Report DATA ANALYSIS: Insurance Claim Analysis

Rosario Licciardello UNICT - 1000069335 Prof. Antonio Punzo

Report DATA ANALYSIS - Insurance Claim Analysis: Demographic and Health

Rosario Licciardello

2024-06-25

Contents

0. Introduction to the report

This study utilizes a comprehensive dataset encompassing demographic and health information derived from insurance claims, available on Kaggle. The dataset includes variables such as age, gender, BMI, blood pressure, presence of chronic diseases like diabetes, smoking status, region, and insurance claim amounts. These variables provide a rich basis for examining the intricate relationships between health indicators and insurance claims.

The primary objective is to employ the statistical programming language R to conduct a thorough exploration of the dataset. The analysis comprises:

- **Univariate Analysis**: This step involves examining the distribution and key statistics of individual variables to understand their characteristics.
- **Multivariate Analysis**: This stage focuses on exploring the relationships between multiple variables simultaneously to uncover potential interactions and correlations.
- **Principal Component Analysis (PCA)**: PCA is employed to reduce the dimensionality of the dataset, facilitating the identification of principal components that explain the maximum variance.
- **Clustering Analysis**: This method is used to identify natural groupings within the data, thereby uncovering hidden patterns and insights.

1. Data Cleaning and Preparation

The initial steps involve cleaning the insurance dataset to prepare it for analysis. We set the `PatientID` as row names, remove unnecessary columns such as `Index`, handle missing values, and ensure data integrity by filtering out rows with empty `region` values. Furthermore, a function `num_bins` is defined to automatically calculate the number of bins for histograms using Scott's rule.

```r
#defining the list of the libraries
lib <- c(
  "moments", "gamlss", "psych", "corrplot", "ggplot2", "factoextra",
  "hopkins", "NbClust", "stats", "fpc", "cluster", "clValid", "mclust",
  "nnet", "here", "dplyr", "GGally", "ggcorrplot", "car", "smotefamily",
  "skimr", "Metrics", "patchwork", "stringr", "gamlss.dist","factoextra","caret",
"seriation","fclust"
) #defining the list of the package we are using

#loading all the lybraries

lapply(lib, library, character.only = TRUE)

# Importing the dataset using the here package
insurance <- read.csv(here::here("data/processed", "insurance_data.csv"))
```

```
# Assigning the dataset to a variable
dataset <- insurance

# Setting PatientID as row names
rownames(dataset) <- dataset[, 2]
dataset <- dataset[, -1] # Removing PatientID column
dataset <- dataset[, -1] # Removing Index column

# Removing rows with missing values and empty region values
dataset <- na.omit(dataset)
dataset <- dataset[dataset$region != "", ]

# Function to calculate the number of bins for histograms
num_bins <- function(dataset, variable) {
  n <- length(dataset[[variable]])
  bin_width <- 3.5 * sd(dataset[[variable]]) / (n^(1/3))
  num_bins <- (max(dataset[[variable]]) - min(dataset[[variable]])) / bin_width
  return(as.integer(num_bins))
}

name<-names(dataset) #this is needed to select the name of all the variables
```

2. Univariate Analysis

In the univariate analysis, we will examine each variable in the insurance dataset. This involves computing descriptive statistics, visualizing distributions, and identifying suitable statistical models that best fit the data. Our objective is to gain a comprehensive understanding of each variable's characteristics and distributions independently.

To provide a quick overview of the dataset, we utilize the `skimr` package to generate summary statistics for all variables. This includes measures such as mean, standard deviation, minimum, maximum, and quantiles, helping us to grasp the central tendencies and spread of our dataset efficiently.

```
skim(dataset) # producing a quick analysis of the dataset
```

Data summary

Name	dataset
Number of rows	1332
Number of columns	9
________________________	
Column type frequency:	
character	4
numeric	5
________________________	

Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
gender	0	1	4	6	0	2	0
diabetic	0	1	2	3	0	2	0
smoker	0	1	2	3	0	2	0
region	0	1	9	9	0	4	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
age	0	1	38.09	11.11	18.00	29.00	38.00	47.00	60.00	▆▇▇▇▅
bmi	0	1	30.66	6.12	16.00	26.20	30.35	34.73	53.10	▁▇▇▂▁
bloodpressure	0	1	94.19	11.45	80.00	86.00	92.00	99.00	140.00	▇▆▂▁▁
children	0	1	1.10	1.21	0.00	0.00	1.00	2.00	5.00	▇▂▂▁▁
claim	0	1	13325.25	12109.62	1121.87	4760.16	9412.97	16781.33	63770.43	▇▂▁▁▁

This is a preliminary introduction to the variables:

- **Age**: The `age` variable represents the age of individuals in the insurance dataset. It is a continuous numerical variable that indicates the age of each insured person.
- **Gender**: The `gender` variable indicates the gender of individuals in the dataset. It is a categorical variable with two levels (typically "Male" and "Female"), representing the gender identity of each insured person.
- **BMI (Body Mass Index)**: The `bmi` variable represents the Body Mass Index (BMI) of individuals. It is a continuous numerical variable that quantifies the ratio of weight to height squared, providing an indicator of body fatness and health status.
- **Blood Pressure**: The `bloodpressure` variable represents the blood pressure readings of individuals. It is likely to consist of two numerical values, systolic and diastolic pressure, providing insights into cardiovascular health.
- **Diabetic**: The `diabetic` variable indicates whether individuals have been diagnosed with diabetes. It is a binary categorical variable with levels "Yes" or "No," reflecting the presence or absence of diabetes.

- **Children**: The `children` variable indicates the number of children each insured person has. It is a discrete numerical variable that represents family size or dependents.
- **Smoker**: The `smoker` variable indicates whether individuals are smokers. It is a binary categorical variable with levels “Yes” or “No,” indicating smoking status.
- **Region**: The `region` variable represents the geographic region where individuals reside. It is a categorical variable that categorizes individuals based on their location, providing insights into regional differences in insurance claims and health outcomes.
- **Claim**: The `claim` variable represents the insurance claim amounts filed by individuals. It is a continuous numerical variable that quantifies the financial value of insurance claims made by each insured person.

To assess the goodness of fit of various theoretical distributions to the observed data in the insurance dataset, we will employ two primary criteria: the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC).

- **Akaike Information Criterion (AIC)**:
  - Formula: $\text{AIC} = -2 \times \log(\text{likelihood}) + 2 \times \text{number of parameters}$
- **Bayesian Information Criterion (BIC)**:
  - Formula: $\text{BIC} = -2 \times \log(\text{likelihood}) + \text{number of parameters} \times \log(\text{sample size})$

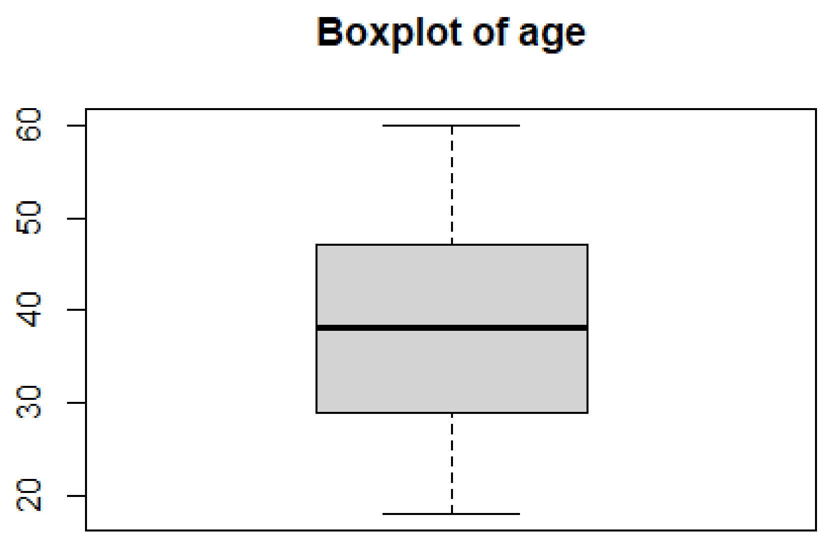
The model with the lowest AIC and BIC values across different distributions is selected as the best-fit model for each variable, providing insights into the underlying data distribution while balancing model complexity.

Age Variable

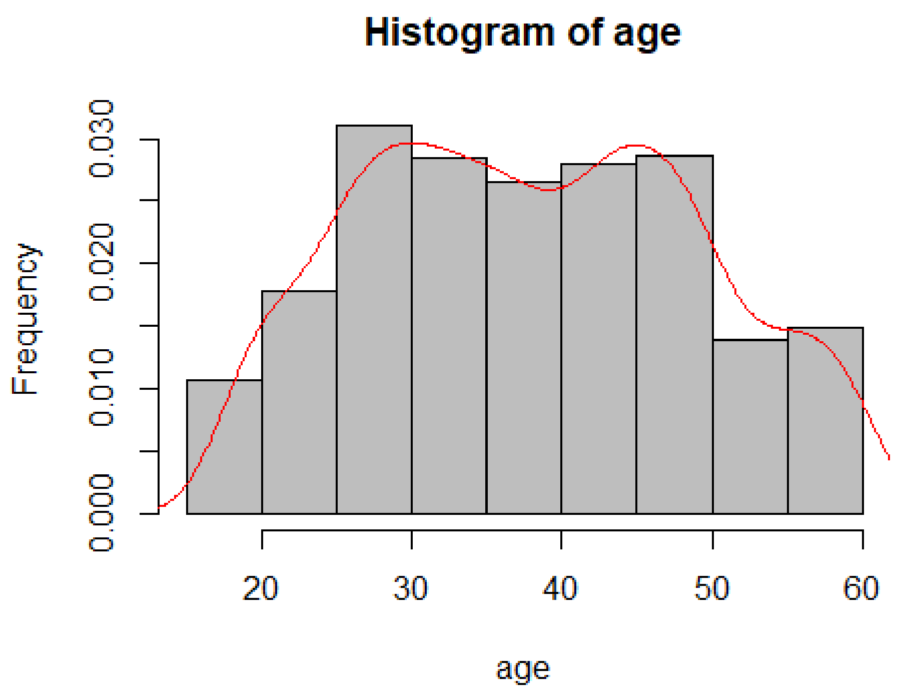
The age variable is analyzed to understand its distribution and key statistics:

```
# Selecting the 'age' variable
variable <- "age"
print(variable)

## [1] "age"

# Summary statistics
summary_age <- summary(dataset[, variable])
print(summary_age)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.00   29.00   38.00   38.09   47.00   60.00

# Mean, variance, and standard deviation
mean_age <- mean(dataset[, variable])
var_age <- var(dataset[, variable])
sd_age <- sd(dataset[, variable])
print(paste("Mean:", mean_age))

## [1] "Mean: 38.0863363363363"
```

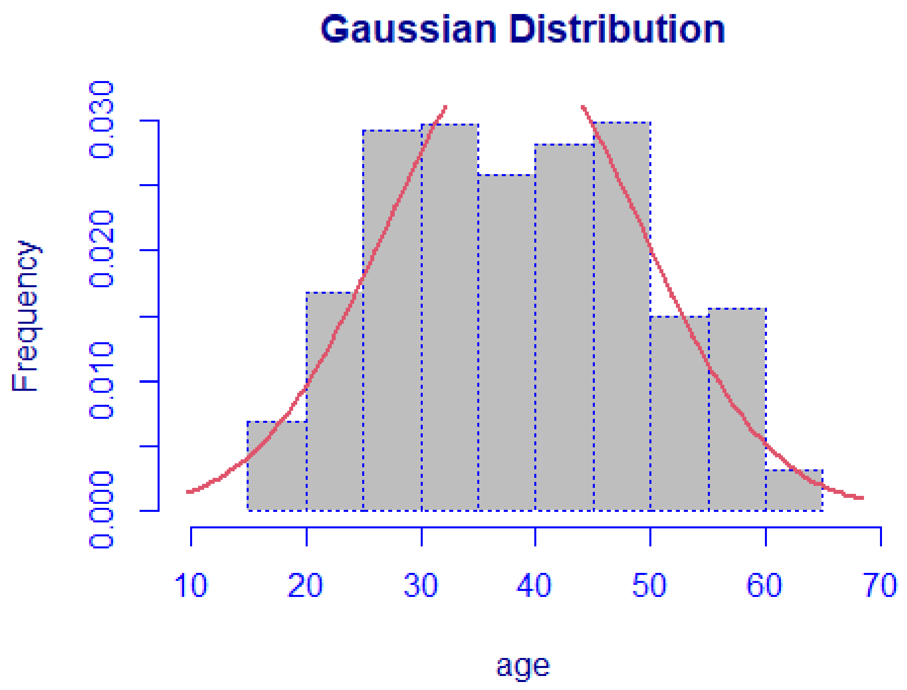
```r
print(paste("Variance:", var_age))

## [1] "Variance: 123.494418723758"

print(paste("Standard Deviation:", sd_age))

## [1] "Standard Deviation: 11.1128042691194"

# Skewness and Kurtosis
skewness_age <- e1071::skewness(dataset[, variable])

## Registered S3 methods overwritten by 'proxy':
##   method               from
##   print.registry_field registry
##   print.registry_entry registry

## Registered S3 method overwritten by 'e1071':
##   method       from
##   print.fclust fclust

kurtosis_age <- e1071::kurtosis(dataset[, variable])
print(paste("Skewness:", skewness_age))

## [1] "Skewness: 0.111477576683256"

print(paste("Kurtosis:", kurtosis_age))

## [1] "Kurtosis: -0.954169634852498"

# Boxplot
boxplot(dataset[, variable], main = paste("Boxplot of", variable))
```



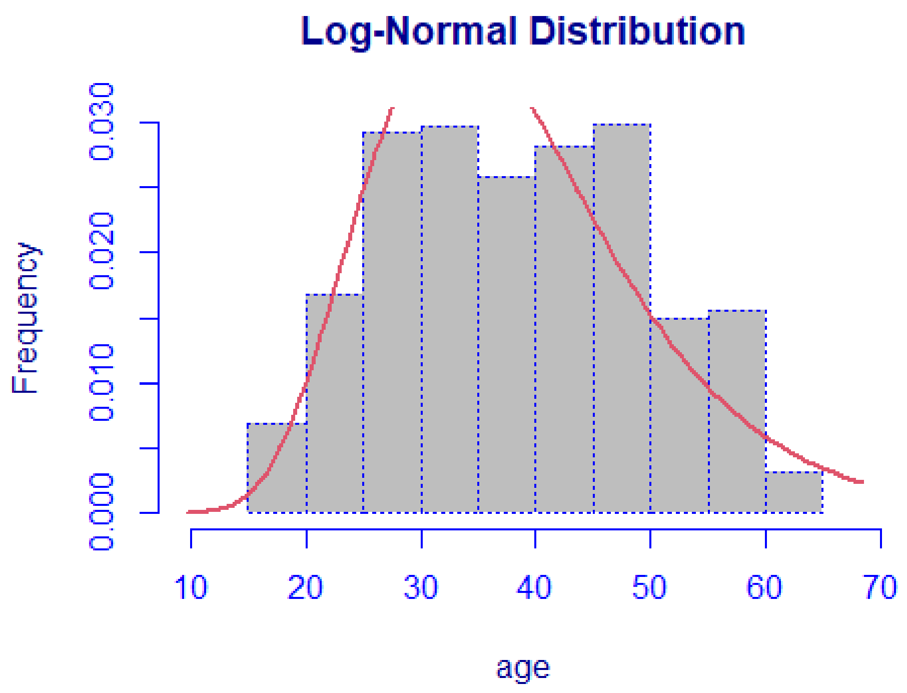
```r
# Histogram with density line
nb <- num_bins(dataset, variable)
histogram_age <- hist(dataset[, variable], main = paste("Histogram of", variable),
xlab = variable, ylab = "Frequency", prob = TRUE, breaks = nb, col = "grey")
lines(density(dataset[, variable]), col = "red")
```



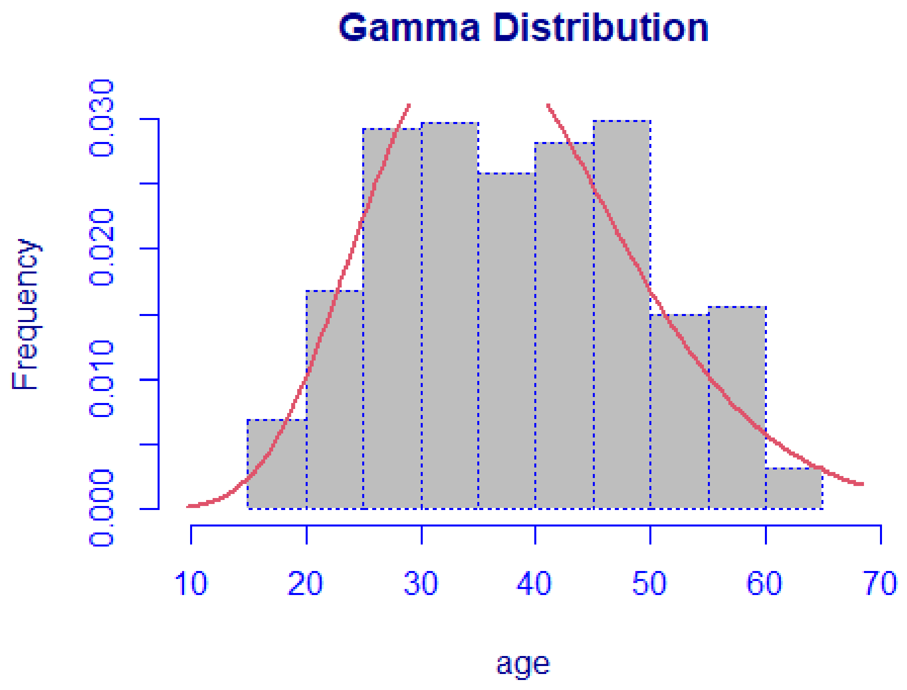
```
# Fit various distributions and calculate AIC
fit_gaussian <- histDist(dataset[, variable], family = NO, nbins = nb, main =
"Gaussian Distribution", xlab = variable, ylab = "Frequency")
```



