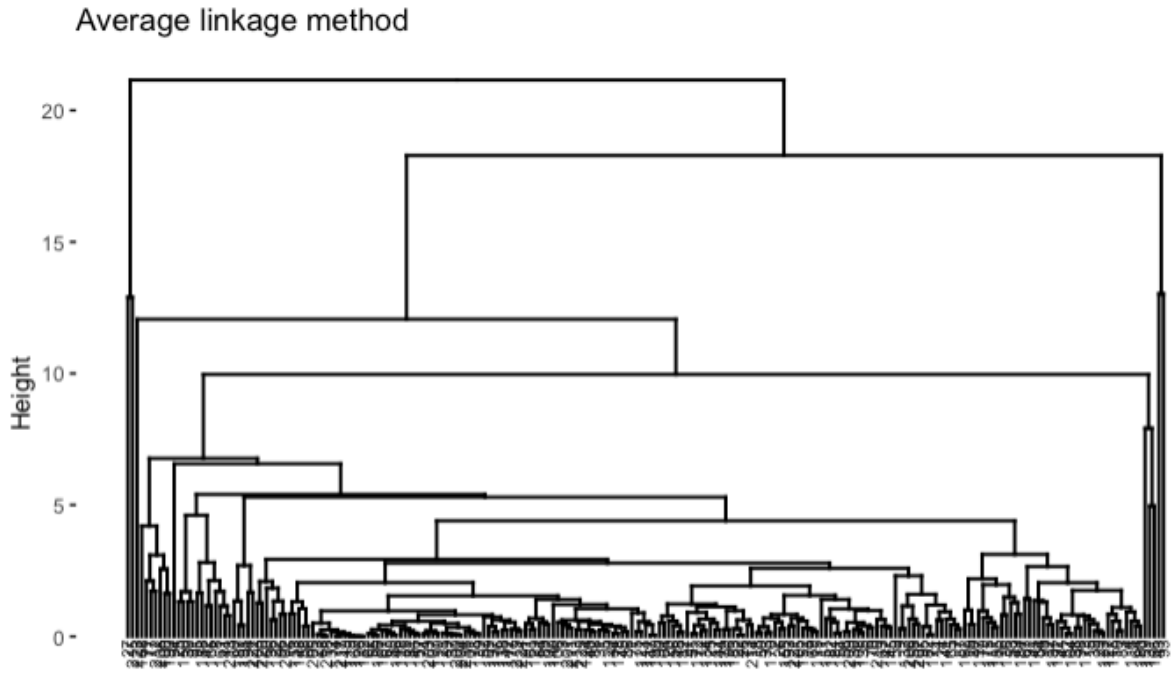
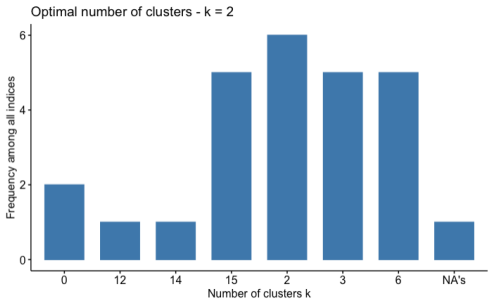


Figure 93: Dendrogram

```
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
* 5 proposed 3 as the best number of clusters
* 5 proposed 6 as the best number of clusters
* 1 proposed 12 as the best number of clusters
* 1 proposed 14 as the best number of clusters
* 5 proposed 15 as the best number of clusters

                  ***** Conclusion *****

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


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



5.2.8 Cluster Algorithm base on average linkage method

```
res.hc3 <- hclust(d=dist.man, method = "average")
fviz_dend(res.hc3, cex=0.5, main = "Average linkage method")
> res.coph<-cophenetic(res.hc3)
> cor(dist.man,res.coph)
[1] 0.9080841
nc4<-NbClust(df, diss=NULL, method="average", distance = "manhattan")
fviz_nbclust(nc4)


group7<-cutree(res.hc3, k=2)
fviz_dend(res.hc3, k=2, cex=0.5, k_colors=c("red","yellow"),
          color_labels_by_k = TRUE,
```

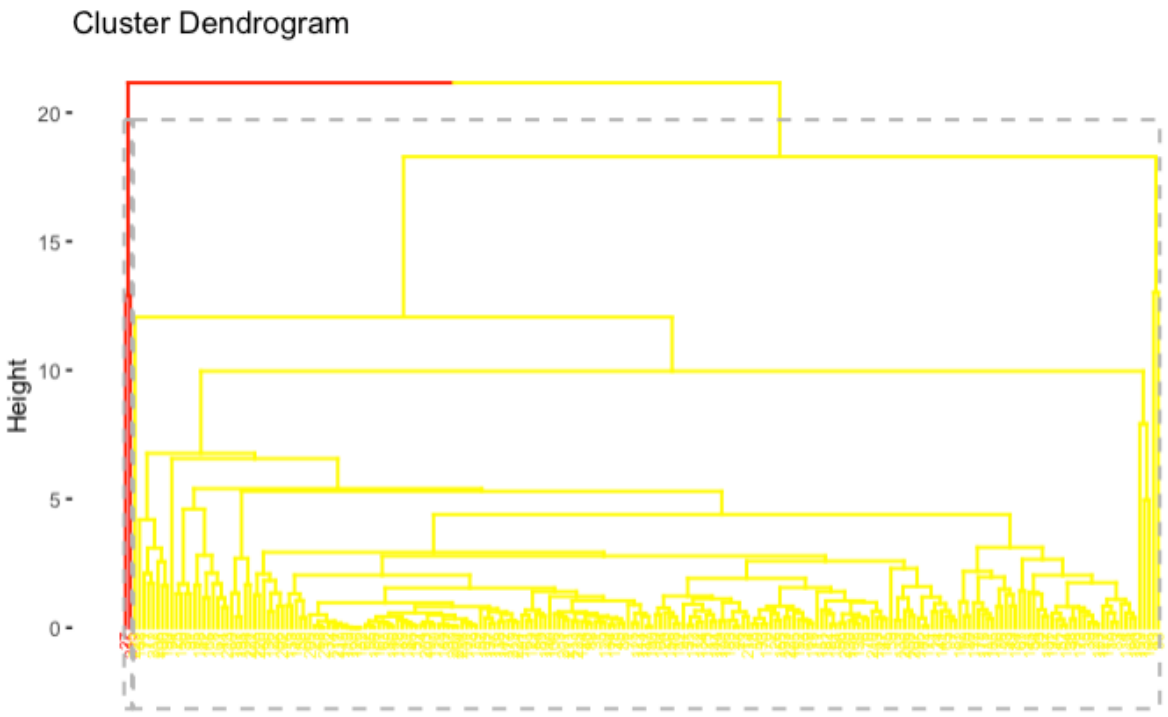


Figure 95: PCs space with k=2

```
          rect=TRUE)
pairs(df, gap=0,
main="scatterplot matrix with average linkage method and manhattan distance distance k=2",
pch=21, bg=c("red","yellow")[group7])
fviz_cluster(list(data=df, cluster=group7), palette=c("red","yellow"),
ellipse.type = "convex",
main="PCs space, cluster plot with average linkage
method and manhattan distance",
repeal=FALSE, ggtheme = theme_classic())
```

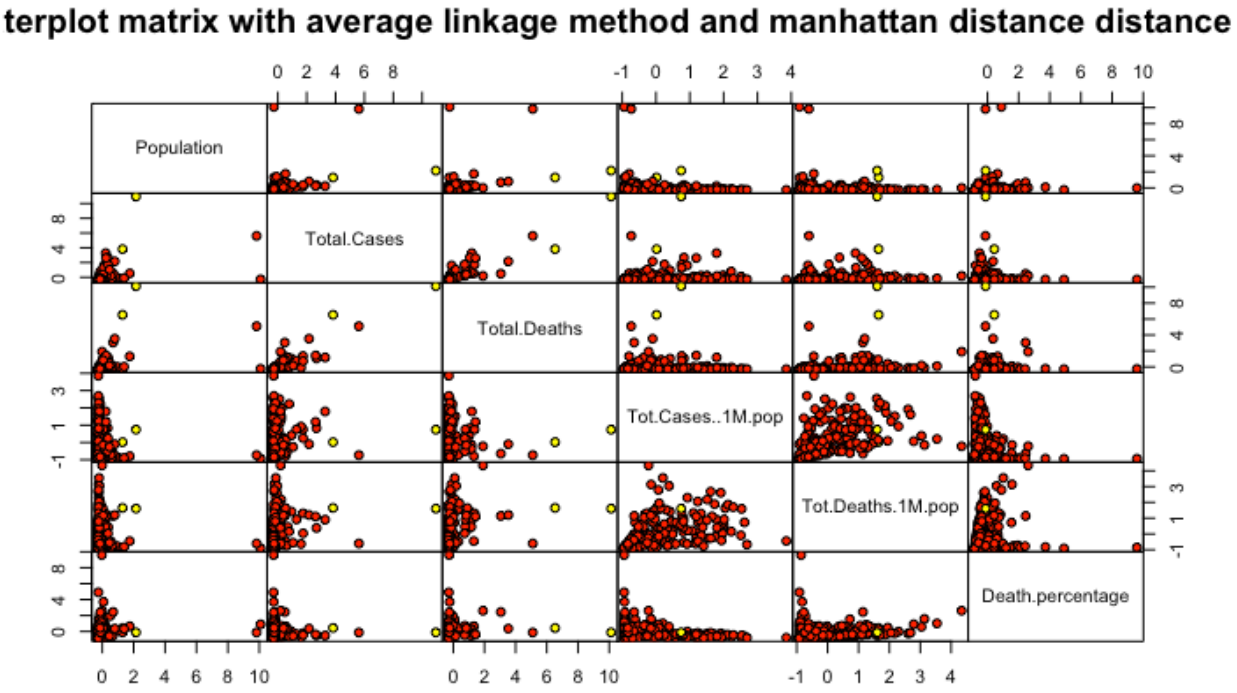


Figure 96: PCs space with k=2

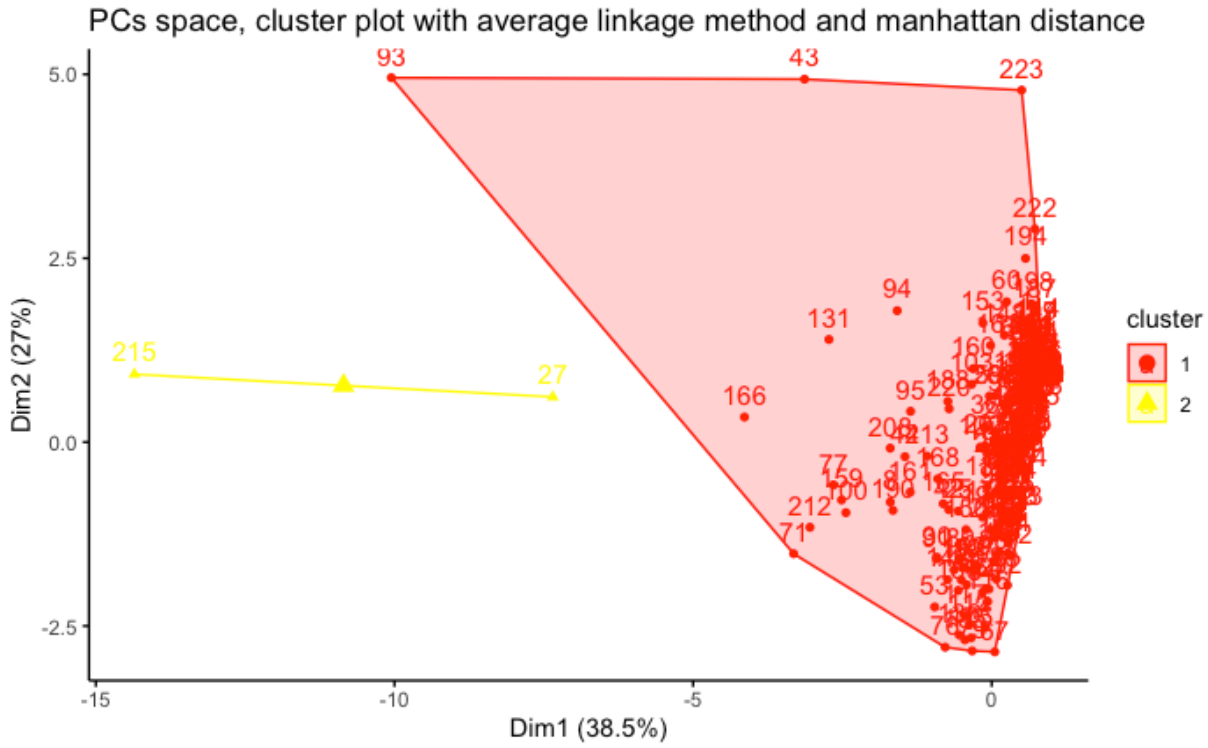


Figure 97: PCs space with k=2

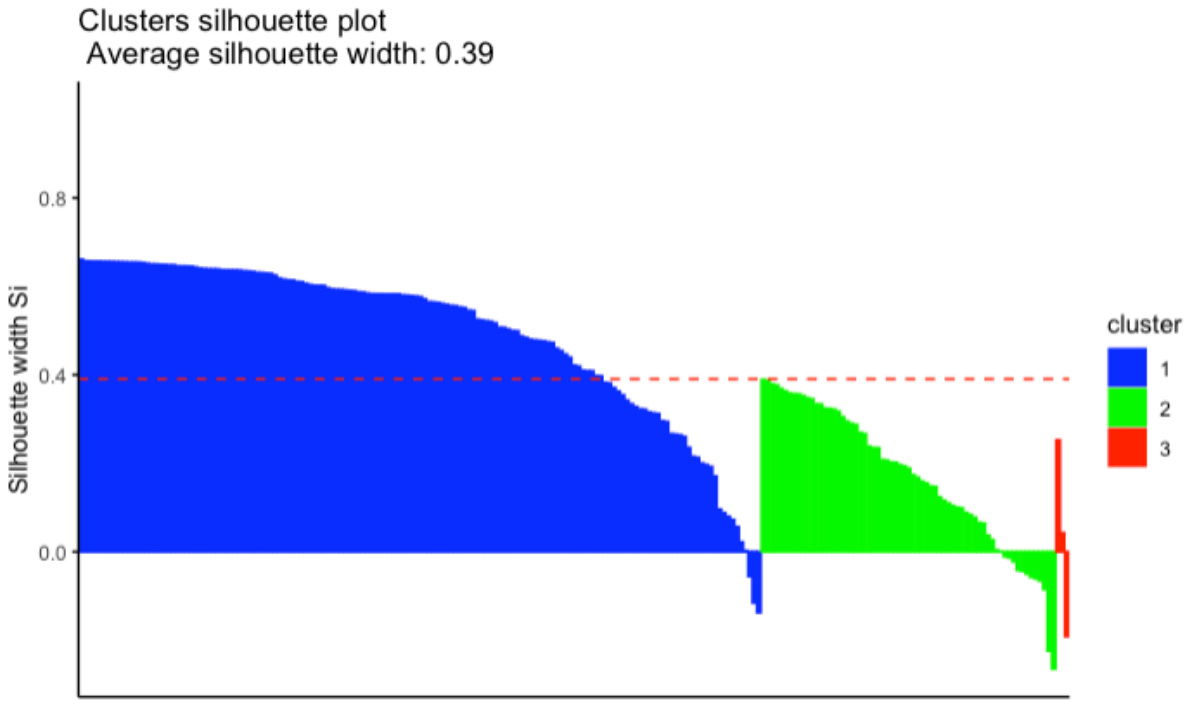


Figure 98: Average silhouette

5.3 CLUSTER VALIDATION STATISTICS

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

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

5.3.1 Internal Cluster Validation based on Ward's linkage method and euclidean distance

The following step consist of evaluate the clustering result, starting from the internal cluster validation of the cluster algorithm with Ward's linkage method and Euclidean distance. The silhouette width is an approach that aims to measure the separation and the cohesion of the clusters to be analysed. His values are included in the interval [-1;1]; we can affirm that values close to 1 indicates that units are well clustered, instead if the values are closer to -1 or 0, we can say that the clustering results are not good.

```
hclust<-eclust(df, "hclust", k=3,
hc_metric = "euclidean", hc_method = "ward.D2", graph = FALSE)
fviz_silhouette(hclust, palette=c("blue","green","red"), ggtheme=theme_classic())
```

```
 Cluster  Size    ave.sil.width

1    155    0.49
2    67    0.17
3    3    0.03
hclust$silinfo$clus.avg.widths
hclust$silinfo$avg.width
[1] 0.3907996
```

From the silhouette we can extract the average silhouette widths of both clusters (close to 1 is well clustered). Next to be computed is the DUNN index. The DUNN index is an internal validation measure. A higher Dunn Index will indicate compact, well-separeted clusters, while a lower index will indicate less compact or less well-separated clusters.

```
hc_stats <- cluster.stats(dist(df),hclust$cluster)
hc_stats$dunn
[1] 0.01284207
```

5.3.2 Internal Cluster validation based on single linkage and Euclidean distance

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

```
hclust2<-eclust(df,"hclust", k=3,
hc_metric = "euclidean",
hc_method = "single", graph = FALSE)
fviz_silhouette(hclust2,
palette=c("blue","green","red"), ggtheme=theme_classic())
cluster size   avg.sil.width
1     222    0.80
2      2    0.25
3      1    0.00
```

from the silhouette we can extract the average silhouette widths of both clusters and then the DUNN index:

```
hclust2$silinfo$clus.avg.widths
[1] 0.8008506 0.2505237 0.0000000
hclust2$silinfo$avg.width
[1] 0.7923995
hc_stats1<-cluster.stats(dist(df),hclust2$cluster)
hc_stats$dunn
[1] 0.01284207
```

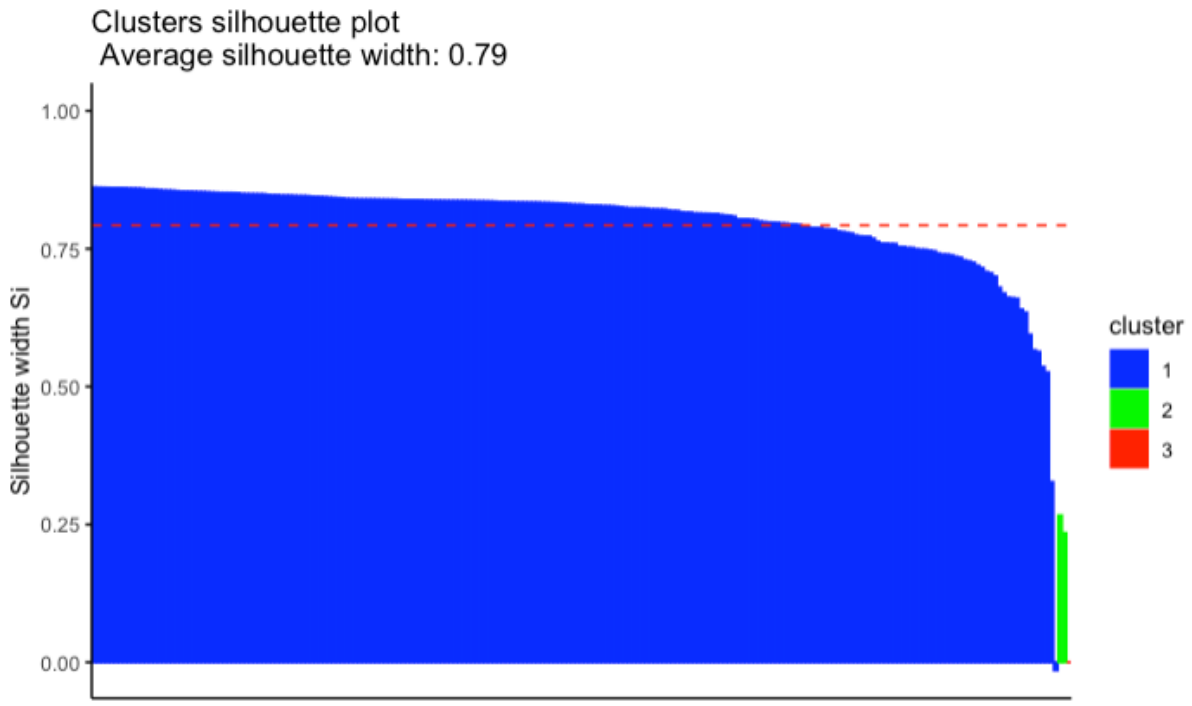


Figure 99: Average silhouette

5.3.3 Internal Cluster Validation based on complete linkage method and Euclidean distance

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