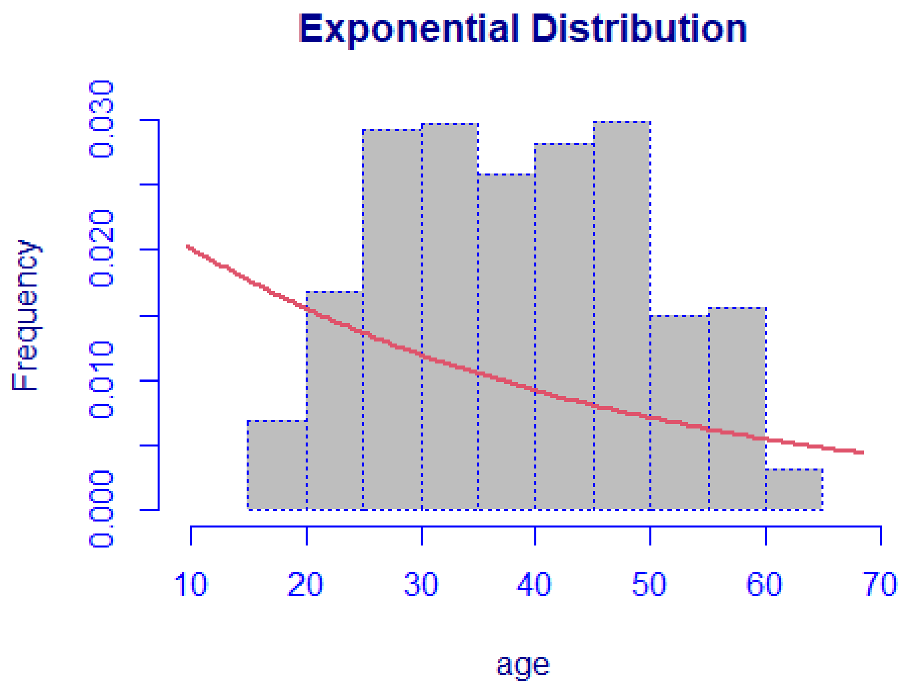
```
fit_logno <- histDist(dataset[, variable], family = LOGNO, nbins = nb, main =
"Log-Normal Distribution", xlab = variable, ylab = "Frequency")
```



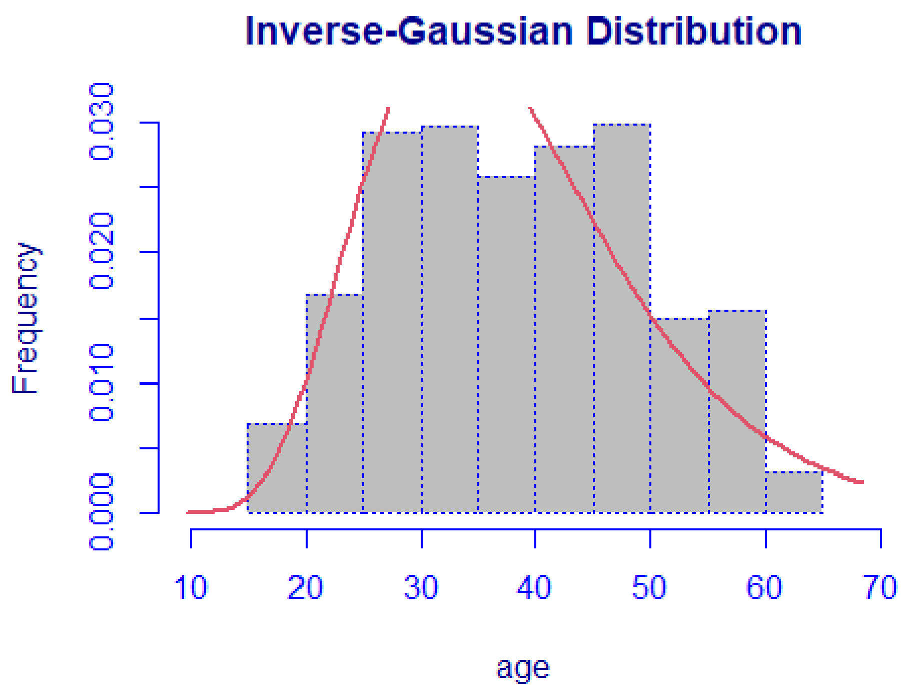
```
fit_gamma <- histDist(dataset[, variable], family = GA, nbins = nb, main = "Gamma
Distribution", xlab = variable, ylab = "Frequency")
```



```
fit_exponential <- histDist(dataset[, variable], family = EXP, nbins = nb, main =
"Exponential Distribution", xlab = variable, ylab = "Frequency")
```



```r
fit_invgauss <- histDist(dataset[, variable], family = IG, nbins = nb, main =
"Inverse-Gaussian Distribution", xlab = variable, ylab = "Frequency")
```

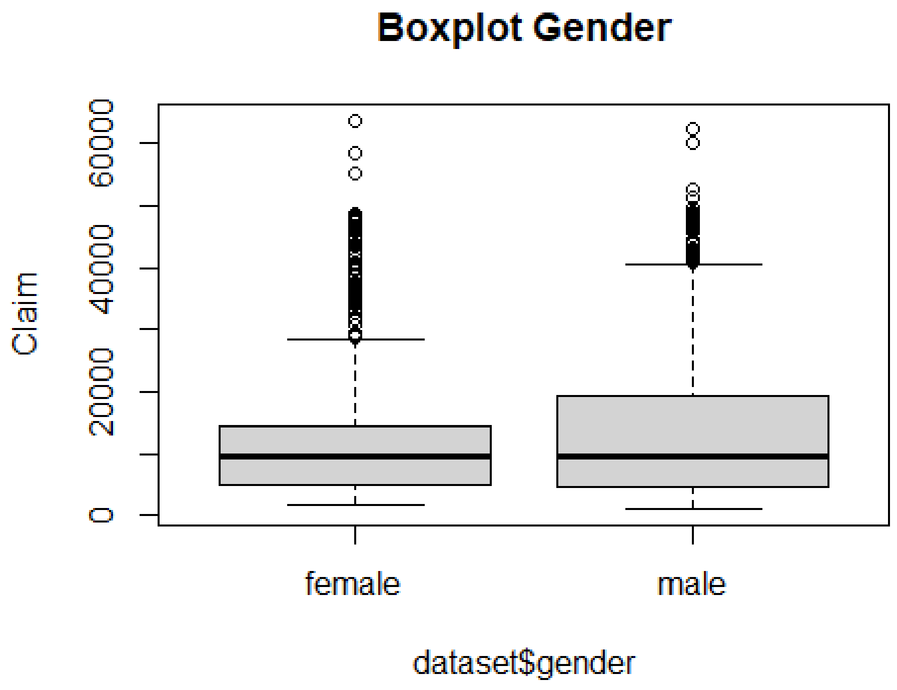


```r
# Calculate AIC for each distribution
AIC_values <- c(AIC(fit_gaussian), AIC(fit_logno), AIC(fit_gamma),
AIC(fit_exponential), AIC(fit_invgauss))
BIC_values <- c(fit_gaussian$sbc, fit_logno$sbc, fit_gamma$sbc,
fit_exponential$sbc, fit_invgauss$sbc)

# Create a data frame to display AIC and BIC values
goodness_of_fit_age <- data.frame(
  Distribution = c("Gaussian", "Log-Normal", "Gamma", "Exponential", "Inverse-
Gaussian"),
  AIC = AIC_values,
  BIC = BIC_values
)

# Display the goodness of fit results
print("Goodness of Fit - Age Variable:")

## [1] "Goodness of Fit - Age Variable:"

print(goodness_of_fit_age)

##       Distribution      AIC      BIC
## 1         Gaussian 10198.22 10208.61
## 2       Log-Normal 10230.88 10241.27
## 3            Gamma 10191.13 10201.52
## 4      Exponential 12362.58 12367.77
## 5 Inverse-Gaussian 10225.93 10236.32
```

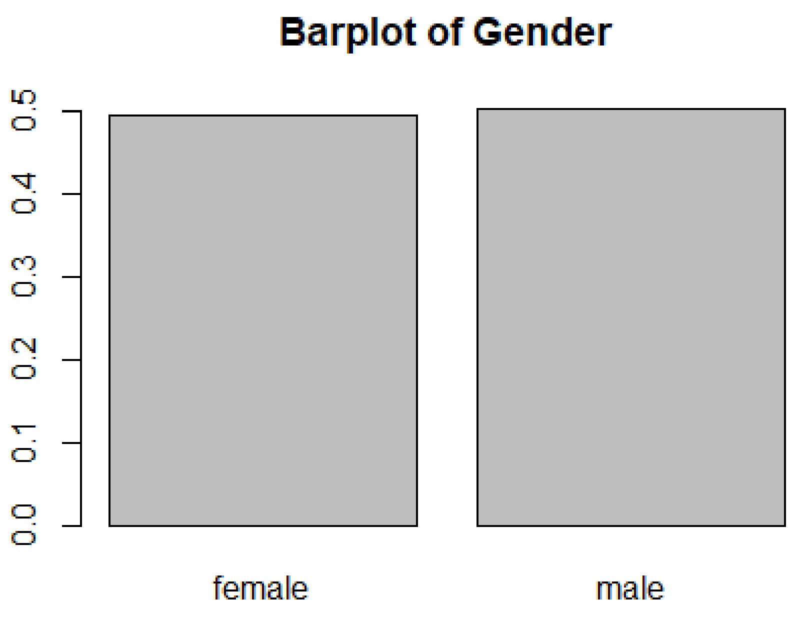
Gender Variable

The gender variable is analyzed to understand its distribution and key statistics:

```
variable<-name[2]
print(variable)

## [1] "gender"

table(select(dataset,variable))

## Warning: Using an external vector in selections was deprecated in tidyselect
1.1.0.
## ℹ Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(variable)
##
##   # Now:
##   data %>% select(all_of(variable))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## gender
## female   male
##    662    670

plot_title <- str_to_title(variable)
boxplot(dataset$clai ~ dataset$gender, main = paste("Boxplot",plot_title), ylab =
"Claim")
```



```
relfreq <- table(dataset[,variable])/length(dataset[,variable])
relfreq

##
##   female     male
## 0.496997 0.503003

barplot(relfreq, main = paste("Barplot of", plot_title))
```

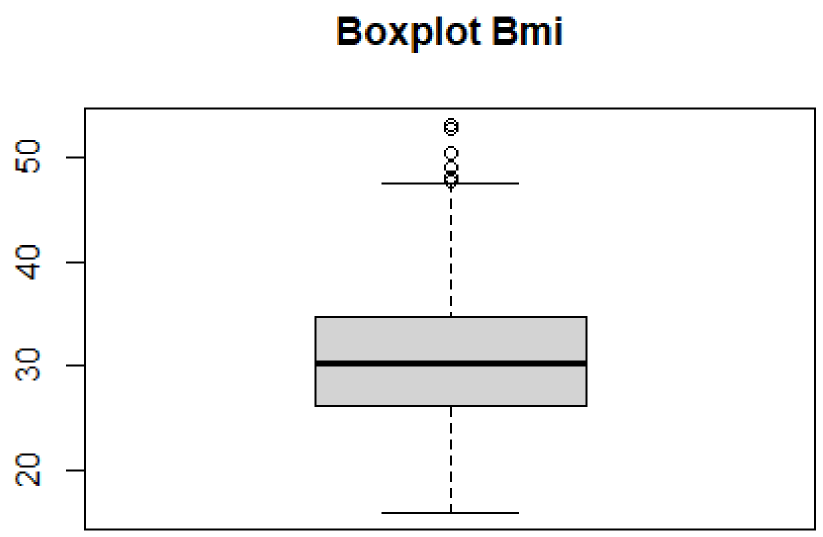


Body Mass Index Variable

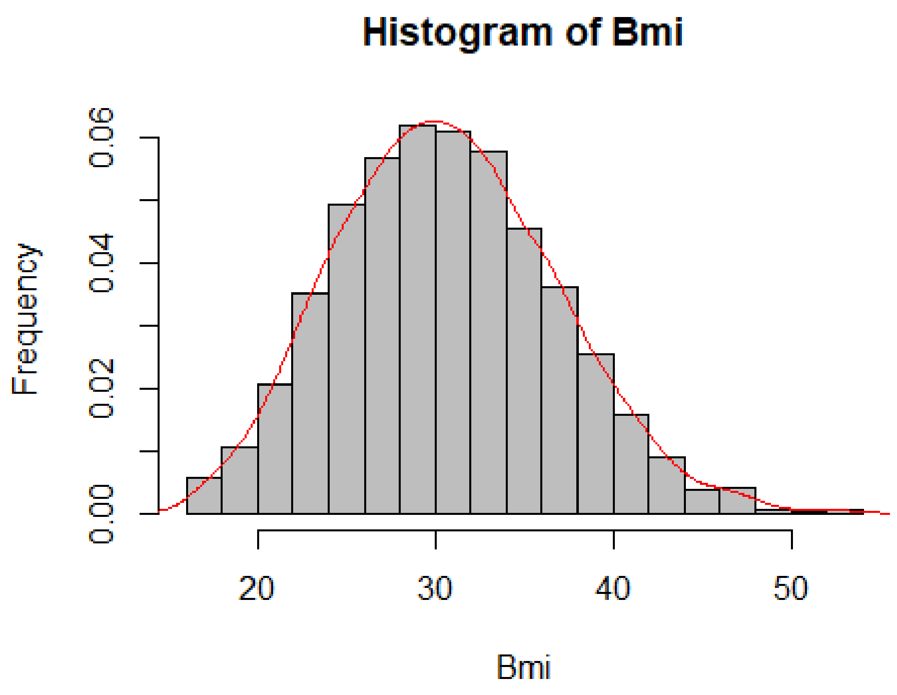
The Body Mass Index variable is analyzed to understand its distribution and key statistics:

```
variable<-name[3]
print(variable)

## [1] "bmi"

summary(select(dataset,variable))

##       bmi
##  Min.   :16.00
##  1st Qu.:26.20
##  Median :30.35
##  Mean   :30.66
##  3rd Qu.:34.73
##  Max.   :53.10

mean(dataset[,variable])

## [1] 30.65833

var(dataset[,variable])

## [1] 37.44176

sd(dataset[,variable])
```

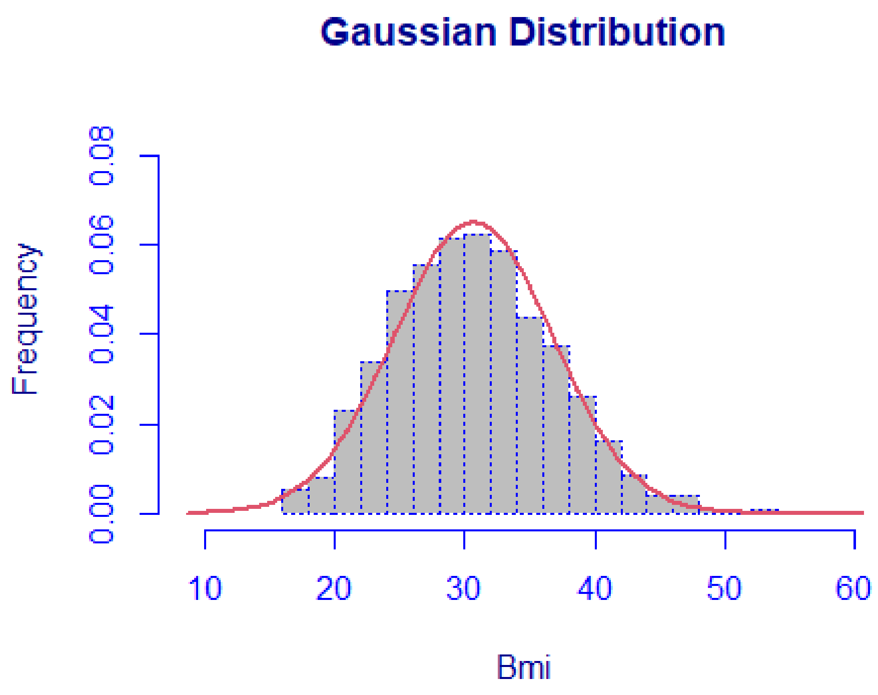
```
## [1] 6.118967

skewness(dataset[,variable])

## [1] 0.2895689

kurtosis(dataset[,variable])

## [1] 2.930315

plot_title <- str_to_title(variable)
boxplot(dataset[,variable], main = paste("Boxplot",plot_title))
```



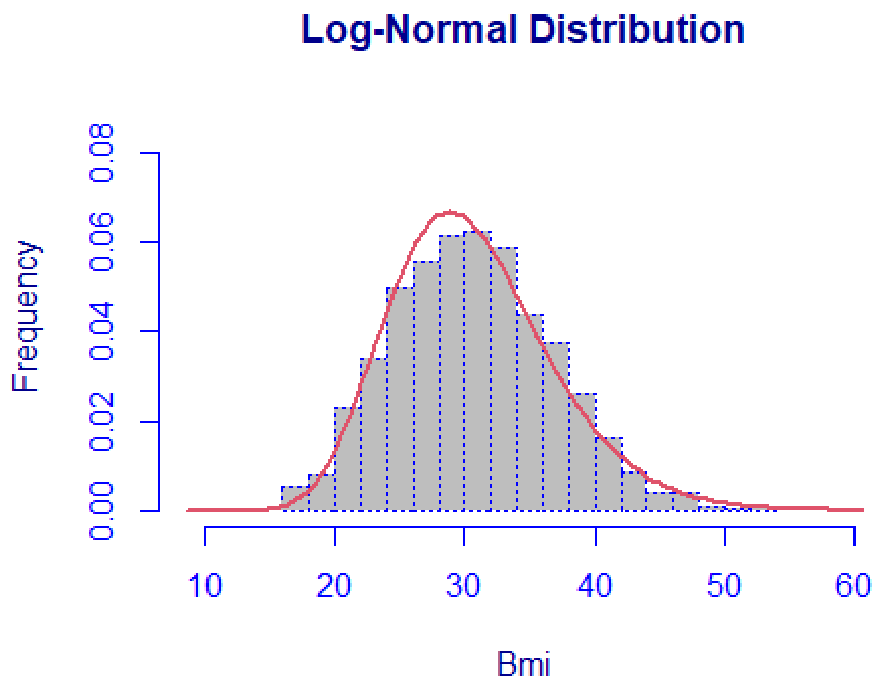
```
nb<-num_bins(dataset,variable)
hist(dataset[,variable], main = paste("Histogram of", plot_title), xlab =
plot_title, ylab = "Frequency", prob = TRUE, breaks = nb, col = "grey")
lines(density(dataset[,variable]), col = "red")
```



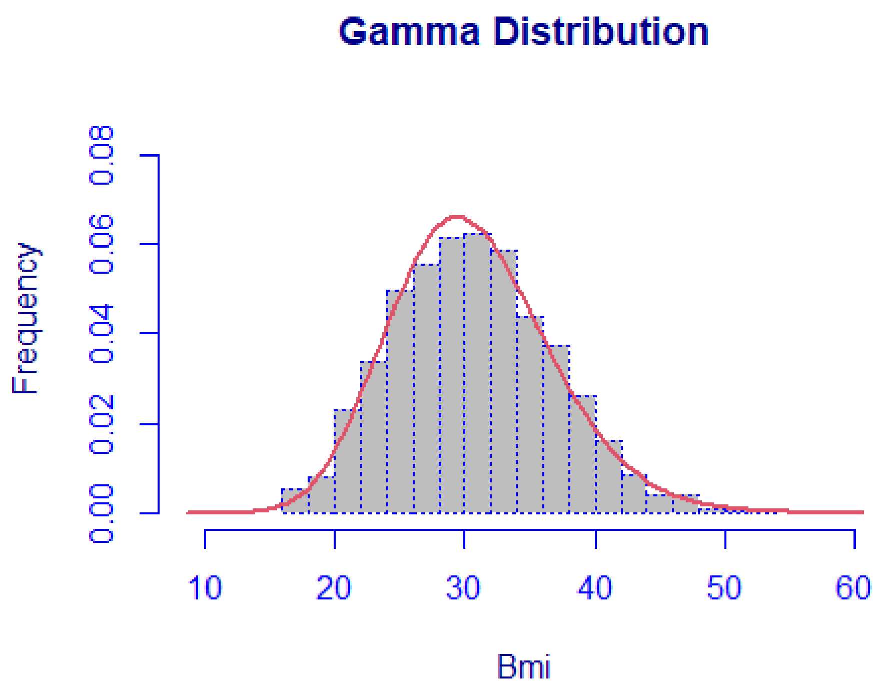
```
# Fit various distributions to the data and plot the histograms with fit lines
hist_data <- hist(dataset[, variable], plot = FALSE, breaks = nb, prob = TRUE)

## Warning in hist.default(dataset[, variable], plot = FALSE, breaks = nb, :
## l'argomento 'probability' non ha fatto uso di

max_f <- max(hist_data$density)
fit_gaussian <- histDist(dataset[, variable], family = NO, nbins = nb,
                         main = "Gaussian Distribution", xlab = plot_title,
                         ylab = "Frequency", ylim = c(0, max_f * 1.40))
```



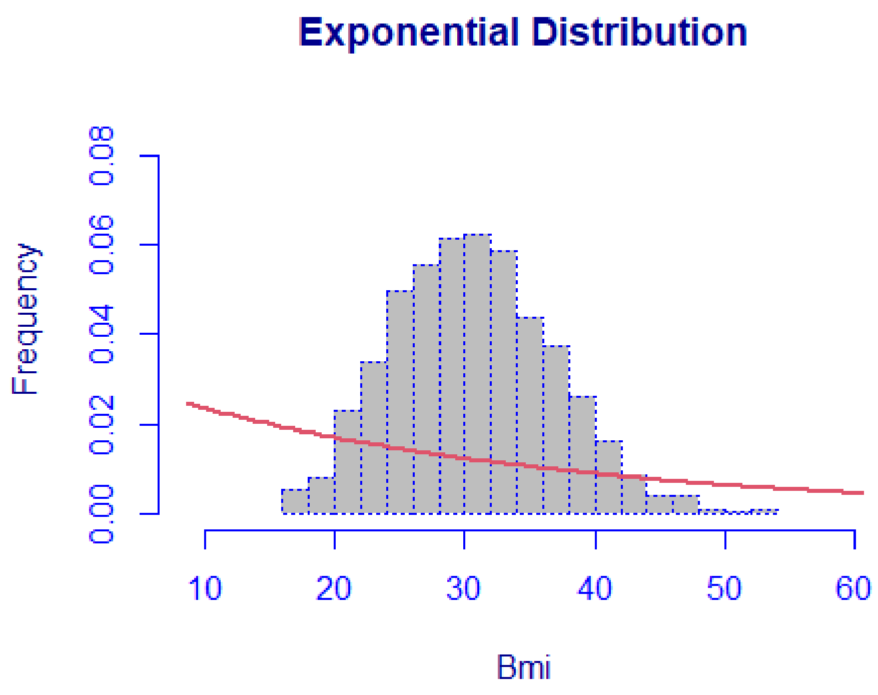
```
fit_logno <- histDist(dataset[, variable], family = LOGNO, nbins = nb,
                      main = "Log-Normal Distribution", xlab = plot_title,
                      ylab = "Frequency", ylim = c(0, max_f * 1.40))
```



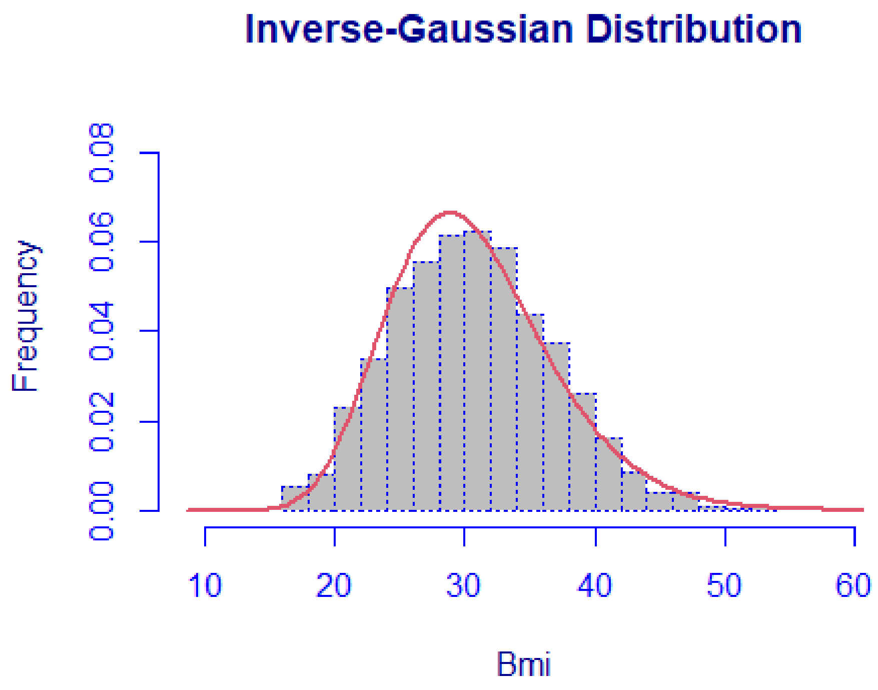
```
fit_gamma <- histDist(dataset[, variable], family = GA, nbins = nb,
                      main = "Gamma Distribution", xlab = plot_title,
                      ylab = "Frequency", ylim = c(0, max_f * 1.40))
```



```
fit_exponential <- histDist(dataset[, variable], family = EXP, nbins = nb,
                            main = "Exponential Distribution", xlab = plot_title,
                            ylab = "Frequency", ylim = c(0, max_f * 1.40))
```



```
fit_invgauss <- histDist(dataset[, variable], family = IG, nbins = nb,
                         main = "Inverse-Gaussian Distribution", xlab =
plot_title,
                         ylab = "Frequency", ylim = c(0, max_f * 1.40))
```



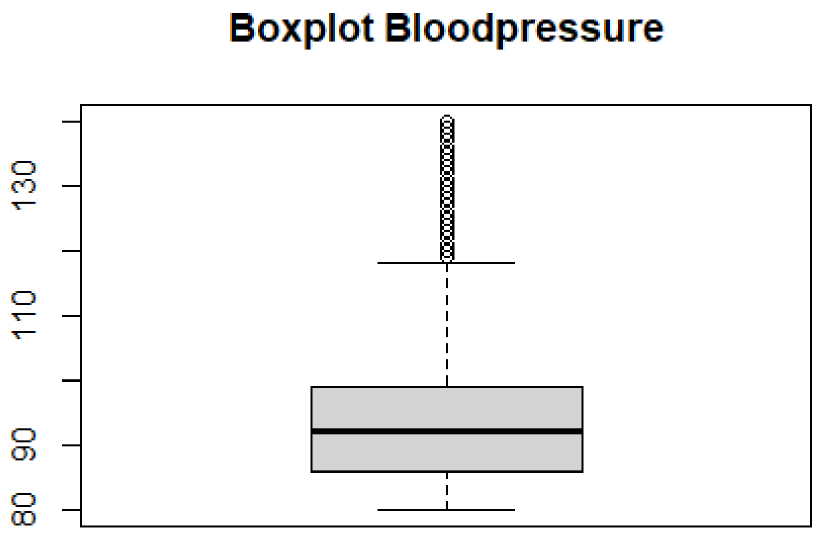
```
AIC_values <- c(AIC(fit_gaussian), AIC(fit_logno), AIC(fit_gamma),
AIC(fit_exponential), AIC(fit_invgauss))
BIC_values <-
c(fit_gaussian$sbc,fit_logno$sbc,fit_gamma$sbc,fit_exponential$sbc,fit_invgauss$sb
c)

goodness_of_fit <- data.frame(
  Distribution = c("Gaussian", "Log-Normal", "Gamma", "Exponential",
                   "Inverse-Gaussian"),
  AIC = AIC_values,
  BIC = BIC_values
)
print(goodness_of_fit)

##       Distribution       AIC       BIC
## 1         Gaussian  8608.604  8618.992
## 2       Log-Normal  8600.845  8611.234
## 3            Gamma  8586.695  8597.084
## 4      Exponential 11784.618 11789.812
## 5 Inverse-Gaussian  8600.674  8611.063
```

Bloodpressure Variable

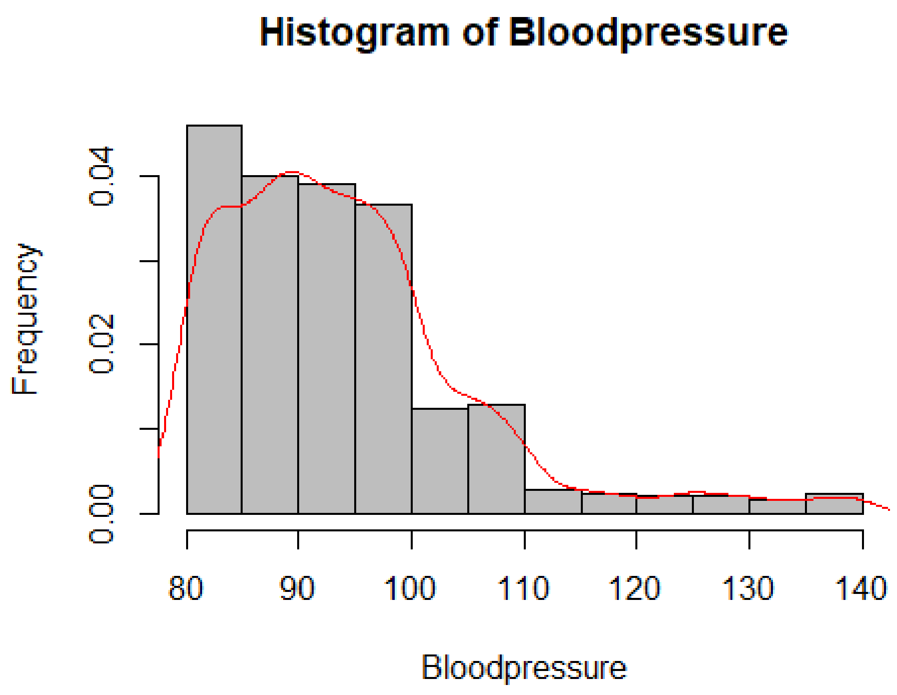
The bloodpressure variable is analyzed to understand its distribution and key statistics:

```
variable<-name[4]
print(variable)

## [1] "bloodpressure"

summary(select(dataset,variable))

##  bloodpressure
##  Min.   : 80.00
##  1st Qu.: 86.00
##  Median : 92.00
##  Mean   : 94.19
##  3rd Qu.: 99.00
##  Max.   :140.00

summary(select(dataset,variable))

##  bloodpressure
##  Min.   : 80.00
##  1st Qu.: 86.00
##  Median : 92.00
##  Mean   : 94.19
##  3rd Qu.: 99.00
##  Max.   :140.00

mean(dataset[,variable])

## [1] 94.18919
```

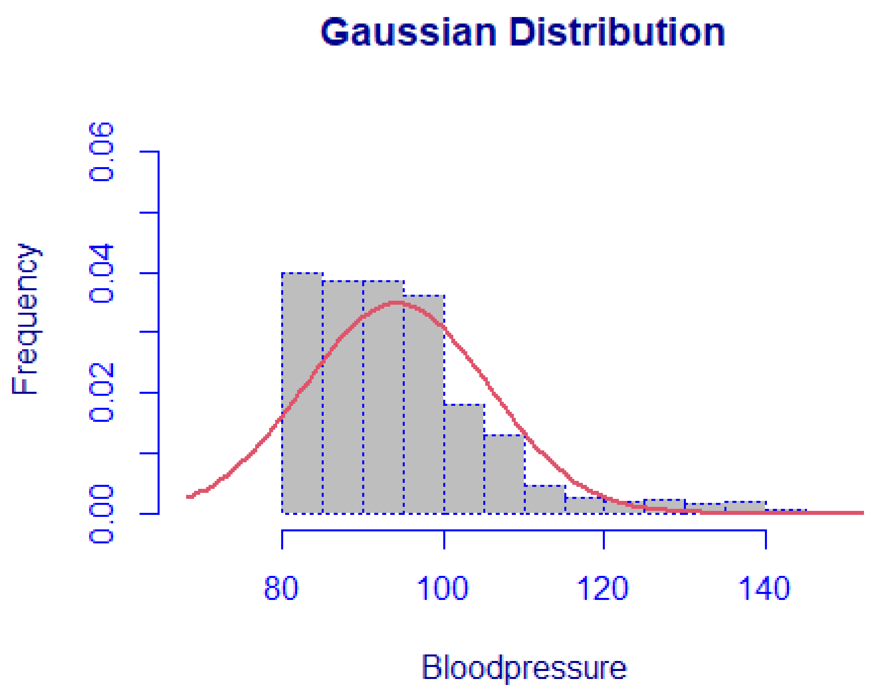
```
var(dataset[,variable])

## [1] 130.992

sd(dataset[,variable])

## [1] 11.44517

skewness(dataset[,variable])

## [1] 1.482186

kurtosis(dataset[,variable])

## [1] 5.867536

plot_title <- str_to_title(variable)
boxplot(dataset[,4], main = paste("Boxplot",plot_title))
```



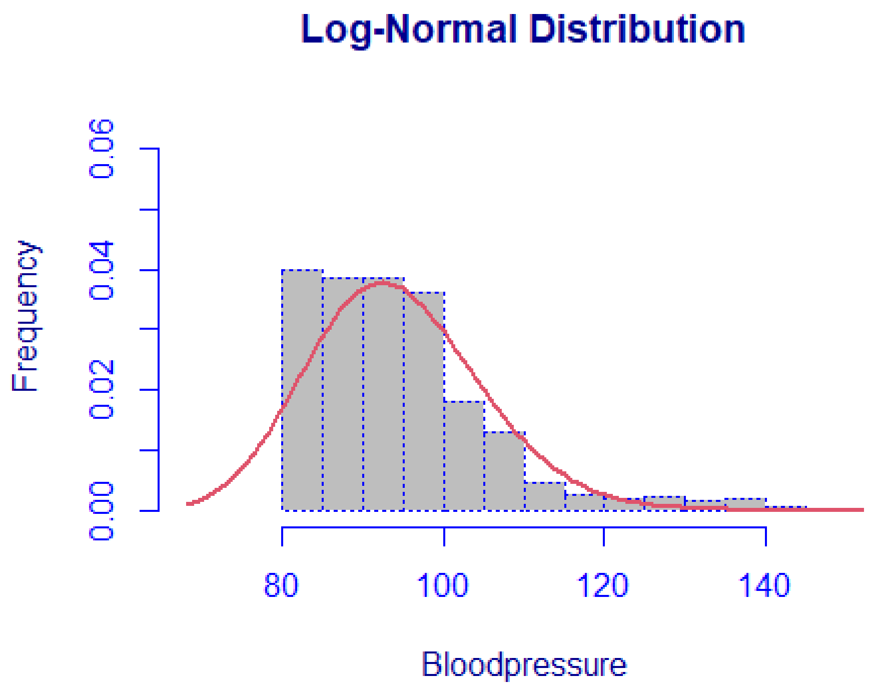
```
nb<-num_bins(dataset,variable)
hist(dataset[,variable], main = paste("Histogram of", plot_title), xlab =
plot_title, ylab = "Frequency", prob = TRUE, breaks = nb, col = "grey")
lines(density(dataset[,variable]), col = "red")
```