```
hclust3<-eclust(df,"hclust",k=2,
hc_metric = "euclidean",hc_method = "complete", graph=FALSE)
fviz_silhouette(hclust3, palette=c("red","yellow"),ggtheme=theme_classic())
hclust3$silinfo$clus.avg.widths
[1] 0.7933449 0.2464234
hclust3$silinfo$avg.width
[1] 0.7884834
hc_stats3<-cluster.stats(dist(df),hclust3$cluster)
hc_stats3$dunn
[1] 0.2187332
cluster size avg.sil.width
1     223       0.79
2       2       0.25
```

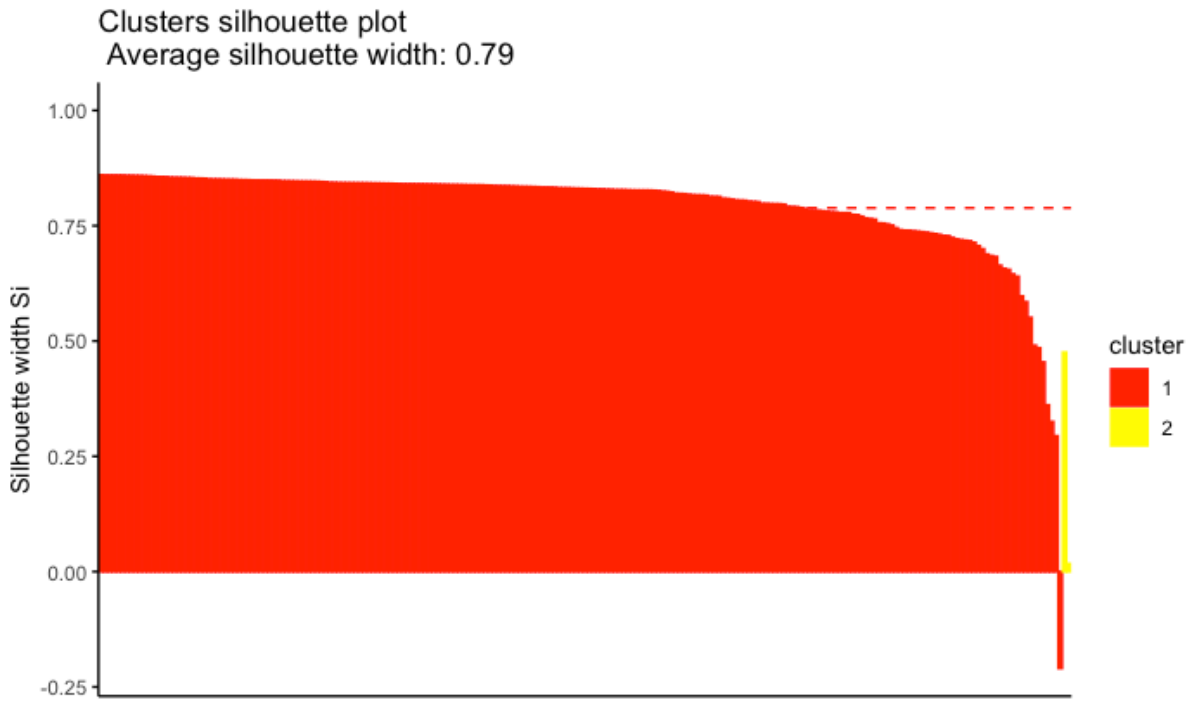


Figure 100: Average silhouette

5.3.4 Internal Cluster Validation based on Average linkage method and Euclidean Distance

The following step consist of evaluate the clustering result, starting from the internal cluster validation of the cluster algorithm with average linkage method and Euclidean distance. Compute starting from the silhouette width:

```
hclust4<-eclust(df,"hclust",k=2,
hc_metric = "euclidean",hc_method = "average", graph=FALSE)
fviz_silhouette(hclust4, palette=c("blue","red"), ggtheme=theme_classic())
hclust4$silinfo$clus.avg.widths
[1] 0.8355399 0.0000000
hclust4$silinfo$avg.width
[1] 0.8318264
hc_stats4<- cluster.stats(dist(df),hclust4$cluster)
hc_stats4$dunn
[1] 0.5052005

cluster size avg.sil.width
1 224 0.84
2 1 0.00
```

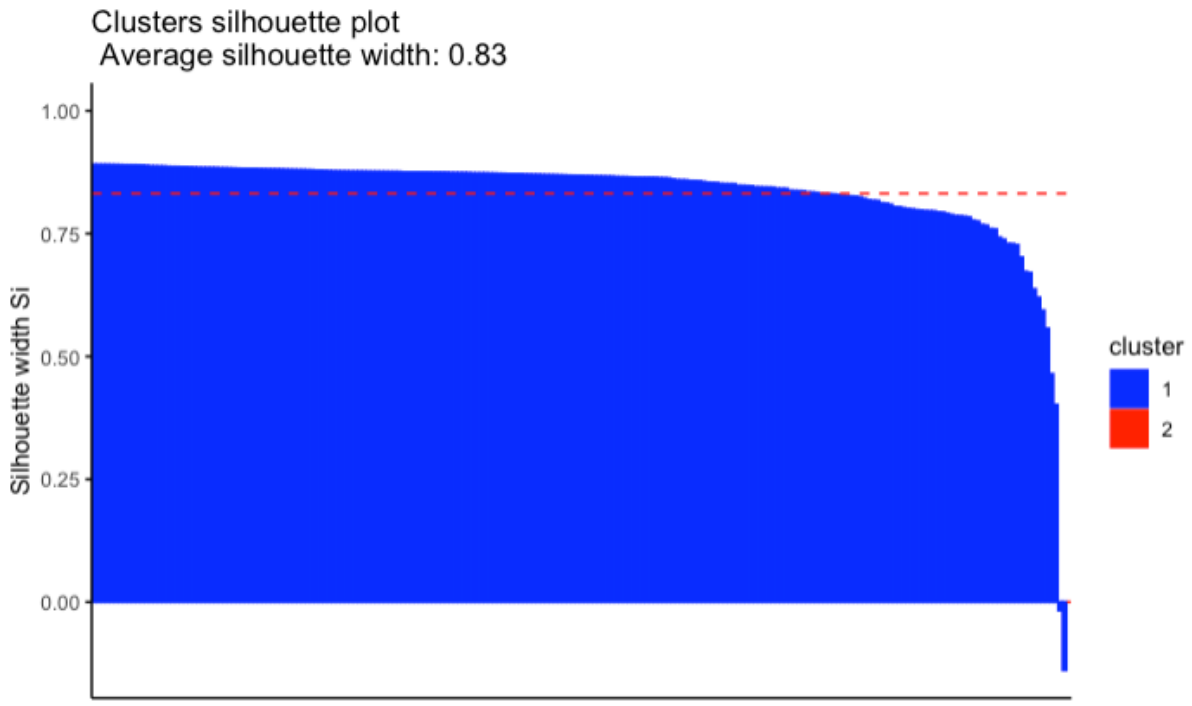


Figure 101: Average silhouette

5.3.5 Internal Cluster Validation based on ward's linkage method and Manhattan distance

The following step consist of evaluate the clustering result, starting from the internal cluster validation of the cluster algorithm with Ward's linkage method and Manhattan distance. Let's compute the silhouette width:

```
hclust5<-eclust(df,"hclust",k=4,
hc_metric = "manhattan", hc_method = "ward.D2", graph=FALSE)
fviz_silhouette(hclust5,
palette=c("red","yellow","blue","black"),
ggtheme=theme_classic())
hclust5$silinfo$clus.avg.widths
[1] 0.61359053 0.07772059 0.14479425 0.00000000
hclust5$silinfo$avg.width
[1] 0.2780295
hc_stats5<-cluster.stats(dist(df),hclust$cluster)
hc_stats5$dunn
[1] 0.01284207
cluster size avg.sil.wdith
1 84 0.61
2 138 0.08
3 2 0.14
4 1 0.00
```

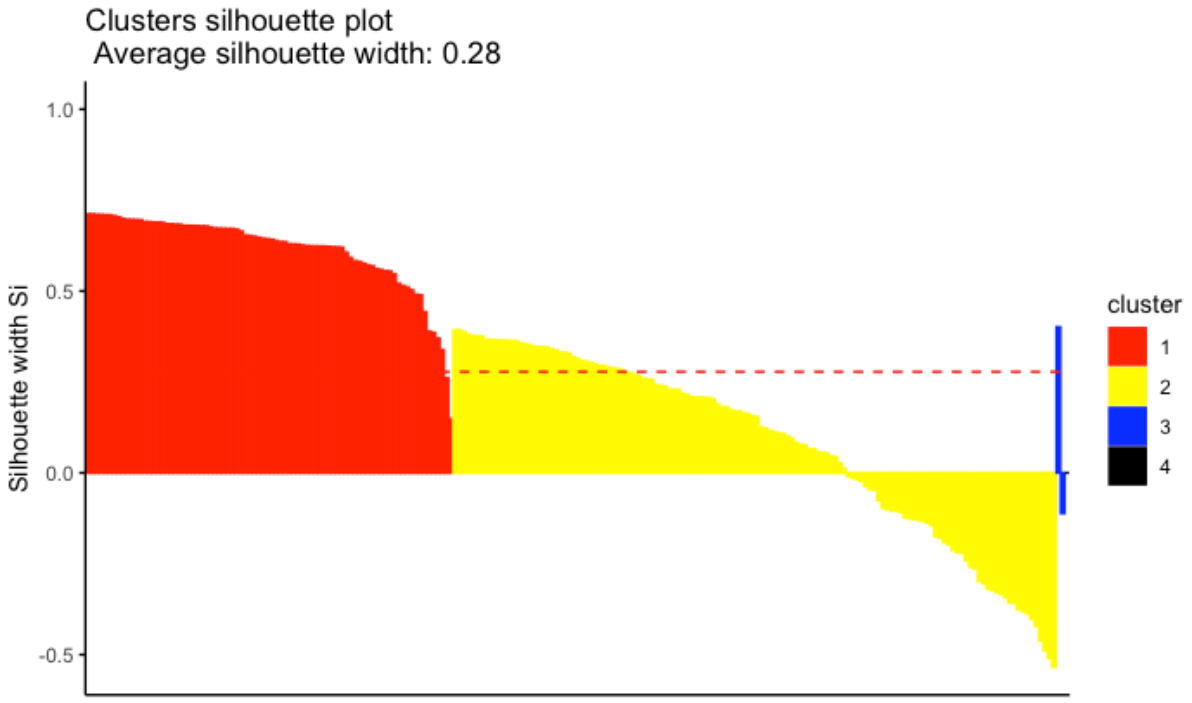


Figure 102: Average silhouette

5.3.6 Internal Cluster Validation based single linkage method and manhattan distance

The following step consist of evaluate the clustering result, starting from the internal cluster validation of the cluster algorithm with single linkage method and Manhattan distance. Let's compute the silhouette width:

```
hclust6<-eclust(df,"hclust",k=3, hc_metric = "manhattan", hc_method = "single", graph=FALSE)
fviz_silhouette(hclust6, palette=c("red","yellow","blue"),ggtheme=theme_classic())
hclust6$silinfo$clus.avg.widths
[1] 0.8215245 0.0000000 0.0000000
hclust6$silinfo$avg.width
[1] 0.8142221
hc_stats6<-cluster.stats(dist(df),hclust6$cluster)
hc_stats6$dunn
[1] 0.601583

cluster  size  ave.sil.width

1 223 0.82
2 1 0.00
3 1 0.00
```

Figure 103: Average silhouette

5.3.7 Internal Cluster Validation based on complete linkage method and Manhattan distance

The following step consist of evaluate the clustering result, starting from the internal cluster validation of the AHC with complete linkage method and Manhattan distance. Let's compute the silhouette width:

```
hclust7<-eclust(df,"hclust",k=2,
hc_metric = "manhattan", hc_method = "complete", graph=FALSE)
fviz_silhouette(hclust7, palette=c("red","yellow"), ggtheme=theme_classic())
hclust7$silinfo$clus.avg.widths
[1] 0.8034609 0.3411199
hclust7$silinfo$avg.width
[1] 0.7993512
hc_stats7<-cluster.stats(dist(df), hclust7$cluster)
hc_stats7$dunn
[1] 0.2187332
cluster size ave.sil.width
1 223 0.80
2 2 0.34
```

5.3.8 Internal Cluster Validation based on average linkage method and Manhattan distance

The following step consist of evaluate the clustering result, starting from the internal cluster validation of the AHC with average linkage method and Manhattan distance. Let's compute the silhouette width:

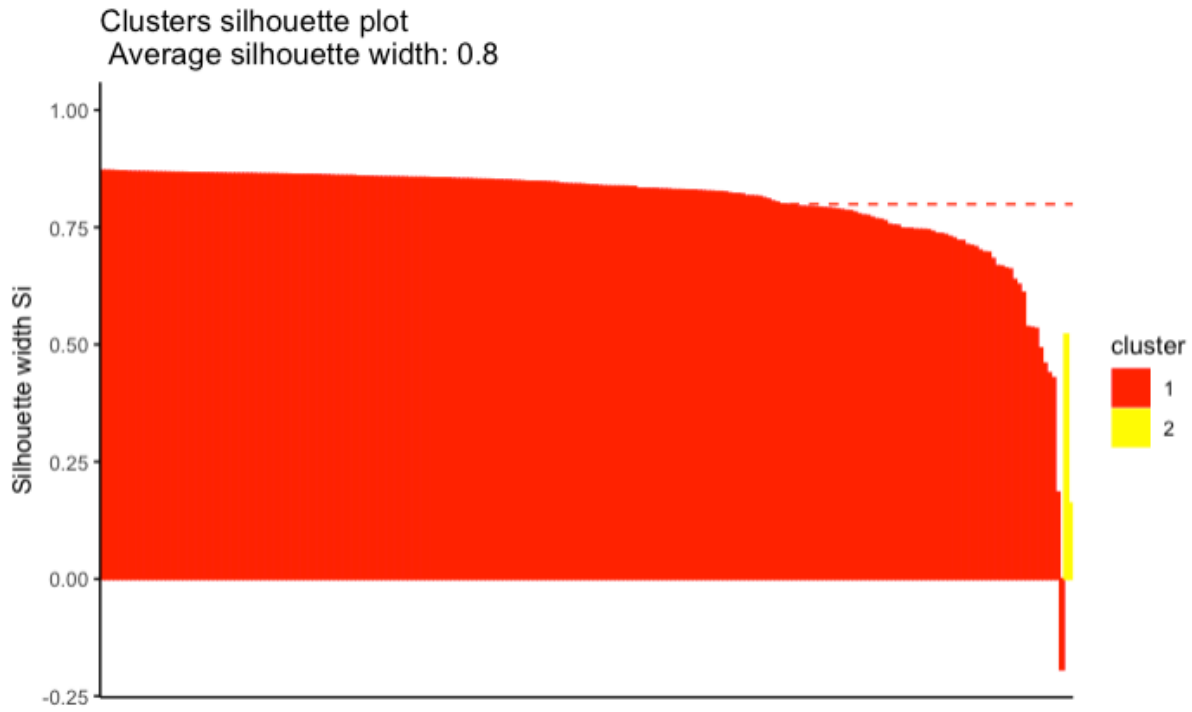


Figure 104: Average silhouette

```
hclust8<-eclust(df,"hclust", k=2,
hc_metric = "manhattan", hc_method = "average", graph=FALSE)
fviz_silhouette(hclust8, palette=c("red","blue"),ggtheme=theme_classic())
hclust8$silinfo$clus.avg.widths
[1] 0.8034609 0.3411199
hclust8$silinfo$avg.width
[1] 0.7993512
hc_stats8<-cluster.stats(dist(df),hclust8$cluster)
hc_stats8$dunn
[1] 0.218733
cluster size ave.sil.width

1 223     0.80
2 2     0.34
```

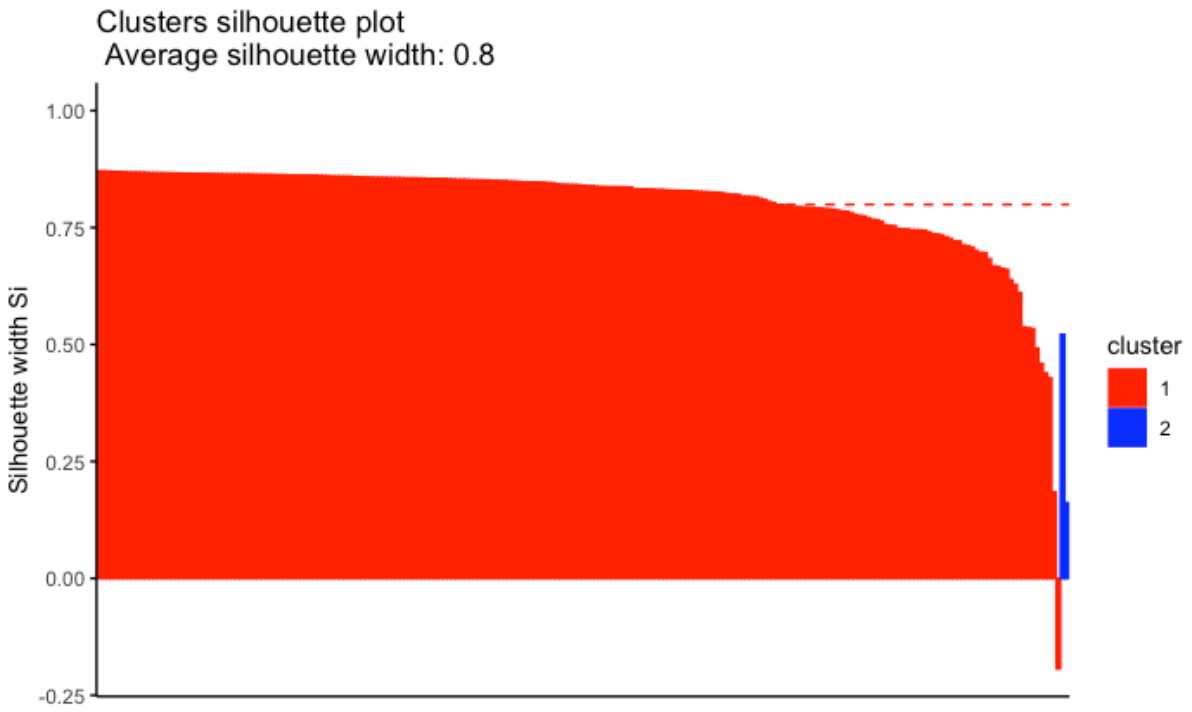


Figure 105: Average silhouette

5.4 PARTIONING CLUSTERING

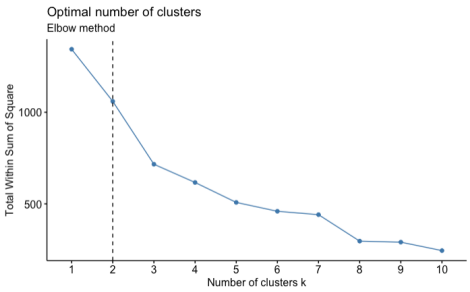
The partitioning clustering approach is another method used to perform the cluster analysis. There are two different types of partitioning clustering:

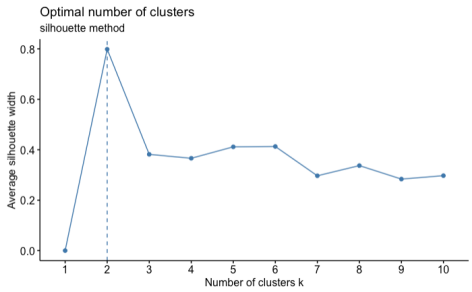
- K-means, that classifies the units to the clusters K, in order to provide a high cluster cohesion and high cluster separation.
- K-medoids, where each cluster is represented by one of the data points (also known as cluster medoids) in the cluster.

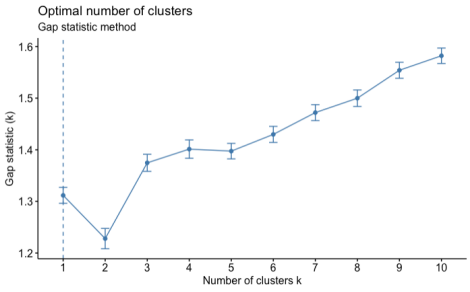
5.4.1 K-Means

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

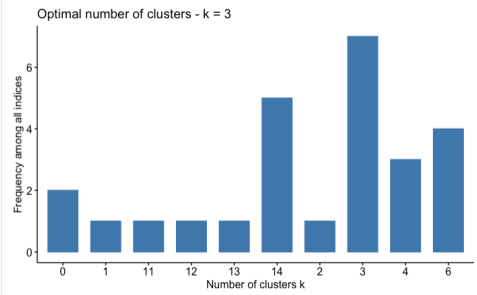
```
fviz_nbclust(df,kmeans, method = "wss")+
  labs(subtitle = "Elbow method")+
  geom_vline(xintercept = 2, linetype=2)
fviz_nbclust(df, kmeans, method="silhouette")+
  labs(subtitle ="silhouette method")
fviz_nbclust(df, kmeans, method="gap_stat",nboot=500)+
  labs(subtitle = "Gap statistic method")
  nb1<-NbClust(df, method="kmeans")
fviz_nbclust(nb1)
```