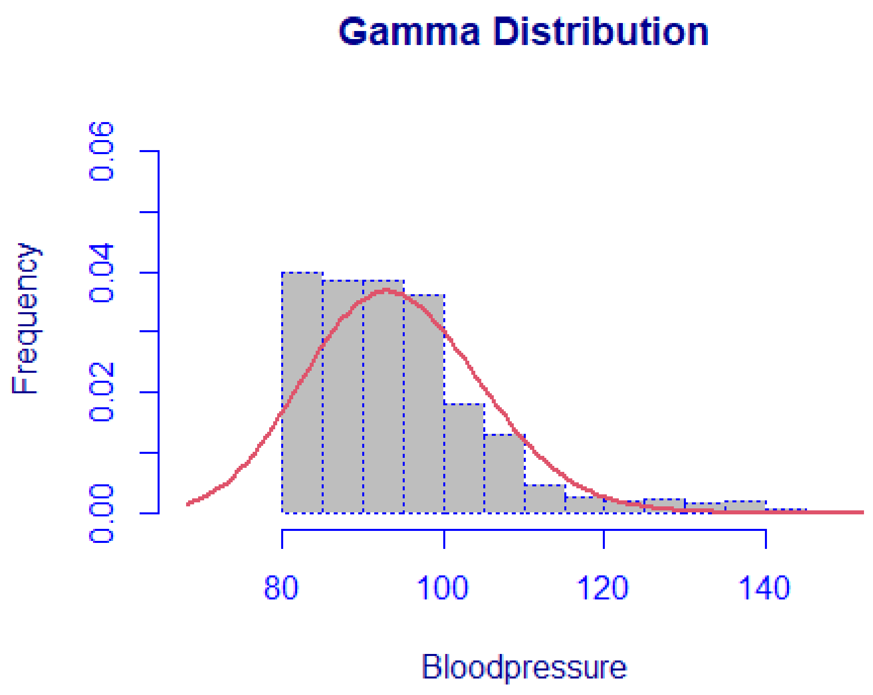
```
# Fit various distributions to the data and plot the histograms with fit lines
hist_data <- hist(dataset[, variable], plot = FALSE, breaks = nb, prob = TRUE)

## Warning in hist.default(dataset[, variable], plot = FALSE, breaks = nb, :
## l'argomento 'probability' non ha fatto uso di

max_f <- max(hist_data$density)
fit_gaussian <- histDist(dataset[, variable], family = NO, nbins = nb,
                         main = "Gaussian Distribution", xlab = plot_title,
                         ylab = "Frequency", ylim = c(0, max_f * 1.40))
```



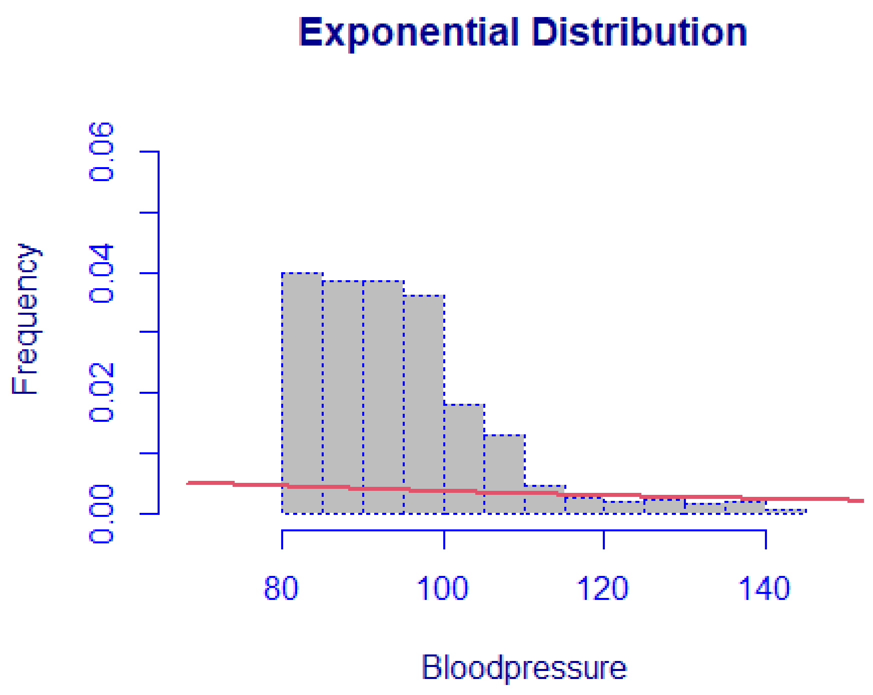
```
fit_logno <- histDist(dataset[, variable], family = LOGNO, nbins = nb,
                      main = "Log-Normal Distribution", xlab = plot_title,
                      ylab = "Frequency", ylim = c(0, max_f * 1.40))
```



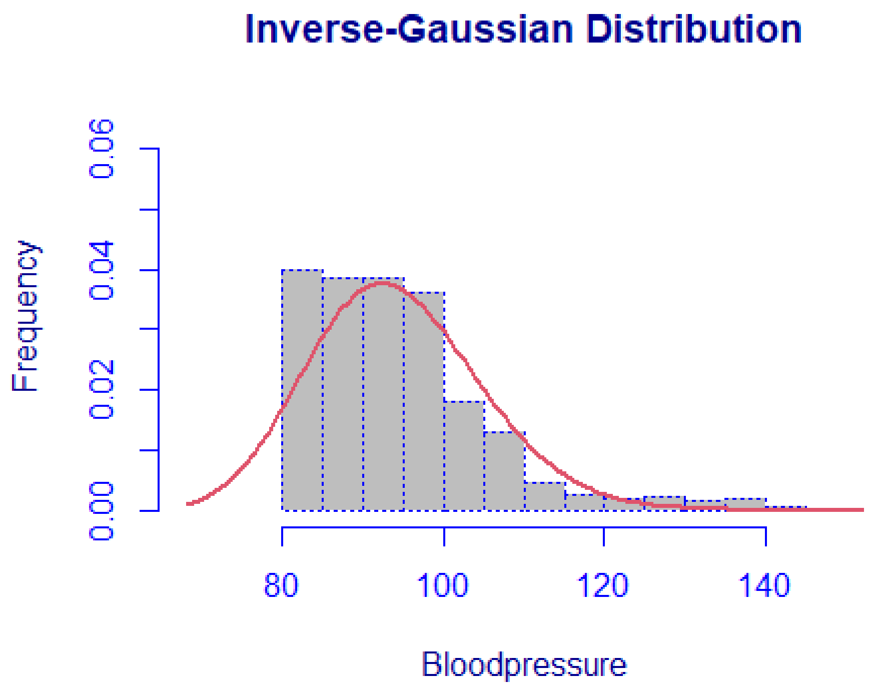
```
fit_gamma <- histDist(dataset[, variable], family = GA, nbins = nb,
                      main = "Gamma Distribution", xlab = plot_title,
                      ylab = "Frequency", ylim = c(0, max_f * 1.40))
```



```
fit_exponential <- histDist(dataset[, variable], family = EXP, nbins = nb,
                            main = "Exponential Distribution", xlab = plot_title,
                            ylab = "Frequency", ylim = c(0, max_f * 1.40))
```



```
fit_invgauss <- histDist(dataset[, variable], family = IG, nbins = nb,
                         main = "Inverse-Gaussian Distribution", xlab =
plot_title,
                         ylab = "Frequency", ylim = c(0, max_f * 1.40))
```



```
AIC_values <- c(AIC(fit_gaussian), AIC(fit_logno), AIC(fit_gamma),
AIC(fit_exponential), AIC(fit_invgauss))
BIC_values <-
c(fit_gaussian$sbc,fit_logno$sbc,fit_gamma$sbc,fit_exponential$sbc,fit_invgauss$sb
c)

goodness_of_fit <- data.frame(
  Distribution = c("Gaussian", "Log-Normal", "Gamma", "Exponential",
                   "Inverse-Gaussian"),
  AIC = AIC_values,
  BIC = BIC_values
)
print(goodness_of_fit)
```

```
##       Distribution      AIC      BIC
## 1         Gaussian 10276.73 10287.12
## 2       Log-Normal 10083.67 10094.06
## 3            Gamma 10141.52 10151.91
## 4      Exponential 14774.69 14779.89
## 5 Inverse-Gaussian 10085.35 10095.74
```

Diabetic Variable

The diabetic variable is analyzed to understand its distribution and key statistics:

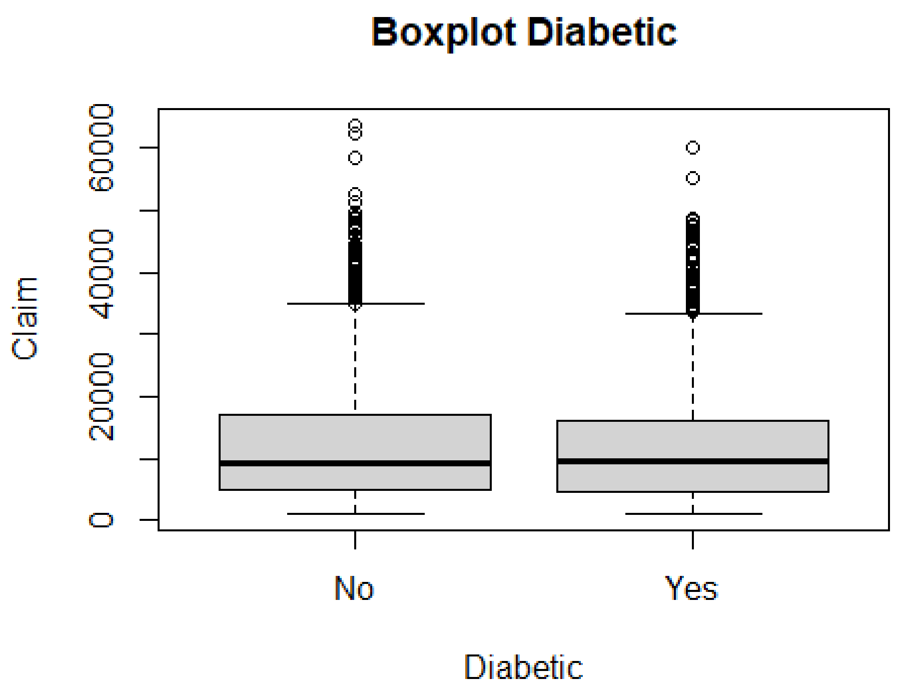
```
variable<-name[5]
print(variable)
```

```
## [1] "diabetic"
```

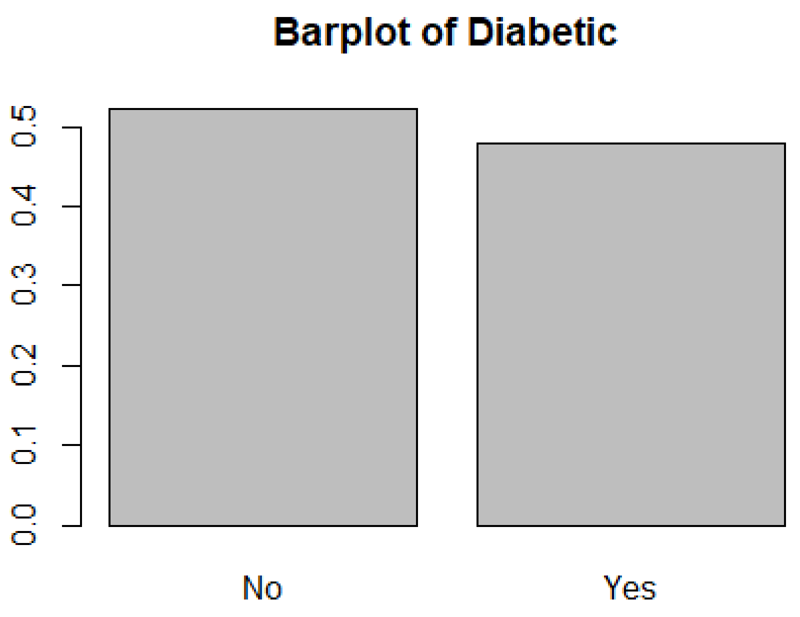
```
table(select(dataset,variable))
```

```
## diabetic
##  No Yes
## 695 637
```

```
plot_title <- str_to_title(variable)
boxplot(dataset$claim ~ dataset[,5], main = paste("Boxplot",plot_title),
xlab=plot_title, ylab="Claim")
```



```
relfreq <- table(dataset[,variable])/length(dataset[,variable])
relfreq

##
##        No       Yes
## 0.5217718 0.4782282

barplot(relfreq, main = paste("Barplot of", plot_title))
```

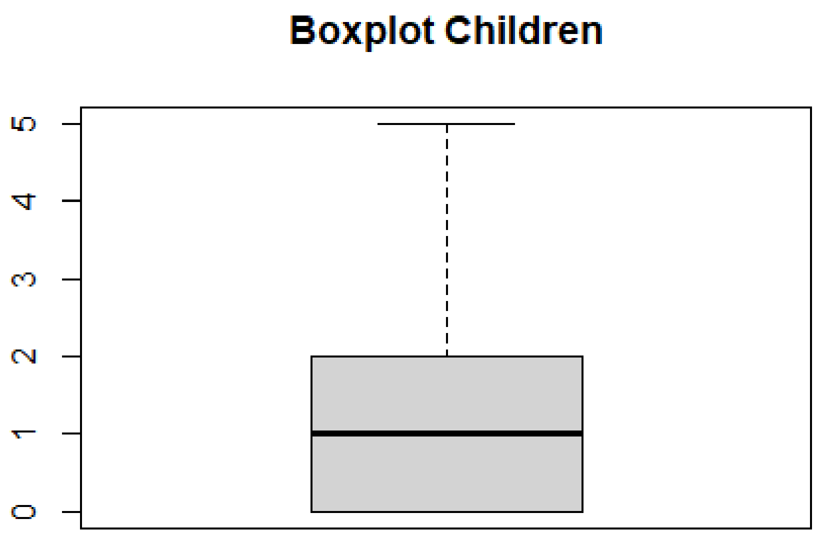


Children Variable

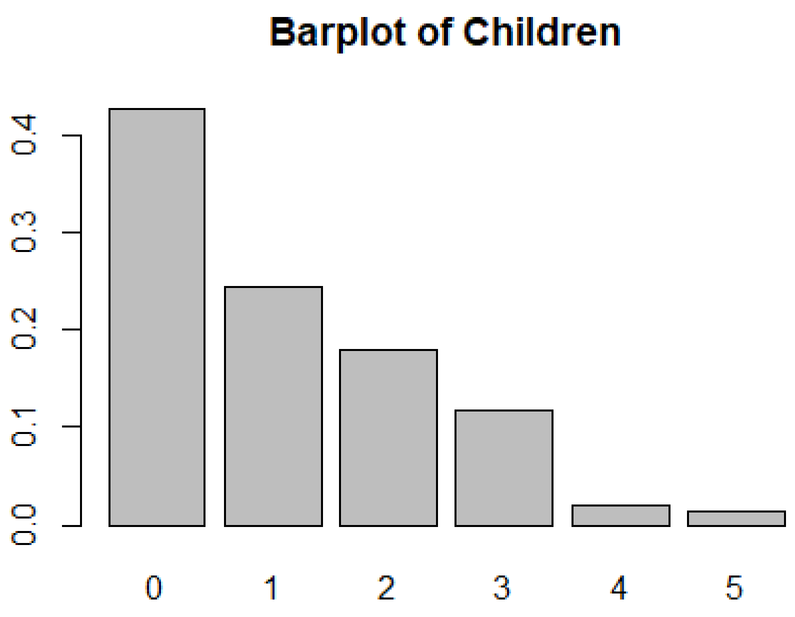
The children variable is analyzed to understand its distribution and key statistics:

```
variable<-name[6]
print(variable)

## [1] "children"

table(select(dataset,variable))

## children
##   0   1   2   3   4   5
## 568 324 240 157  25  18

summary(select(dataset,variable))

##     children
##  Min.   :0.0
##  1st Qu.:0.0
##  Median :1.0
##  Mean   :1.1
##  3rd Qu.:2.0
##  Max.   :5.0

mean(dataset[,variable])

## [1] 1.09985
```

```
var(dataset[,variable])

## [1] 1.454335

sd(dataset[,variable])

## [1] 1.205958

skewness(dataset[,variable])

## [1] 0.9315665

kurtosis(dataset[,variable])

## [1] 3.188275

plot_title <- str_to_title(variable)
boxplot(dataset[,6], main = paste("Boxplot",plot_title))
```



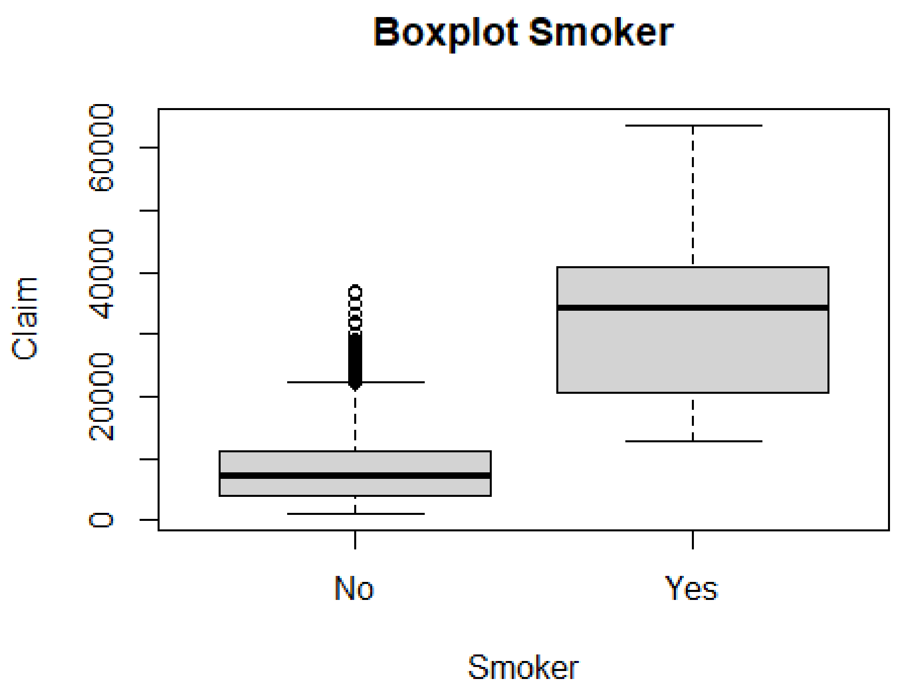
```
relfreq <- table(dataset[,variable])/length(dataset[,variable])
relfreq

##
##          0          1          2          3          4          5
## 0.42642643 0.24324324 0.18018018 0.11786787 0.01876877 0.01351351

barplot(relfreq, main = paste("Barplot of", plot_title))
```



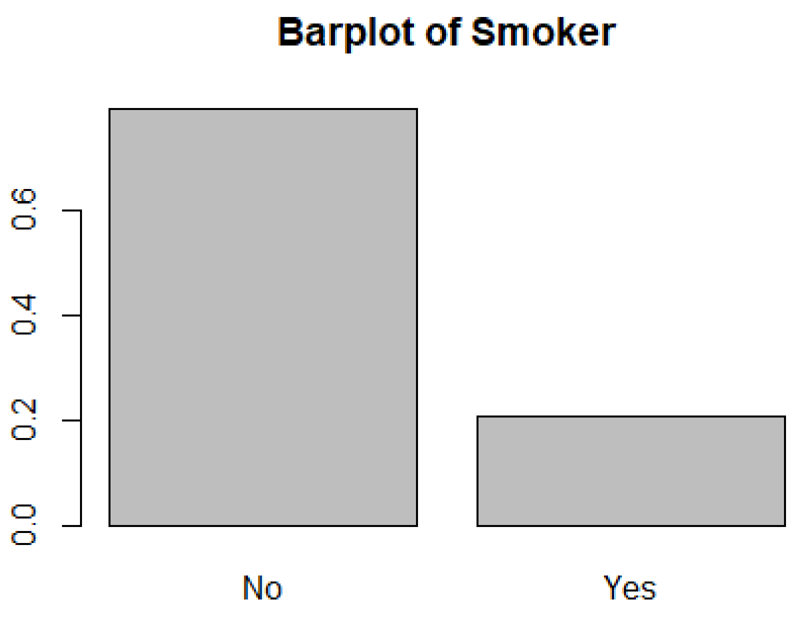
Smoker Variable

The smoker variable is analyzed to understand its distribution and key statistics:

```
variable<-name[7]
print(variable)

## [1] "smoker"

table(select(dataset,variable))  #this is to create a vertical output

## smoker
##   No  Yes
## 1058  274

plot_title <- str_to_title(variable)
boxplot(dataset$claim ~ dataset[,7], main = paste("Boxplot",plot_title),
xlab=plot_title, ylab="Claim")
```



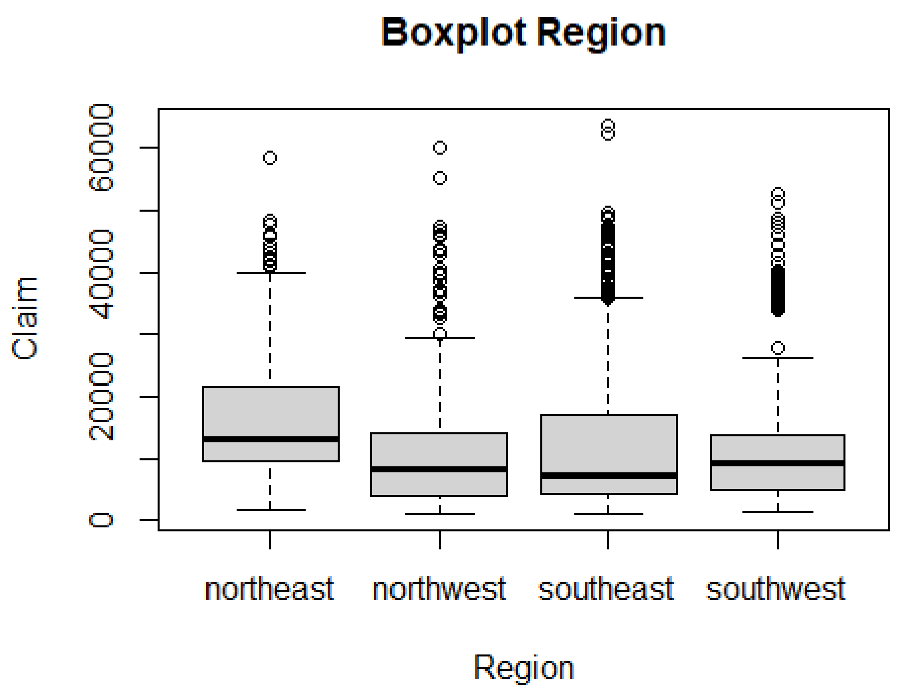
```
relfreq <- table(dataset[,variable])/length(dataset[,variable])
relfreq

##
##        No       Yes
## 0.7942943 0.2057057

barplot(relfreq, main = paste("Barplot of", plot_title))
```



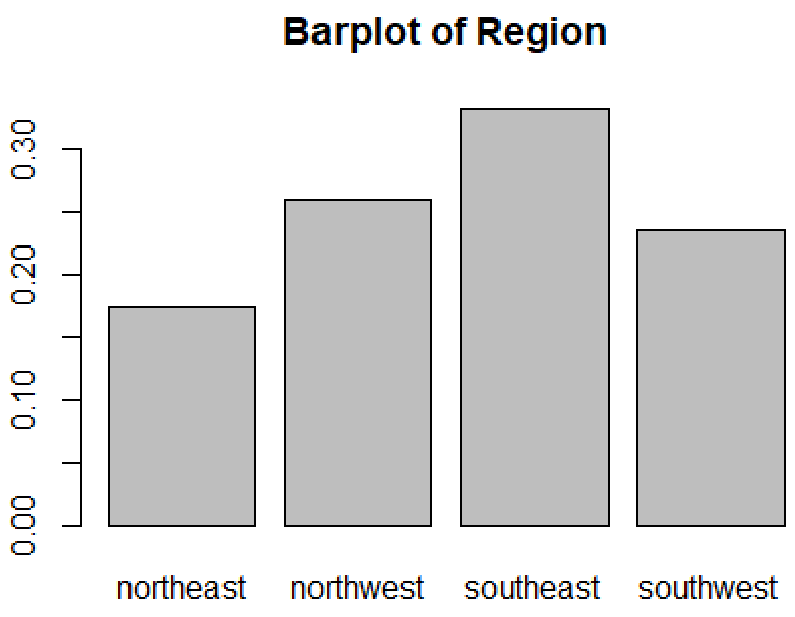
Region Variable

The region variable is analyzed to understand its distribution and key statistics:

```
variable<-name[8]
print(variable)

## [1] "region"

table(select(dataset,variable))  #this is to create a vertical output

## region
## northeast northwest southeast southwest
##       231       345       442       314

plot_title <- str_to_title(variable)
boxplot(dataset$claim ~ dataset[,8], main = paste("Boxplot",plot_title),
xlab=plot_title, ylab="Claim")
```



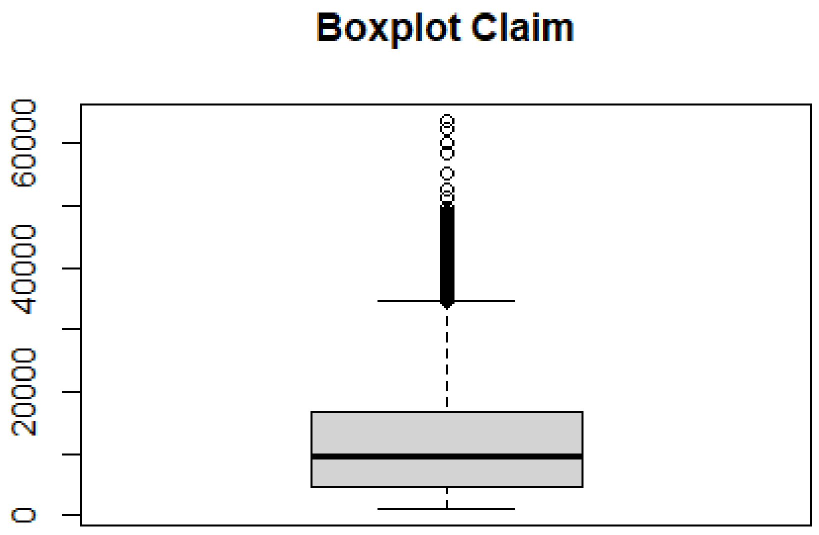
```
relfreq <- table(dataset[,variable])/length(dataset[,variable])
relfreq

##
## northeast northwest southeast southwest
## 0.1734234 0.2590090 0.3318318 0.2357357

barplot(relfreq, main = paste("Barplot of", plot_title))
```



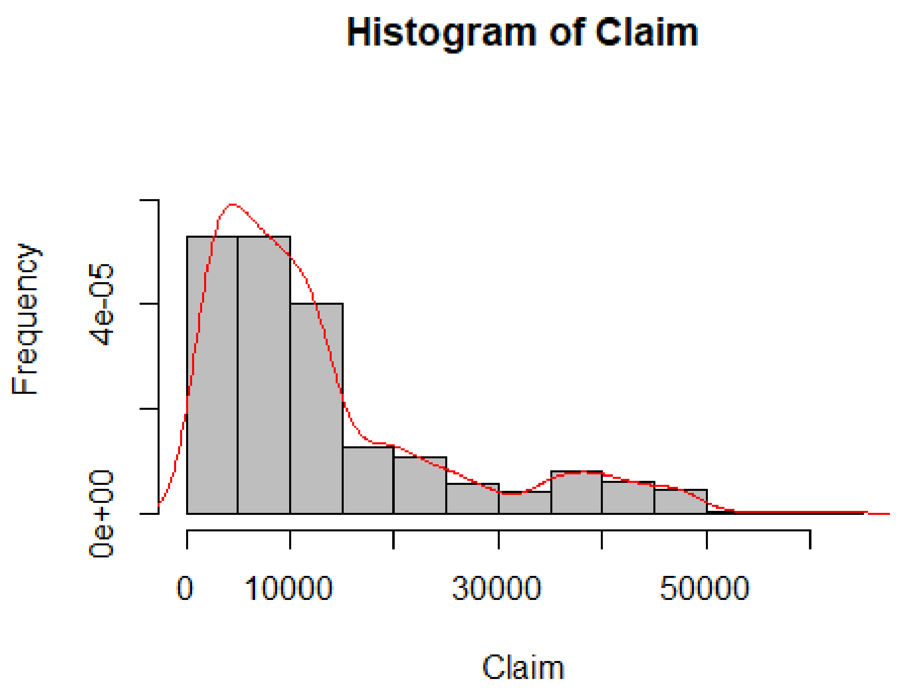
Claim Variable

The claim variable is analyzed to understand its distribution and key statistics:

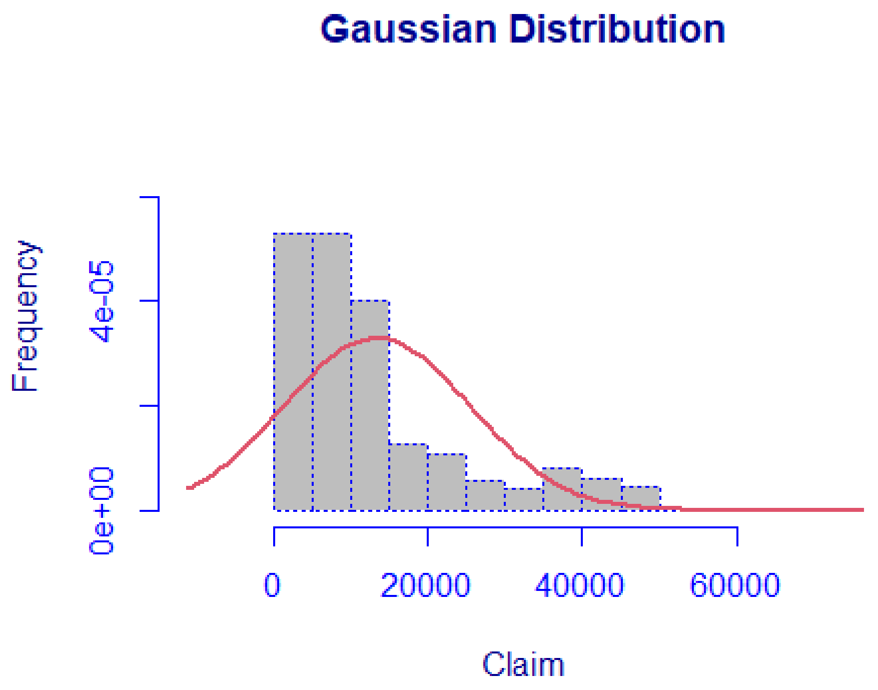
```
variable<-name[9]
print(variable)

## [1] "claim"

summary(select(dataset,variable))  #this is to create a vertical output

##      claim
##  Min.   : 1122
##  1st Qu.: 4760
##  Median : 9413
##  Mean   :13325
##  3rd Qu.:16781
##  Max.   :63770

plot_title <- str_to_title(variable)
boxplot(dataset[,9], main = paste("Boxplot",plot_title))
```



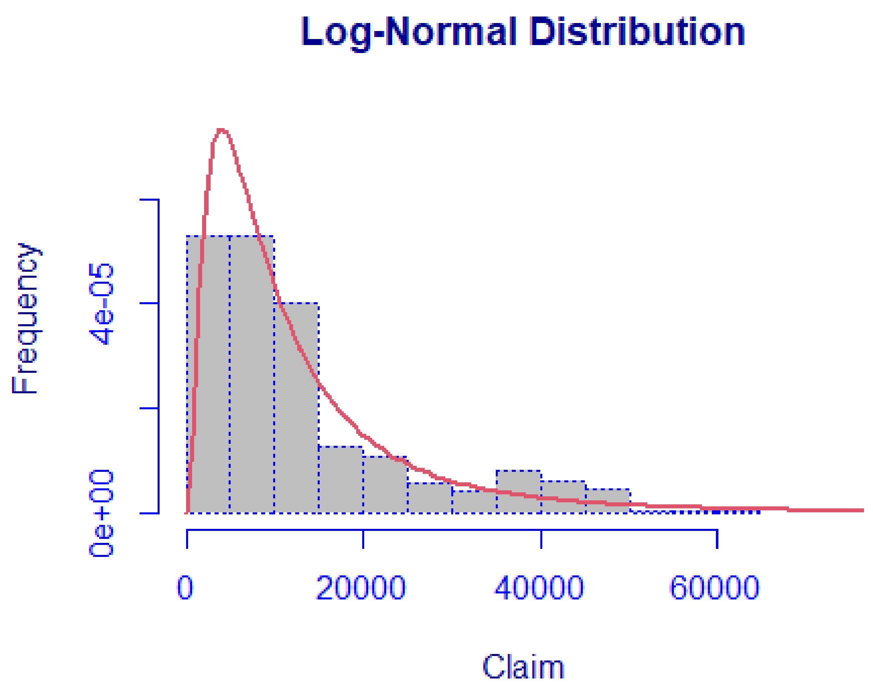
```
nb<-num_bins(dataset,variable)
hist_data <- hist(dataset[, variable], plot = FALSE, breaks = nb, prob = TRUE)

## Warning in hist.default(dataset[, variable], plot = FALSE, breaks = nb, :
## l'argomento 'probability' non ha fatto uso di

max_f <- max(hist_data$density)
hist(dataset[,variable], main = paste("Histogram of", plot_title), xlab =
plot_title, ylab = "Frequency", prob = TRUE, breaks = nb, col = "grey",ylim = c(0,
max_f * 1.40))
lines(density(dataset[,variable]), col = "red")
```



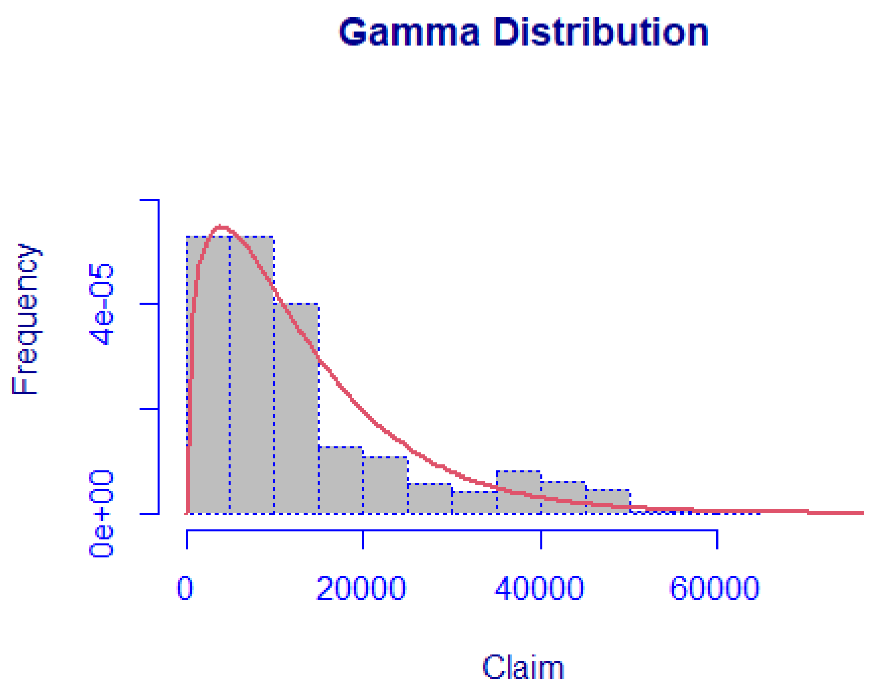
```
# Fit various distributions to the data and plot the histograms with fit lines
fit_gaussian <- histDist(dataset[, variable], family = NO, nbins = nb,
                         main = "Gaussian Distribution", xlab = plot_title,
                         ylab = "Frequency", ylim = c(0, max_f * 1.40))
```