```
* 5 proposed 14 as the best number of clusters

                ***** Conclusion *****

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


*******************************************************************
Among all indices:
==================
* 2 proposed  0 as the best number of clusters
* 1 proposed  1 as the best number of clusters
* 1 proposed  2 as the best number of clusters
* 7 proposed  3 as the best number of clusters
* 3 proposed  4 as the best number of clusters
* 4 proposed  6 as the best number of clusters
* 1 proposed  11 as the best number of clusters
* 1 proposed  12 as the best number of clusters
* 1 proposed  13 as the best number of clusters
* 5 proposed  14 as the best number of clusters

Conclusion
==========
* According to the majority rule, the best number of clusters is  3 .
```



```
km.res<-kmeans(df,2,nstart=25)
print(km.res)
K-means clustering with 2 clusters of sizes 3, 222

Cluster means:
   Population Total.Cases Total.Deaths Tot.Cases..1M.pop Tot.Deaths.1M.pop Death.percentage
1  4.42429219  6.79470560   7.25047700      0.0093912184        0.88199721     0.0597499227
2 -0.05978773 -0.09182035  -0.09797942     -0.0001269084       -0.01191888    -0.0008074314

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
  2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2
  2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [58] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[115] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[172] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 1 2 2 2 2 2 2 2 2 2 2

Within cluster sum of squares by cluster:
[1]  89.68976 892.20199
 (between_SS / total_SS =  26.9 %)
```

I want to show the obtained clusters in the original space:

```
cl<-km.res$cluster
pairs(df,pch=21,col=c("blue","yellow")[cl])
fviz_cluster(km.res, data=df, geom="point", ellipse.type = "norm", palette=c("blue","yellow"), ggtheme =
```

The following step consist of evaluate the clustering result, starting from the internal cluster validation of the K-means method. First to be computed is the silhouette width.

```
km1<-eclust(df,"kmeans",k=2, graph=FALSE)
fviz_silhouette(km1, palette=c("blue","yellow"),ggtheme=theme_classic())
```

```
km1$silinfo$clus.avg.widths
km1$silinfo$avg.width
km_stats<-cluster.stats(dist(df),km1$cluster)
km_stats$dunn
[1] 0.1818092 0.8070563
[1] 0.7987197
[1] 0.2620914
```

After the internal validation, we compute using the K-Medoids approach.

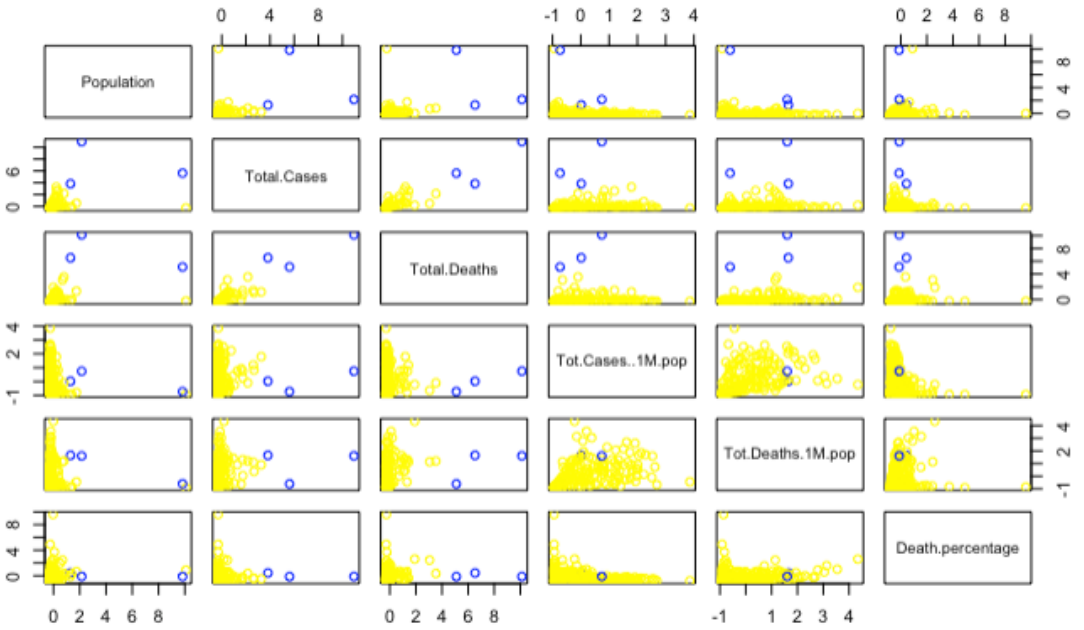


Figure 107: Scatter plot K-means

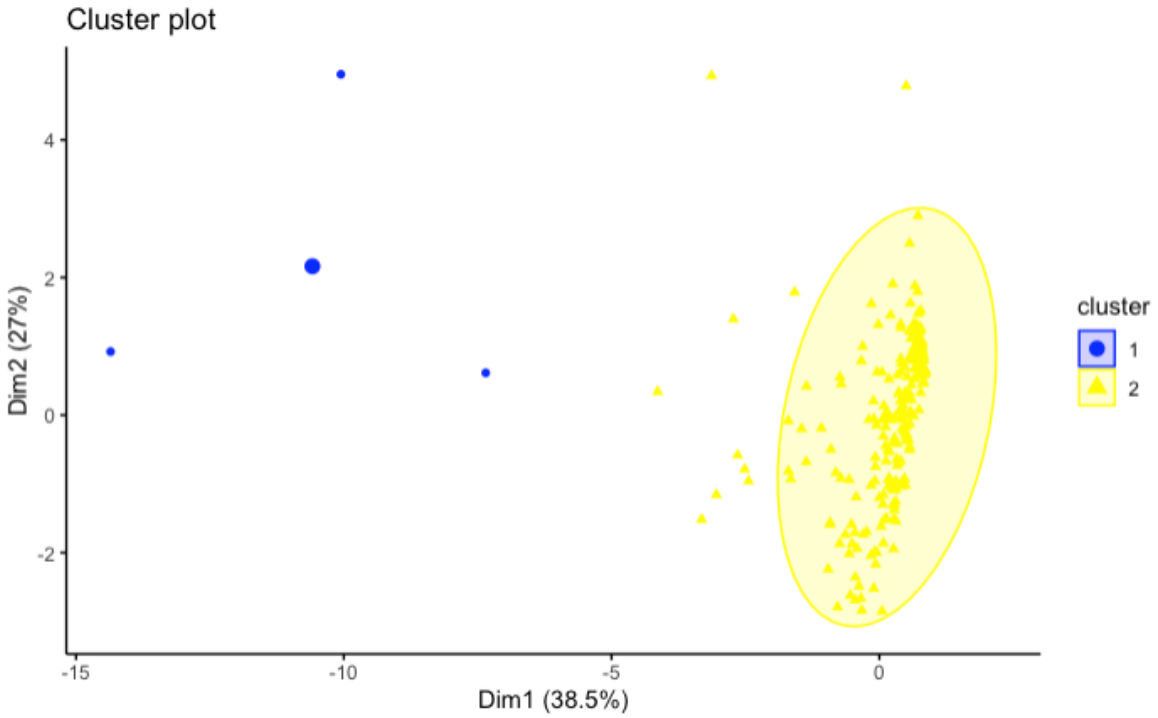


Figure 108: PCs space K-means

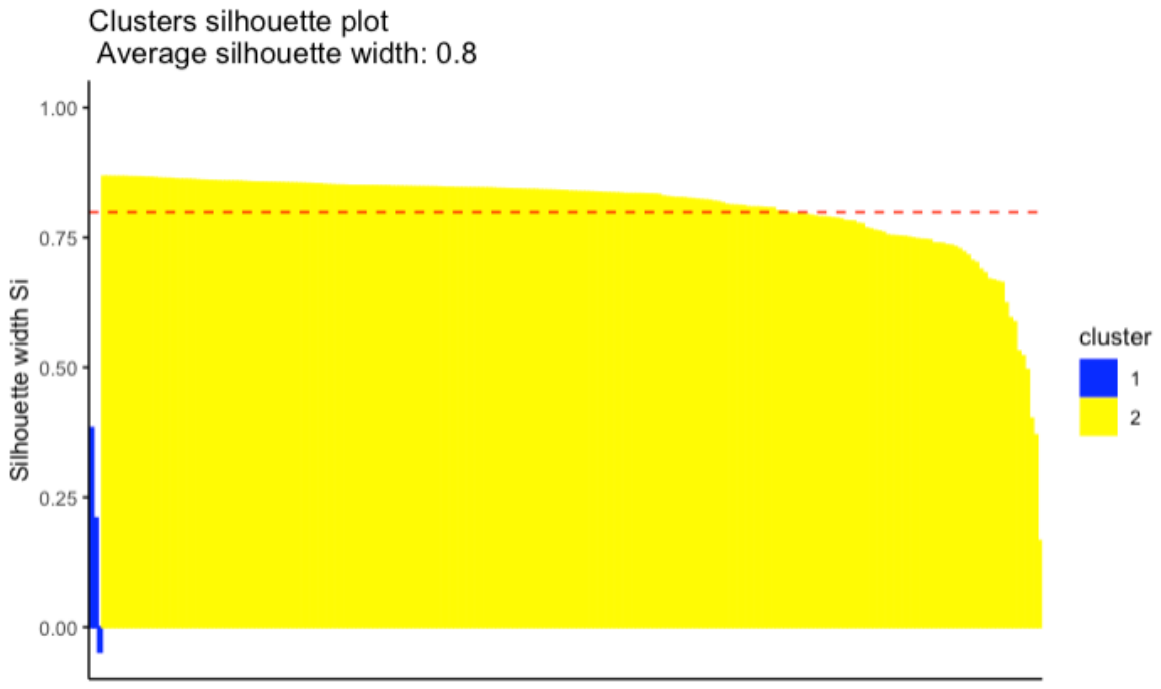


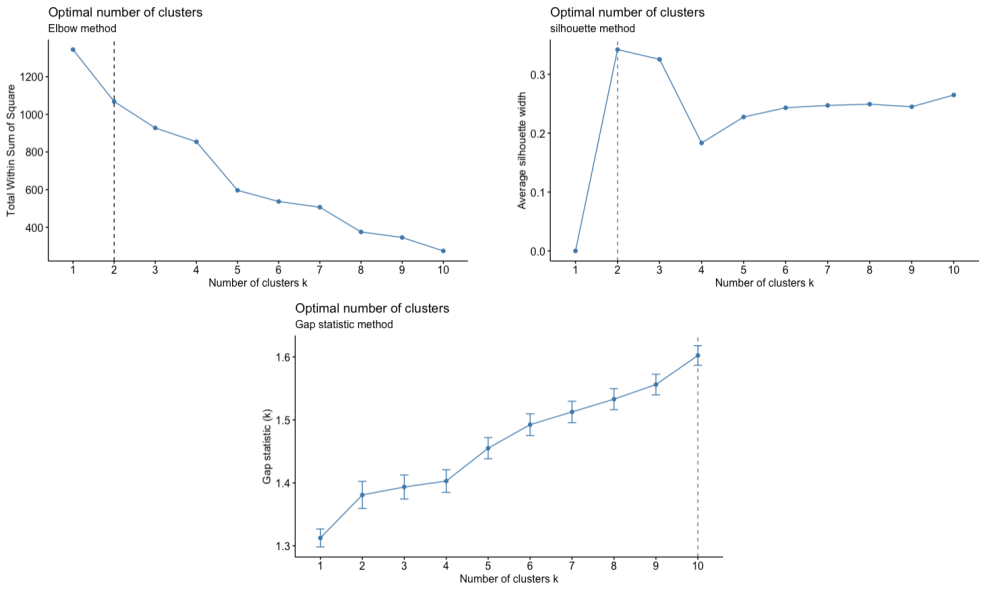
Figure 109: average silhouette

5.4.2 K-Medoids