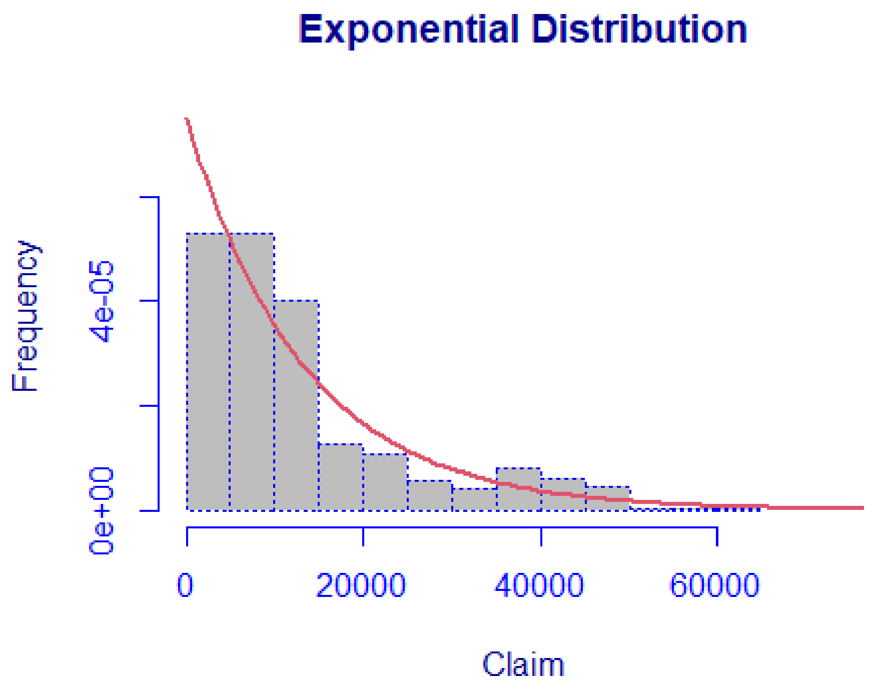
```
fit_logno <- histDist(dataset[, variable], family = LOGNO, nbins = nb,
                      main = "Log-Normal Distribution", xlab = plot_title,
                      ylab = "Frequency", ylim = c(0, max_f * 1.40))
```



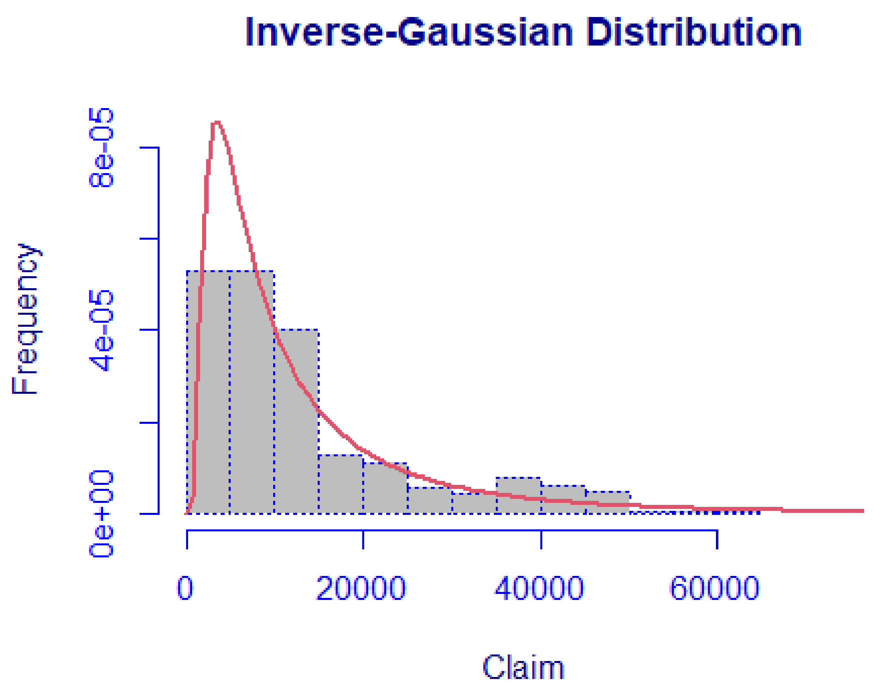
```
fit_gamma <- histDist(dataset[, variable], family = GA, nbins = nb,
                      main = "Gamma Distribution", xlab = plot_title,
                      ylab = "Frequency", ylim = c(0, max_f * 1.40))
```



```
fit_exponential <- histDist(dataset[, variable], family = EXP, nbins = nb,
                            main = "Exponential Distribution", xlab = plot_title,
                            ylab = "Frequency", ylim = c(0, max_f * 1.40))
```



```
fit_invgauss <- histDist(dataset[, variable], family = IG, nbins = nb,
                         main = "Inverse-Gaussian Distribution", xlab =
plot_title,
                         ylab = "Frequency", ylim = c(0, max_f * 1.60))
```



```
AIC_values <- c(AIC(fit_gaussian), AIC(fit_logno), AIC(fit_gamma),
AIC(fit_exponential), AIC(fit_invgauss))
BIC_values <-
c(fit_gaussian$sbc,fit_logno$sbc,fit_gamma$sbc,fit_exponential$sbc,fit_invgauss$sb
c)

goodness_of_fit <- data.frame(
  Distribution = c("Gaussian", "Log-Normal", "Gamma", "Exponential",
                   "Inverse-Gaussian"),
  AIC  = AIC_values,
  BIC  = BIC_values
)
print(goodness_of_fit)

##       Distribution      AIC      BIC
## 1         Gaussian 28829.33 28839.72
## 2       Log-Normal 27798.37 27808.76
## 3            Gamma 27875.77 27886.16
## 4      Exponential 27967.12 27972.31
## 5 Inverse-Gaussian 27785.90 27796.29
```

It is noteworthy to highlight the presence of outliers across several variables, particularly in “claim”. Given the medical nature of the data, I opted not to delete or use winsorizing to mitigate the impact of outliers in the dataset. Specifically for “claim”, I chose to retain outliers to explore how these extreme cases of claims are distributed and whether they reveal underlying patterns or narratives.

3. Multivariate Analysis and PCA

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a fundamental unsupervised statistical technique used for dimensionality reduction in datasets without labeled outputs. By identifying correlations among variables in the original dataset, PCA aims to transform the data into a new space defined by a smaller set of orthogonal components called principal components (PCs). These components are prioritized based on their ability to explain the maximum variance in the original dataset, with each subsequent component explaining a decreasing fraction of the variability. Despite being latent variables derived from linear combinations of mean-centered original variables, interpreting each principal component can be complex.

To effectively apply Principal Component Analysis (PCA) to the dataset, it is crucial to assess the interrelationships among variables through correlation analysis. Higher correlations indicate stronger contributions of variables to explaining overall variability and enhance PCA’s effectiveness in dimensionality reduction.

In R, this initial step involves using the “psych” and “corrplot” packages to visualize the correlation matrix of the dataset.

We begin by selecting the numerical variables:

```
n_data <- dataset[, sapply(dataset, is.numeric)]
head(n_data)
```

```
##    age  bmi bloodpressure children   claim
## 1   39 23.2            91        0 1121.87
## 2   24 30.1            87        0 1131.51
## 8   19 41.1           100        0 1146.80
## 9   20 43.0            86        0 1149.40
## 10  30 53.1            97        0 1163.46
## 11  36 19.8            88        0 1241.57
```

This feature will be used to compute the covariance and correlation matrices:

```
cov_matrix <- cov(n_data)  # Covariance matrix
corr_matrix <- cor(n_data)  # Correlation matrix
cov_matrix
```

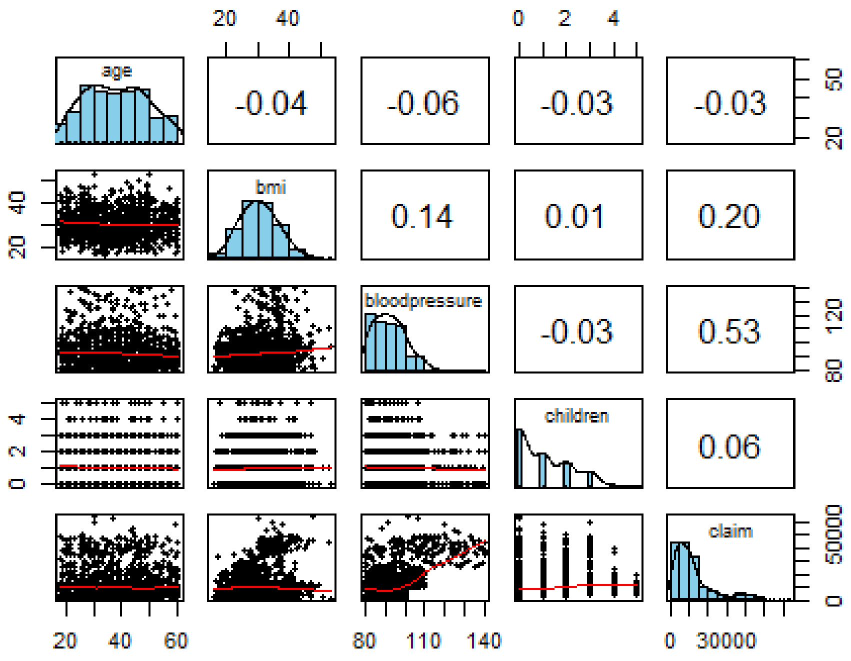
```
##                         age           bmi bloodpressure    children
## age             123.4944187    -2.8662722    -7.6722440  -0.3512267
## bmi              -2.8662722    37.4417562    10.1531180   0.1012334
## bloodpressure    -7.6722440    10.1531180   130.9919792  -0.4742015
## children         -0.3512267     0.1012334    -0.4742015   1.4543351
## claim         -3853.6518175 14815.6061605 73631.3596097 936.8056238
##                     claim
## age          -3.853652e+03
## bmi           1.481561e+04
## bloodpressure  7.363136e+04
## children      9.368056e+02
## claim         1.466429e+08
```

```
corr_matrix
```

```
##                     age         bmi bloodpressure    children       claim
## age          1.00000000 -0.04215176   -0.06032209 -0.02620786 -0.02863639
## bmi         -0.04215176  1.00000000    0.14497696  0.01371872  0.19994511
## bloodpressure -0.06032209  0.14497696    1.00000000 -0.03435645  0.53126344
## children    -0.02620786  0.01371872   -0.03435645  1.00000000  0.06414853
## claim       -0.02863639  0.19994511    0.53126344  0.06414853  1.00000000
```

To visualize the pairwise relationships between variables, we use the pairs.panels function, which provides histograms along the diagonal and scatter plots with correlations:

```
pairs.panels(n_data, hist.col="skyblue", density=TRUE, ellipses=FALSE)
```



Analyzing the plot produced by the pair function, it is observed that there is generally a low level of correlation among the numerical variables in the dataset. Exceptions include the correlation between "claim" and "blood pressure" with a coefficient of 0.52, "claim" and "bmi" with a coefficient of 0.20, and "blood pressure" and "bmi" with a coefficient of 0.14.

In preparation for PCA, the variables are standardized using the scale function. This standardization transforms the dataset to operate with the sample correlation matrix rather than the covariance matrix. PCA typically requires Eigen decomposition of either the sample covariance matrix (if data are not standardized) or the sample correlation matrix (if standardized). Utilizing the correlation matrix is advantageous as it enhances the interpretability and utility of PCA in subsequent analytical procedures.

This preliminary standardization step ensures that PCA can effectively identify and extract the principal components that capture the maximum variability present in the dataset, facilitating deeper insights into the underlying structure of the data.

```
scaled_data <- scale(n_data, center = TRUE, scale = TRUE) # scale data
eigen_R <- eigen(corr_matrix) # eigen decomposition of scaled data
eigen_R

## eigen() decomposition
## $values
## [1] 1.6362314 1.0237487 0.9794611 0.9050767 0.4554821
##
## $vectors
##             [,1]        [,2]        [,3]        [,4]        [,5]
## [1,]  0.11673850  0.57184317  0.79755729 -0.14381774 -0.05085618
## [2,] -0.36334621 -0.09169823 -0.05241402 -0.92226997 -0.07900526
## [3,] -0.64391619  0.16365211 -0.01327467  0.29691068 -0.68575343
## [4,] -0.04457993 -0.79731781  0.58352225  0.07459032 -0.12741665
## [5,] -0.66161637  0.04570692  0.14311083  0.18712265  0.71040767

phi <- -eigen_R$vectors # loadings multiplied by -1 for graphical interpretation
rownames(phi) <- c(names(n_data))
colnames(phi) <- c("PC1", "PC2", "PC3", "PC4", "PC5")
phi

##                       PC1         PC2         PC3         PC4         PC5
## age           -0.11673850 -0.57184317 -0.79755729  0.14381774  0.05085618
## bmi            0.36334621  0.09169823  0.05241402  0.92226997  0.07900526
## bloodpressure  0.64391619 -0.16365211  0.01327467 -0.29691068  0.68575343
## children       0.04457993  0.79731781 -0.58352225 -0.07459032  0.12741665
## claim          0.66161637 -0.04570692 -0.14311083 -0.18712265 -0.71040767

var_pca <- eigen_R$values # variability explained by PCs
var_pca

## [1] 1.6362314 1.0237487 0.9794611 0.9050767 0.4554821
```

After performing Eigen decomposition on the sample correlation matrix, the results consist of two main components: the matrix of Eigenvectors and the vector of Eigenvalues.

The Eigenvectors matrix, referred to as “phi,” defines the axes of the new space created by PCA. Each column represents a principal component, with the entries (loadings) indicating the weight of each original standardized variable in defining that component. These loadings facilitate the interpretation of how each principal component relates to the original variables. The matrix phi has been adjusted by multiplying its entries by -1 for easier interpretation.

Additionally, rows and columns of phi have been renamed to illustrate how each principal component is a linear combination of the standardized variables, weighted by these loadings. The orthogonality of the new variable space is ensured, with loadings showing an inverse relationship between successive components.

The Eigenvalues vector, named “var_pca,” quantifies the proportion of variability explained by each principal component. These values are arranged in decreasing order, indicating that the first few components capture the majority of the dataset’s variability.

Alternative PCA

PCA can also be computed using the `prcomp` function in R, which performs Principal Component Analysis on numeric data. The function takes the dataset as input and optionally scales the variables to have unit variance before analysis, which is controlled by the `scale` parameter set to `TRUE`.

```
pca_data <- prcomp(n_data, scale = TRUE)
pca_data

## Standard deviations (1, .., p=5):
## [1] 1.2791526 1.0118047 0.9896773 0.9513552 0.6748941
##
## Rotation (n x k) = (5 x 5):
##                      PC1         PC2         PC3         PC4         PC5
## age          -0.11673850  0.57184317 -0.79755729  0.14381774  0.05085618
## bmi           0.36334621 -0.09169823  0.05241402  0.92226997  0.07900526
## bloodpressure  0.64391619  0.16365211  0.01327467 -0.29691068  0.68575343
## children      0.04457993 -0.79731781 -0.58352225 -0.07459032  0.12741665
## claim         0.66161637  0.04570692 -0.14311083 -0.18712265 -0.71040767
```

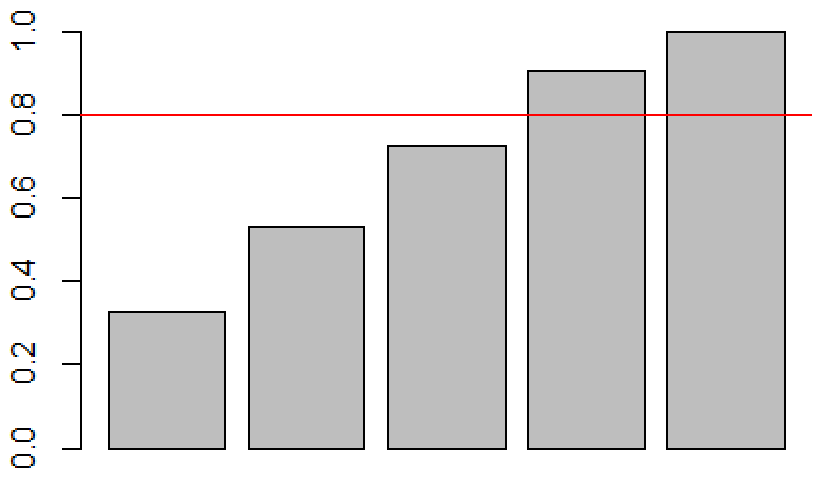
Choice of the number of the principal components

The next step involves selecting the number of principal components in PCA, aiming for a reduced-dimensional space compared to the original dataset. Several heuristic rules guide this selection process, including the cumulative proportion of variance explained (CPVE), Kaiser’s rule, and the scree plot criterion. These criteria will be applied in the subsequent analysis.

According to the cumulative proportion of variance explained criterion, the goal is to retain enough principal components to account for at least 80% of the variability present in the original dataset.

```
# Choosing the number of principal component

PVE <- var_pca/sum(var_pca)
PVE <- round(PVE, 3)
PVE

## [1] 0.327 0.205 0.196 0.181 0.091
```

```r
CPVE <- cumsum(PVE) #cumulative variability explained.
CPVE

## [1] 0.327 0.532 0.728 0.909 1.000

barplot(CPVE)
abline(h = 0.8, col = "red", lwd = 1.5)
```



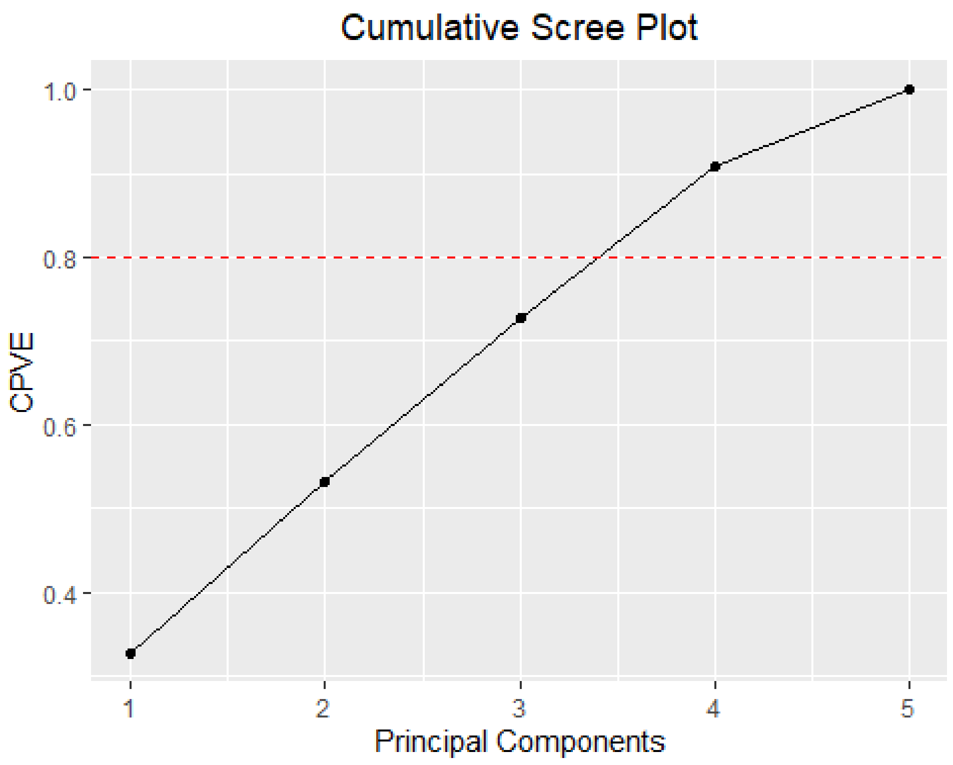
According to CPVE criterion, 4 principal components should been chosen, due to the fact that they are able to explain the 90.9% of the variability.

Kaiser's rule advises selecting the number of principal components based on a benchmark value, typically set at 1 for standardized variables. Alternatively, for original variables, it is based on the mean of the variances of the principal components. The rule suggests retaining principal components whose variance exceeds this benchmark value.

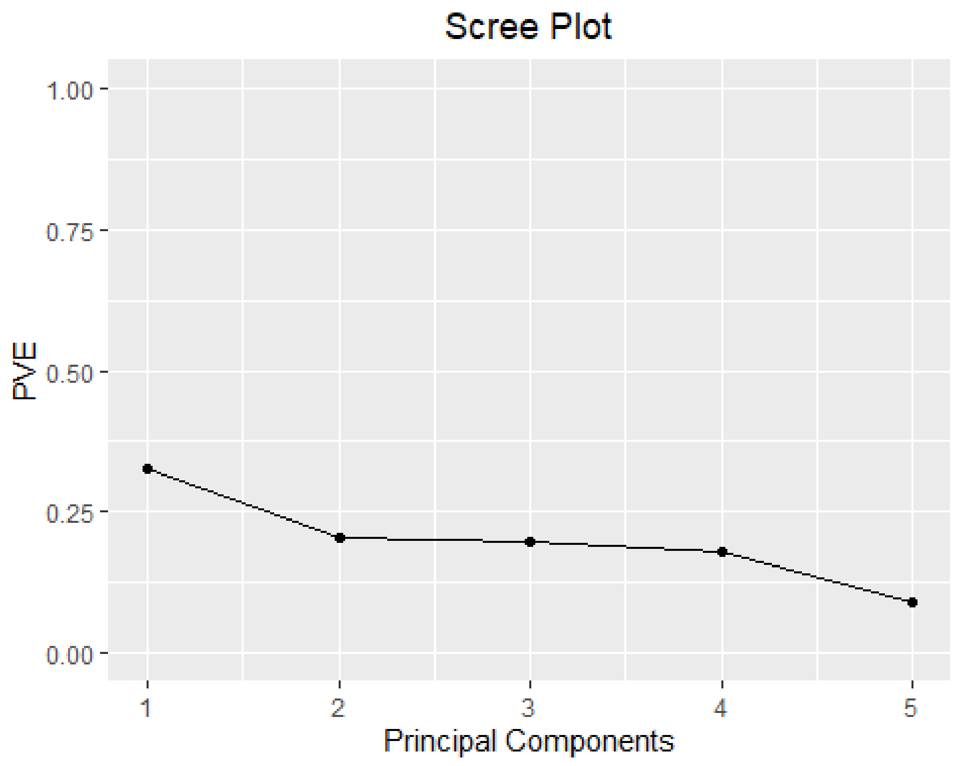
```r
benchmark <- sum(var_pca) / length(var_pca)
print(var_pca)

## [1] 1.6362314 1.0237487 0.9794611 0.9050767 0.4554821

which(var_pca > 1)

## [1] 1 2
```

The scree plot is a graphical tool used in Principal Component Analysis (PCA) to visualize the proportion of variance explained by each principal component. It plots the number of principal components on the x-axis and either the proportion of variance explained or cumulative proportion of variance explained on the y-axis. The key objective is to identify an “elbow” in the plot where the variance explained starts to level off. This elbow point typically indicates the optimal number of principal components to retain for dimensionality reduction. Implementing this criterion in R requires using the “ggplot2” package for plotting the scree plot.

```
CPVE_plot <- qplot(c(1:5), CPVE) +
  geom_line() +
  xlab("Principal Components") +
  ylab("CPVE") +
  ggtitle("Cumulative Scree Plot") +
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_hline(yintercept = 0.8, col = "red", linetype = "dashed")

## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

CPVE_plot # CPVE in relation to PCs number
```



```
PVE_plot <- qplot(c(1:5), PVE) +
  geom_line() +
  xlab("Principal Components") +
  ylab("PVE") +
  ggtitle("Scree Plot") +
  ylim(0, 1) +
  theme(plot.title = element_text(hjust = 0.5))
PVE_plot # PVE in relation to PCs number
```



The scree plot does not exhibit a clear elbow.

Given that the three methods yield different results, I have decided to retain three principal components.

#Interpretation

```
final_phi <- phi[,c(1:3)]
final_phi

##                      PC1         PC2         PC3
## age          -0.11673850 -0.57184317 -0.79755729
## bmi           0.36334621  0.09169823  0.05241402
## bloodpressure  0.64391619 -0.16365211  0.01327467
## children      0.04457993  0.79731781 -0.58352225
## claim         0.66161637 -0.04570692 -0.14311083
```

For PC1, the hierarchy of impacts of the original variables on the explanation of variability (in absolute value) is: claim, bloodpressure, bmi, age, children.

The first principal component, predominantly influenced by the bloodpressure and claim variables, can be interpreted as summarizing the combined effects of these factors. This component captures the largest variation in the data, with claim having the highest positive influence followed closely by bloodpressure.

The second principal component is primarily driven by the children and age variables. This component can be interpreted as capturing variations related to these demographics, with children having the strongest positive influence and age having a strong negative influence.

The third principal component, largely influenced by age and children with opposing signs, represents a contrasting relationship between these two variables. age has a strong negative influence, whereas children has a strong positive influence, indicating this component captures a dimension where these two variables vary inversely.

This analysis suggests that claim and bloodpressure are the most significant factors in the dataset, followed by children and age which have opposing impacts in the second and third components, respectively. The bmi variable has relatively lower influence across all components.

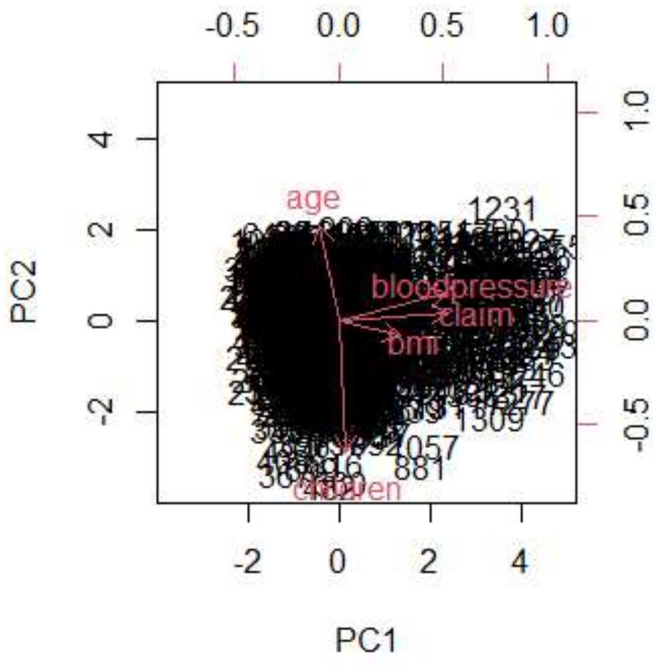
In the next R code segment, we initiate the analysis with setting a seed (`set.seed(123)`) to ensure reproducibility of random processes. The subsequent `biplot(pca_data, scale = 0)` generates a biplot visualizing the Principal Component Analysis (PCA) results for the original dataset `n_data`.

Following this, we partition the dataset based on the 'gender' variable using `createDataPartition(dataset$gender, p = 0.2, list = FALSE)`, selecting 20% of the data (`sdata`) for further analysis. PCA is then applied to this subset (`pca_sdata <- prcomp(sdata, scale = TRUE)`), ensuring standardization of variables (`scale = TRUE`). Another biplot (`biplot(pca_sdata, scale = 0)`) is created to visualize the PCA results specifically for the partitioned data.

These visualizations, following the partitioning steps, offer a clearer understanding of how PCA operates.

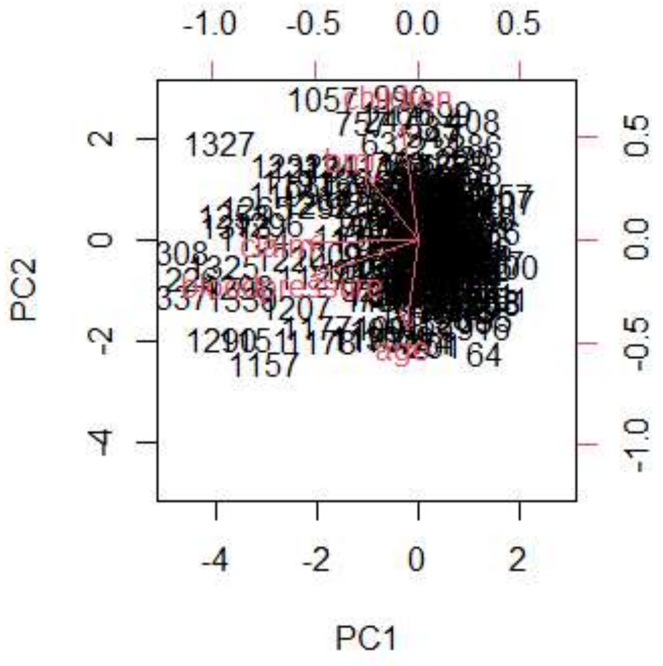
Complete biplot:

```r
set.seed(123)
biplot(pca_data, scale=0)
```



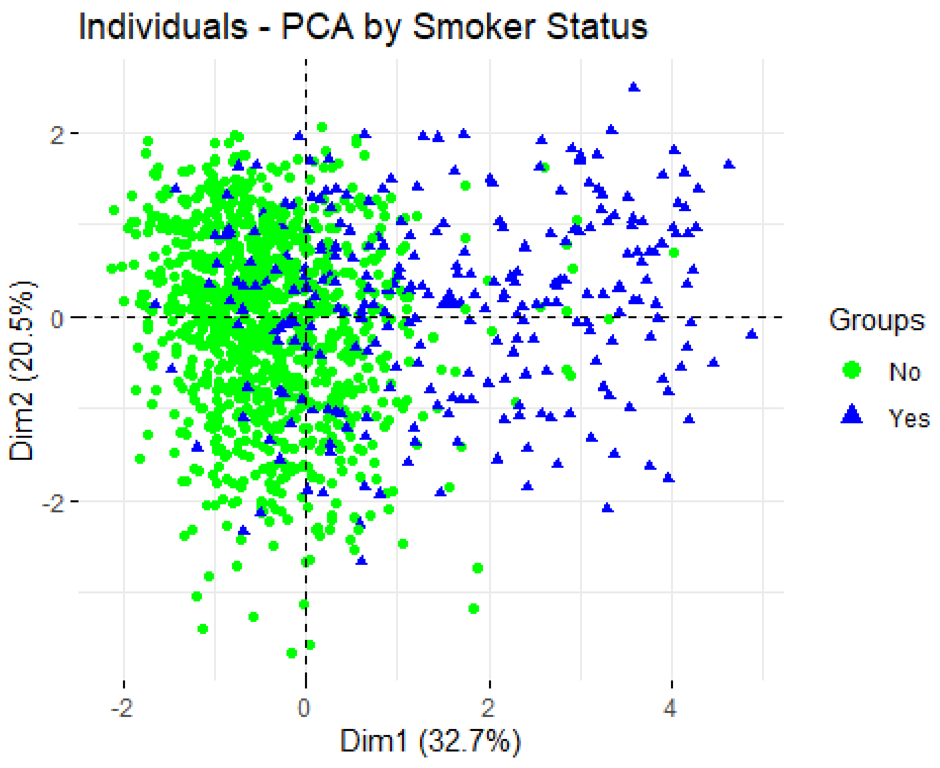
Sample biplot:

```
sdata_index <- createDataPartition(dataset$gender, p = 0.2, list = FALSE)
sdata  <- n_data[sdata_index, ]
pca_sdata <- prcomp(sdata, scale=TRUE)
biplot(pca_sdata, scale=0)
```

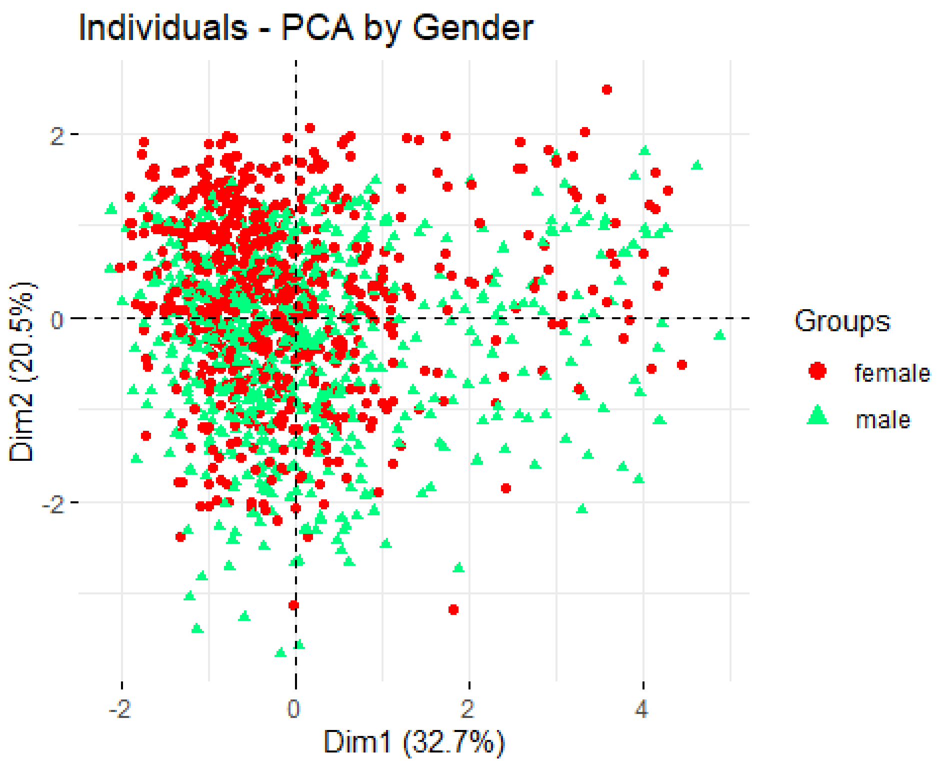


```
pca_data$x <- pca_data$x
final_phi <- as.data.frame(pca_data$x[, 1:3])
fviz_pca_ind(pca_data, geom.ind = "point", habillage = dataset$smoker, palette =
c("green","blue"),title = "Individuals - PCA by Smoker Status")
```

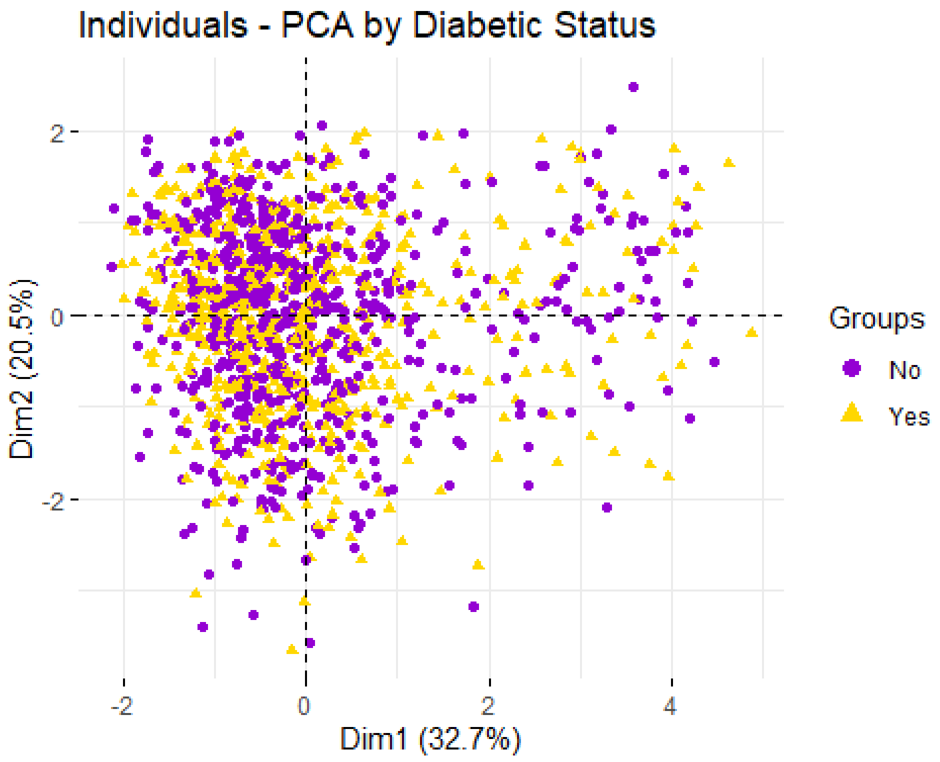
In each next graph, the points are separated by groups defined by various categorical variables. Notably, it is evident that smokers are generally associated with higher values of claims, blood pressure, and BMI.