```
fviz_nbclust(df,cluster::pam, method = "wss")+
  labs(subtitle = "Elbow method")+
  geom_vline(xintercept = 2, linetype=2)
fviz_nbclust(df, cluster::pam, method="silhouette")+
  labs(subtitle ="silhouette method")
fviz_nbclust(df, cluster::pam, method="gap_stat",nboot=500)+
  labs(subtitle = "Gap statistic method")


pam.res<-pam(df, 2)
print(pam.res)
pam.res$clusinfo
cm<-pam.res$clustering
pairs(df,pch=21,
col=c("blue","yellow")[cm])
fviz_cluster(pam.res, data=df,
geom="point",
ellipse.type = "norm", palette=c("blue","yellow"),
ggtheme = theme_classic())
```

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

```
km2<- eclust(df, "pam", k=2, graph=FALSE)
```



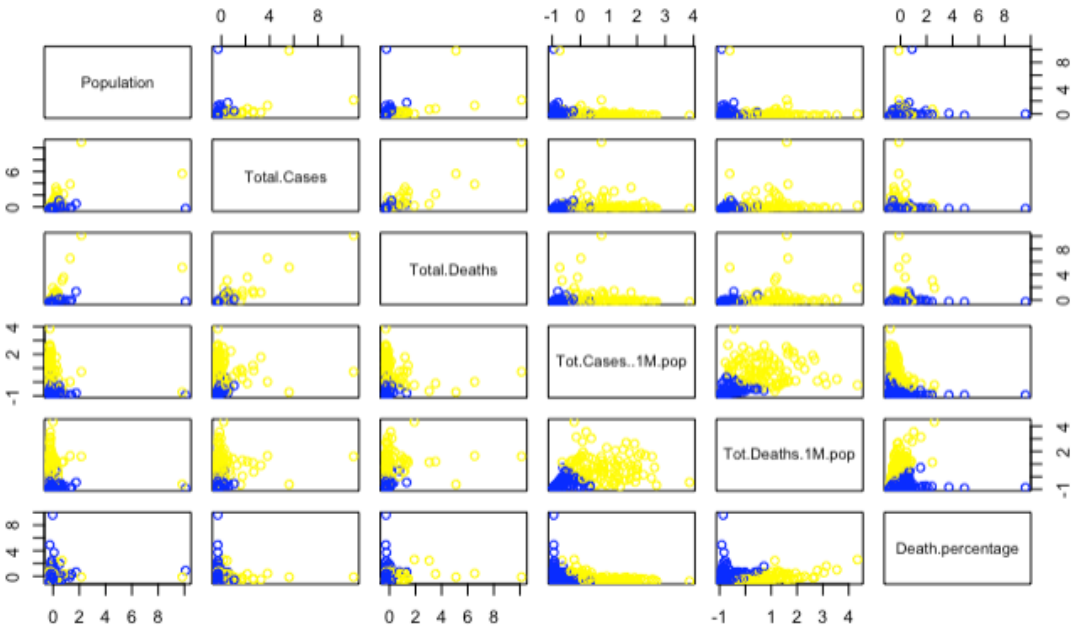


Figure 111: scatter plot

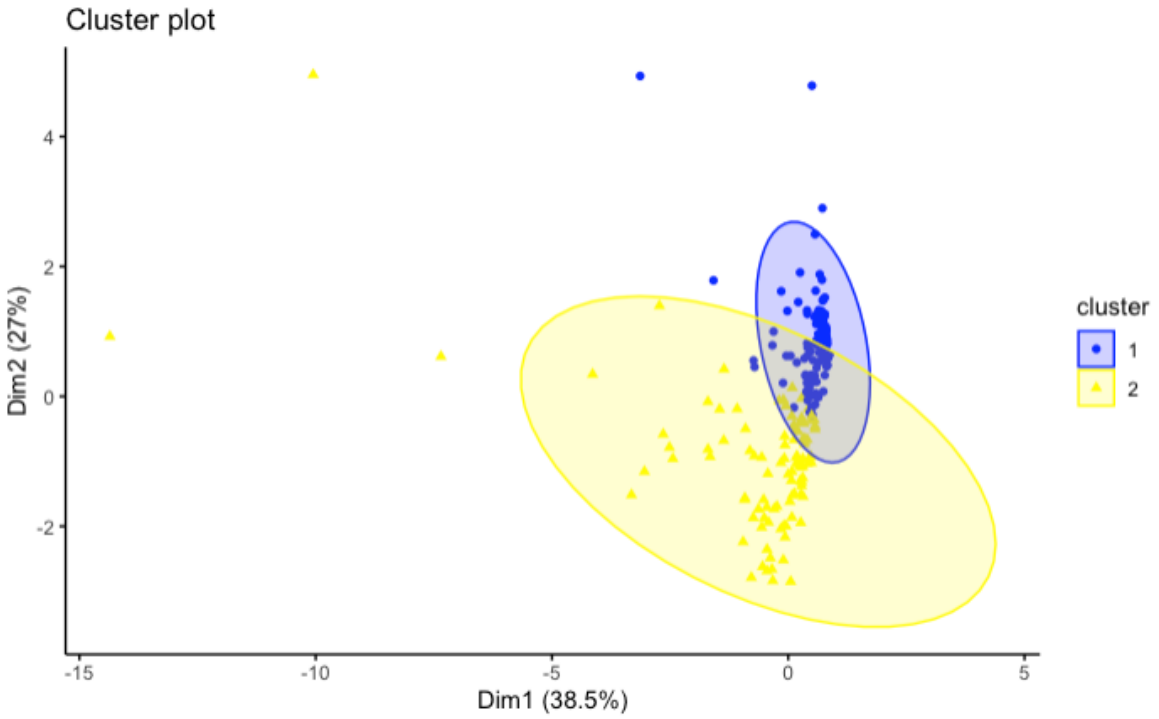


Figure 112: PCs space K-medoids

```
Medoids:
      ID Population Total.Cases Total.Deaths Tot.Cases..1M.pop Tot.Deaths.1M.pop Death.percentage
[1,] 224 -0.1133907 -0.25669618  -0.24234390        -0.8304018        -0.7449226       -0.1108125
[2,] 196 -0.1785649  0.04165381  -0.09409601         0.7361049         0.5839888       -0.4060935
Clustering vector:
  [1] 1 1 1 2 1 2 1 2 2 2 1 2 1 2 2 1 2 1 2 2 1 2 1 2 2 2 2 2 2 2 1 1 1 1 1 1 2 2 1 2 2 1 2 1 1
1 2 2 1 2 2 2 2 1 2 1
 [58] 1 1 1 1 1 1 2 1 1 2 1 1 1 2 2 2 1 1 2 2 1 2 2 2 2 2 2 1 1 1 1 1 1 2 2 2 2 1 2 1 2 2 2 2 1 1 1
2 1 1 1 1 1 1 2 2 1 1
[115] 1 2 2 2 1 1 1 2 2 1 2 1 2 1 1 1 2 1 2 2 1 2 1 1 1 1 1 1 2 2 1 1 1 1 1 2 2 1 1 2 1 2 1 2 2 1
2 2 1 2 2 2 1 2 1 1 2
[172] 2 2 1 2 1 1 1 2 2 1 1 2 2 2 1 1 1 1 2 1 2 1 1 2 2 2 1 1 1 1 1 1 1 1 2 2 2 2 1 1 2 2 2 2 1 1
1 1 1 1 1 1 1 1
Objective function:
   build     swap
1.447461 1.367963

Available components:
 [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"  "clusinfo"   "silinfo"
"diss"
 [9] "call"       "data"
     size max_diss  av_diss diameter separation
[1,]  123 10.24958 0.993978 13.34202  0.1472077
[2,]  102 15.16110 1.818944 15.96080  0.1472077
```

Figure 113: Medoids

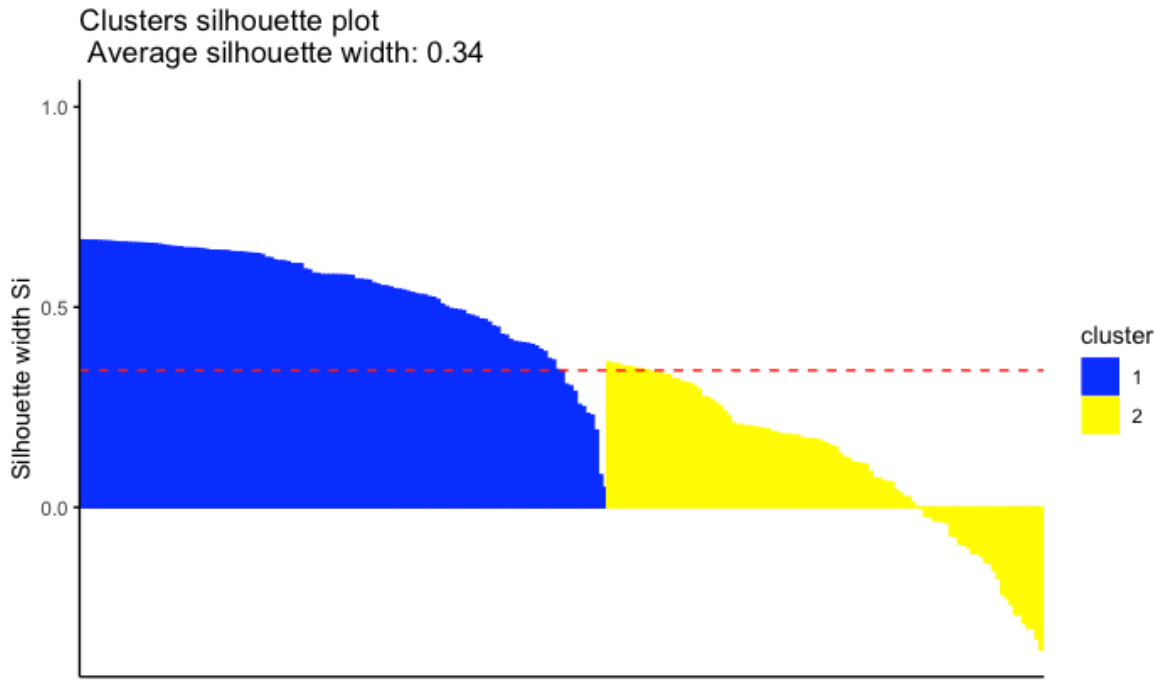


Figure 114: average silhouette

```
fviz_silhouette(km2, palette=c("blue","yellow"), ggtheme=theme_classic())
km2$silinfo$clus.avg.widths
km2$silinfo$avg.width
km_stats2<-cluster.stats(dist(df),km2$cluster)
km_stats2$dunn
[1] 0.5384132 0.1052194
[1] 0.342032
[1] 0.009223076
```

5.5 BEST CLUSTERING ALGORITHM

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

```
intern <- clValid(df, nClust = 2:6,
clMethods = c("hierarchical", "kmeans", "pam"),
validation = c("internal", "stability"), metric = "euclidean")
print(intern)
summary(intern)

Warning: rownames for data not specified, using 1:nrow(data)
Call:
clValid(obj = df, nClust = 2:6, clMethods =
c("hierarchical",
    "kmeans", "pam"), validation = c("internal", "stability"),
```

```
    metric = "euclidean")

Clustering Methods:
 hierarchical kmeans pam

Cluster sizes:
 2 3 4 5 6

Validation measures:
 APN AD ADM FOM Connectivity Dunn Silhouette


Clustering Methods:
 hierarchical kmeans pam

Cluster sizes:
 2 3 4 5 6

Validation Measures:
                              2       3       4       5       6

hierarchical APN            0.0088  0.0066  0.0052  0.0132  0.0171
             AD             2.4848  2.3313  2.2535  2.2271  2.1586
             ADM            0.1107  0.0797  0.0682  0.1128  0.1444
             FOM            0.9276  0.8655  0.8633  0.8489  0.8340
             Connectivity   3.0956  6.7869  9.7159 11.7159 17.8694
             Dunn           0.5052  0.6467  0.4849  0.4849  0.3355
             Silhouette     0.8318  0.7924  0.7572  0.7543  0.6166
kmeans       APN            0.0054  0.0059  0.0903  0.3011  0.1171
             AD             2.3873  2.2988  2.2152  2.1928  1.8046
             ADM            0.0736  0.0651  0.2693  0.7321  0.5299
             FOM            0.9026  0.8704  0.8821  0.8530  0.8274
             Connectivity   7.6202  8.7159 14.1456 16.1456 38.0460
             Dunn           0.2621  0.3057  0.1638  0.1638  0.0330
             Silhouette     0.7987  0.7919  0.6594  0.6567  0.4021
pam          APN            0.0921  0.2966  0.1964  0.2924  0.1983
             AD             2.1307  2.0673  1.8435  1.7855  1.5929
             ADM            0.2894  0.7252  0.4459  0.7134  0.5451
             FOM            0.9551  0.9308  0.9064  0.8908  0.8374
             Connectivity  36.4881 47.4528 54.5865 58.4567 73.8996
             Dunn           0.0092  0.0079  0.0112  0.0099  0.0086
             Silhouette     0.3420  0.3254  0.1833  0.2276  0.2432

Optimal Scores:
    Score        Method               Clusters

APN 0.0052    hierarchical            4
AD 1.5929    pam                    6
ADM 0.0651    kmeans                 3
FOM 0.8274    kmeans                 6
Connectivity 3.0956 hierarchical 2
```

```
Dunn 0.6467 hierarchical         3
Silhouette 0.8318 hierarchical     2
```

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

```
intern1<-clValid(df, nClust = 2:6,
clMethods = c("Hierarchical","kMeans","PAM"),
validation = c("internal","stability"),
metric = "manhattan")
summary(intern1)
Clustering Methods:
 hierarchical kmeans pam

Cluster sizes:
 2 3 4 5 6

Validation Measures:           2     3        4      5        6


hierarchical APN            0.0066  0.0052  0.0096  0.0161  0.0218
             AD             4.1393  3.9489  3.8779  3.8035  3.7084
             ADM            0.1069  0.1131  0.1264  0.1370  0.1312
             FOM            0.8805  0.8627  0.8546  0.8475  0.8297
             Connectivity   5.4690  9.8270 10.8270 11.8270 14.7560
             Dunn           0.1853  0.2920  0.2920  0.2920  0.3216
             Silhouette     0.7994  0.7782  0.7084  0.6862  0.6539
kmeans       APN            0.0054  0.0123  0.1677  0.0481  0.1171
             AD             4.0225  3.8966  3.7976  3.2773  3.0987
             ADM            0.0736  0.0815  0.4566  0.5776  0.5299
             FOM            0.9026  0.8842  0.8796  0.8530  0.8274
             Connectivity   7.2552  9.8270 15.7389 40.9575 41.9575
             Dunn           0.2669  0.2920  0.1048  0.0319  0.0319
             Silhouette     0.8108  0.7782  0.5570  0.3850  0.3837
pam          APN            0.0895  0.2676  0.1954  0.2836  0.2347
             AD             3.5730  3.4170  3.0736  2.9300  2.7701
             ADM            0.2545  0.6385  0.5771  0.6358  0.5962
             FOM            0.9580  0.9433  0.9187  0.9099  0.8840
             Connectivity  34.4849 49.3901 57.1655 73.9623 90.0683
             Dunn           0.0041  0.0092  0.0100  0.0127  0.0031
             Silhouette     0.3502  0.3547  0.2754  0.2783  0.2498

Optimal Scores:

APN           0.0052 hierarchical 3
AD           2.7701 pam 6
ADM           0.0736 kmeans 2
FOM           0.8274 kmeans 6
Connectivity  5.4690 hierarchical 2
Dunn       0.3216 hierarchical 6
```

```
Silhouette   0.8108 kmeans 2
```

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

5.5.1 MODEL BASED CLUSTERING

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

```
mod<-Mclust(df,G=1:9, modelname=NULL)
summary(mod$BIC)
```

```
Best BIC values:
            VVV,7     VVV,8    VVV,9
BIC      956.9364 915.57246 905.7721
BIC diff   0.0000 -41.36393 -51.1643
```

According with the output, we can see that the maximized value of BIC (Bayesian Information Criterion) is at the parsimonious model VVV with 9 clusters. The BIC diff. indicates the difference between the first and the second model. Now we can plot the graphical counterpart of the previous code.

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

As we can see from the image above, we have different curve for each number of clusters and different symbols for all the parsimonious configurations. The highest point is for the VVV model with 9 clusters.

```
summary(mod)
head(round(mod$z,6),40)
```

```
         [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
 [1,] 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000
 [2,] 0.000005 0.999995 0.000000 0.000000 0.000000 0.000000 0.000000
 [3,] 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000
 [4,] 0.000000 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000
 [5,] 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000
 [6,] 0.000000 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000
 [7,] 0.000009 0.999991 0.000000 0.000000 0.000000 0.000000 0.000000
 [8,] 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
 [9,] 0.000021 0.999979 0.000000 0.000000 0.000000 0.000000 0.000000
[10,] 0.000000 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000
[11,] 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
[12,] 0.000000 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000
[13,] 0.000128 0.999872 0.000000 0.000000 0.000000 0.000000 0.000000
[14,] 0.000011 0.999989 0.000000 0.000000 0.000000 0.000000 0.000000
[15,] 0.000000 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000
[16,] 0.999739 0.000000 0.000000 0.000000 0.000000 0.000000 0.000261
```

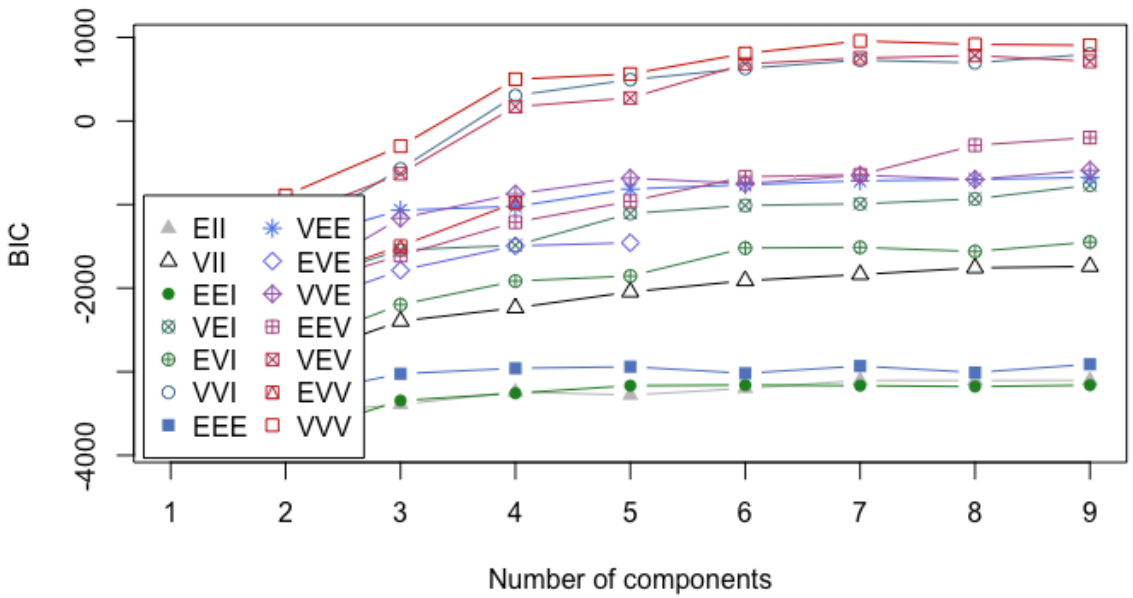


Figure 115

```
[17,] 0.000000 0.000000 0.000000 0.975201 0.024799 0.000000 0.000000
[18,] 0.000010 0.999986 0.000000 0.000000 0.000004 0.000000 0.000000
[19,] 0.000000 0.000000 0.000000 0.000000 1.000000 0.000000 0.000000
[20,] 0.000021 0.998438 0.000000 0.000000 0.001542 0.000000 0.000000
[21,] 0.000000 0.000000 0.000224 0.000000 0.000000 0.999776 0.000000
[22,] 0.000000 0.000000 0.000000 0.000659 0.999341 0.000000 0.000000
[23,] 0.000000 0.000000 0.000000 0.000007 0.000000 0.999993 0.000000
[24,] 0.015917 0.000000 0.984079 0.000000 0.000000 0.000000 0.000004
[25,] 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000
[26,] 0.000002 0.916668 0.000000 0.000000 0.083330 0.000000 0.000000
[27,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 1.000000
[28,] 0.000000 0.000000 0.000000 0.000712 0.999288 0.000000 0.000000
[29,] 0.000000 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000
[30,] 0.999999 0.000000 0.000000 0.000000 0.000000 0.000000 0.000001
[31,] 0.000000 0.000000 0.999986 0.000000 0.000000 0.000014 0.000000
[32,] 0.000000 0.000000 0.001319 0.000000 0.000000 0.998681 0.000000
[33,] 0.000006 0.998409 0.000000 0.001585 0.000000 0.000000 0.000000
[34,] 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000
[35,] 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000
[36,] 0.999999 0.000000 0.000000 0.000000 0.000000 0.000000 0.000001
[37,] 0.000000 0.000002 0.000007 0.000000 0.000000 0.999991 0.000000
[38,] 0.000000 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000
[39,] 0.000000 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000
[40,] 0.000000 0.000000 0.999735 0.000000 0.000000 0.000265 0.000000
```

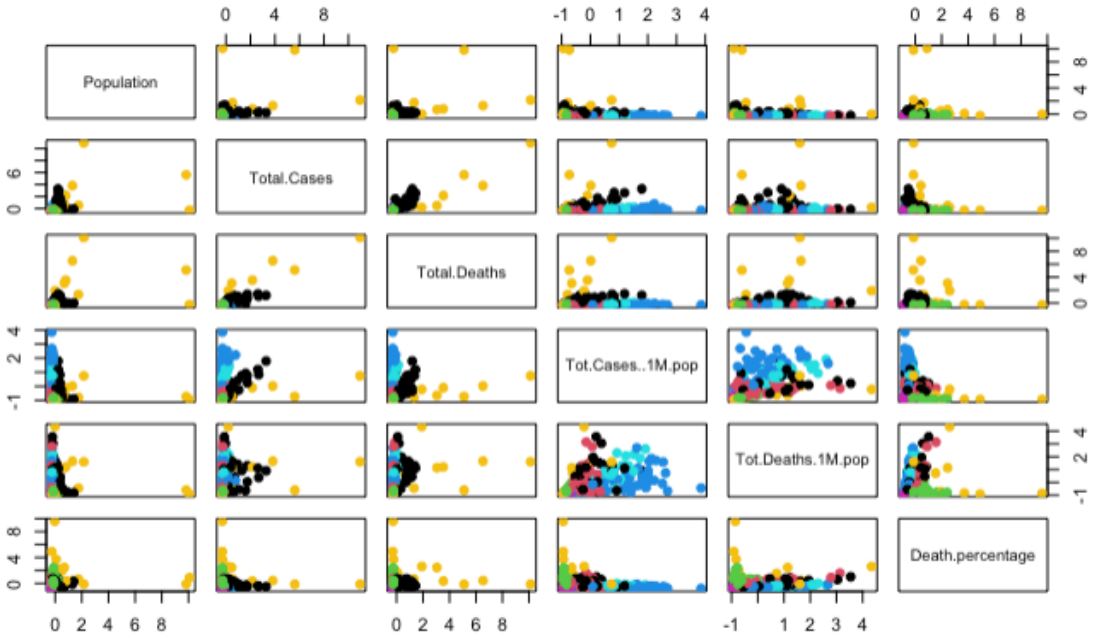


Figure 116

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

```
head(mod$classification,30)
 [1] 3 2 3 4 3 4 2 1 2 4 1 4 2 2 4 1 4 2 5 2 6 5 6 3 2 2 7 5 4 1
```

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

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

```
fviz_mclust(mod,"classification",geom="point", pointsize=1, palette="jco")
```

As we can see, there are 7 clusters carried out by the VVV model in the PCs space.

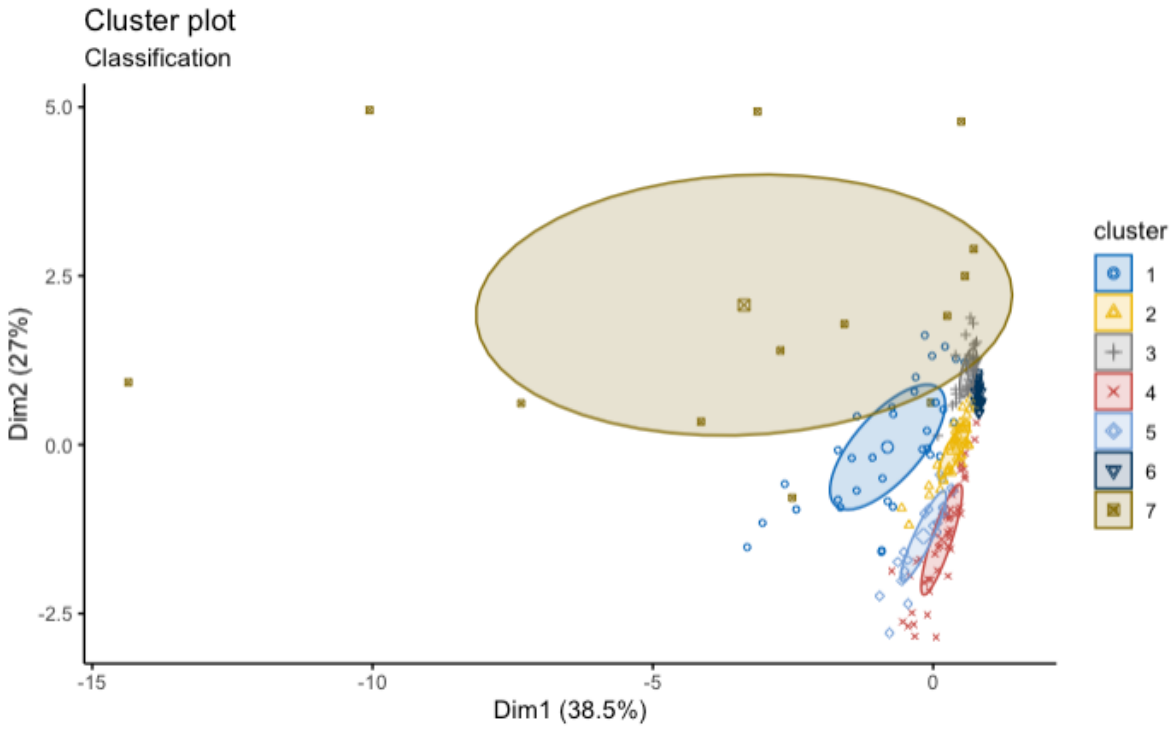

Cluster plot
Classification
5.0
2.5
0.0
-2.5
Dim2 (27%)
-15
-10
-5
0
Dim1 (38.5%)
cluster
1
2
3
4
5
6
7

Figure 117