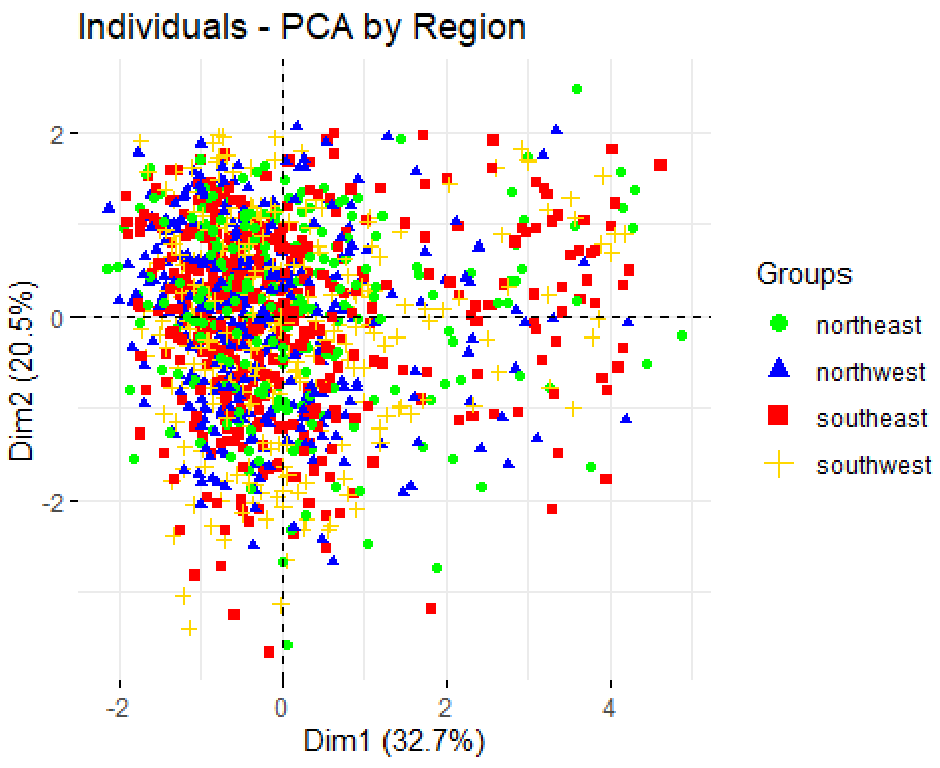
```r
fviz_pca_ind(pca_data, geom.ind = "point", habillage = dataset$gender, palette =
c("red","springgreen") , title = "Individuals - PCA by Gender")
```



```
fviz_pca_ind(pca_data, geom.ind = "point", habillage = dataset$diabetic, palette =
c("darkviolet","gold"), title = "Individuals - PCA by Diabetic Status")
```



```
fviz_pca_ind(pca_data, geom.ind = "point", habillage = dataset$region, palette =
c("green","blue","red","gold"), title = "Individuals - PCA by Region")
```



4. CLUSTER ANALYSIS

The next phase of the analysis is Cluster Analysis (CA), which focuses on identifying patterns among the statistical units within the dataset. CA aims to group these units into clusters based on their similarity according to predefined criteria.

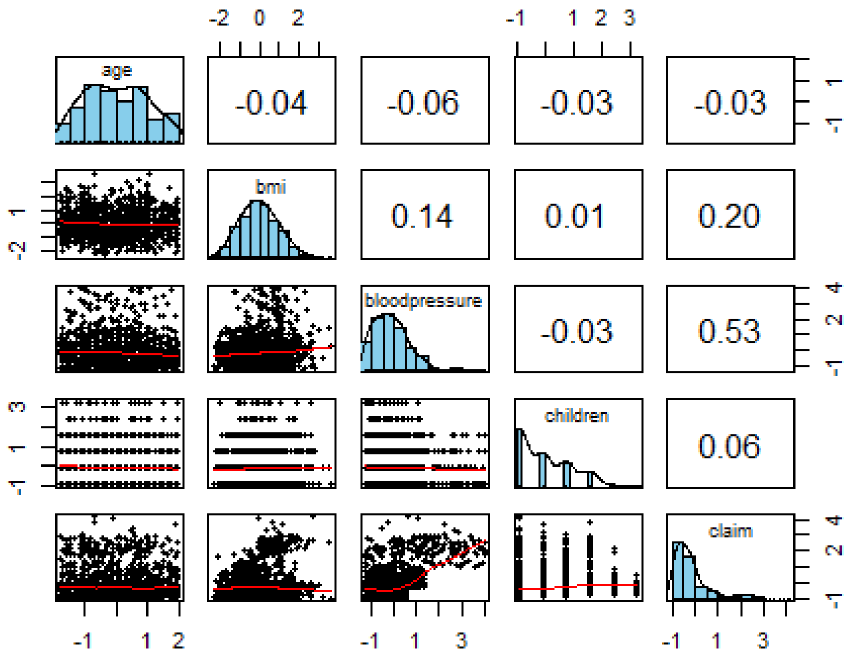
Step 1: Assessment of Cluster Tendency

Before applying clustering algorithms, it is crucial to assess whether the dataset exhibits a natural tendency to form clusters. This initial step, known as "cluster tendency assessment," determines whether meaningful clusters can be identified in the data. This assessment compares the observed dataset with a theoretical benchmark where data points are uniformly distributed at random.

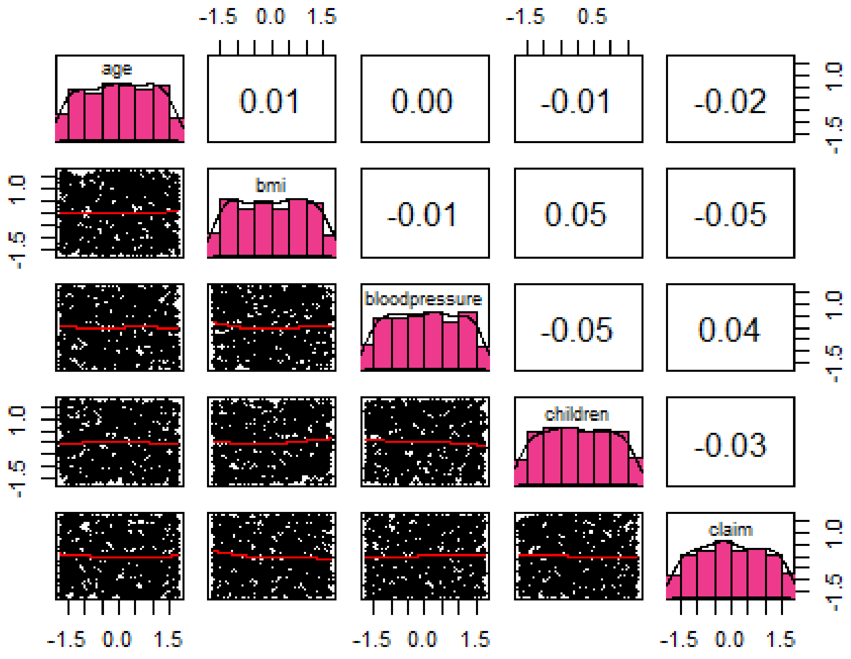
The significance of this step lies in its ability to discern whether clustering algorithms are appropriate for the dataset. A larger disparity between the observed data and the random uniform distribution benchmark suggests a higher likelihood of meaningful clustering within the data.

To begin this assessment, the first task involves creating a matrix of randomly uniformly distributed data. This matrix serves as the theoretical benchmark against which the actual dataset, standardized for empirical analysis, will be compared.

```
random_data <- apply(scaled_data,2, function(x){runif(length(x),min = min(x),max =
max(x))})
random_data <- as.data.frame(random_data)
random_data <- apply(random_data,2,scale)
pairs.panels(scaled_data, hist.col="skyblue", density = TRUE, ellipses = FALSE)
#observed data
```



```
pairs.panels(random_data, hist.col="violetred2", density = TRUE, ellipses = FALSE)
#randomly-distributed data
```



As observed from the comparison between the scatterplot matrices of the actual data and those of randomly uniformly distributed data, it is evident that the correlations in the former are higher in some cases compared to the latter. This discrepancy suggests that the first pair exhibits discernible variations, indicative of potential clustering tendencies.

The evaluation of cluster assessment tendency involves two formal methods, with the first and most reliable being the computation of the Hopkins statistic. This statistic measures the clustering tendency of a dataset by comparing the sum of distances between each data point and its nearest neighbor in the observed data to a benchmark set of randomly generated data. In mathematical terms, the Hopkins statistic ranges between 0 and 1, where values closer to 0 indicate a tendency towards clustering, values around 0.50 suggest no clear clustering structure, and values closer to 1 indicate a well-clustered structure.

To compute the Hopkins statistic in R, you need to install and load the "hopkins" package, which provides the function hopkins for this purpose.

```
hopkins(scaled_data) #we have clustered structure since the value is near 1

## [1] 0.9902523

hopkins(random_data) #there are no cluster in random data

## [1] 0.5566058
```

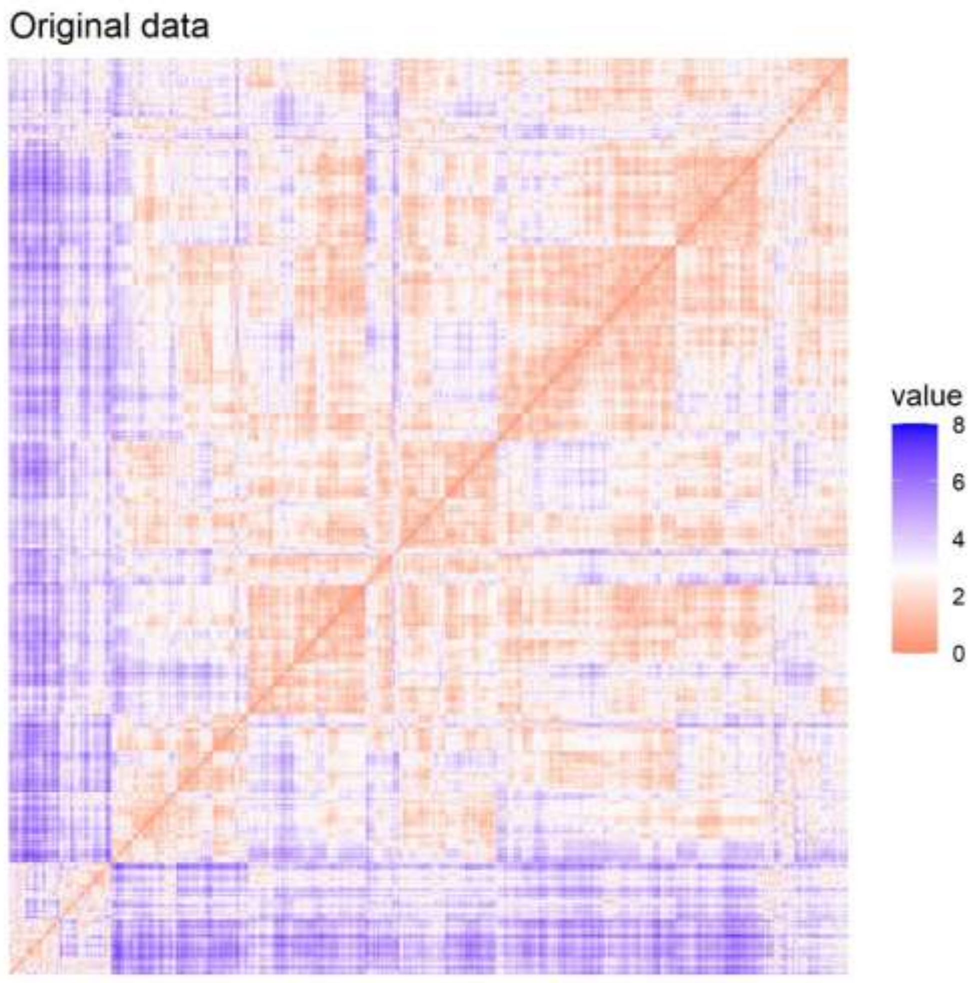
As the evidence shows, the closeness to 1 of the value of Hopkins statistic for scaled data confirms the clustering tendency of the observed data set, while the absence of clustering tendency is confirmed by the closeness of random data's Hopkins statistic to 0.50.

The VAT algorithm is a graphical method used to evaluate clustering tendency in a dataset. It involves computing the ordered dissimilarity matrix using the Euclidean distance measure between statistical units. The output of the VAT algorithm is an ordered dissimilarity image (ODI) where colors indicate similarity (red) or dissimilarity (blue) between data points.

To assess clustering tendency, the VAT algorithm generates ODI images for both the empirical dataset and a randomly generated dataset. By comparing these images and counting the number of red squares along the diagonal, analysts can visually determine the presence of clustering structure. More red squares along the diagonal suggest a stronger clustering tendency.

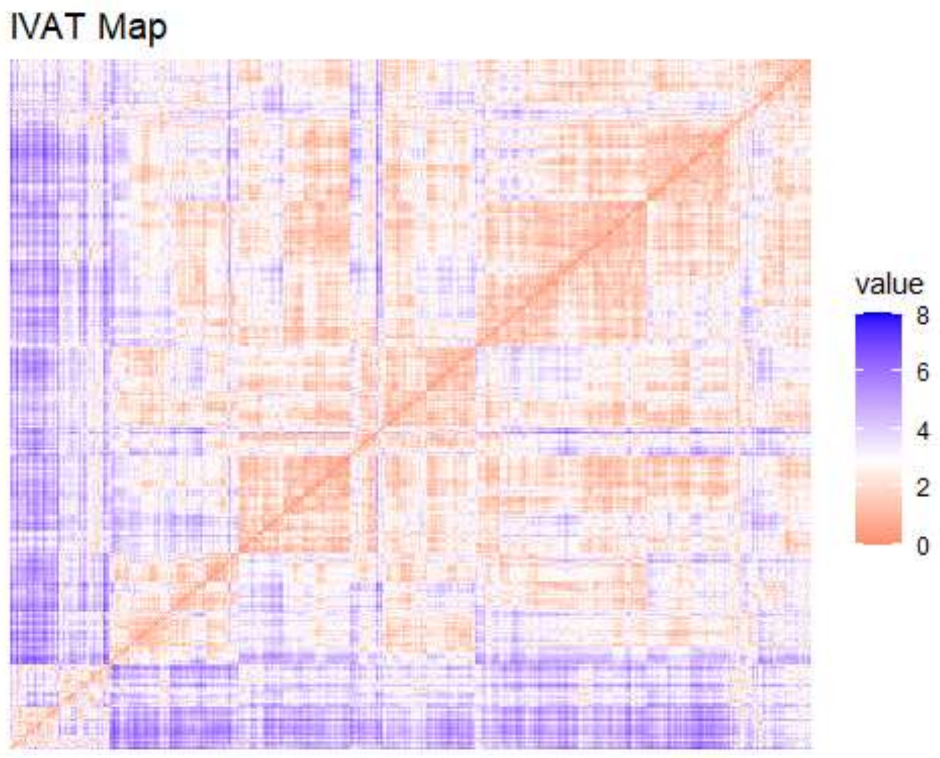
Implementing the VAT algorithm in R requires the installation of the "factoextra" package, which provides the fviz_dist function to visualize the ordered dissimilarity matrix.

```r
eucldist_scaled <- dist(scaled_data, method = "euclidean") # Creation of the distance matrix according to the Euclidean distance

fviz_dist(eucldist_scaled, show_labels = FALSE) + labs(title = "Original data")

head(eucldist_scaled)
```

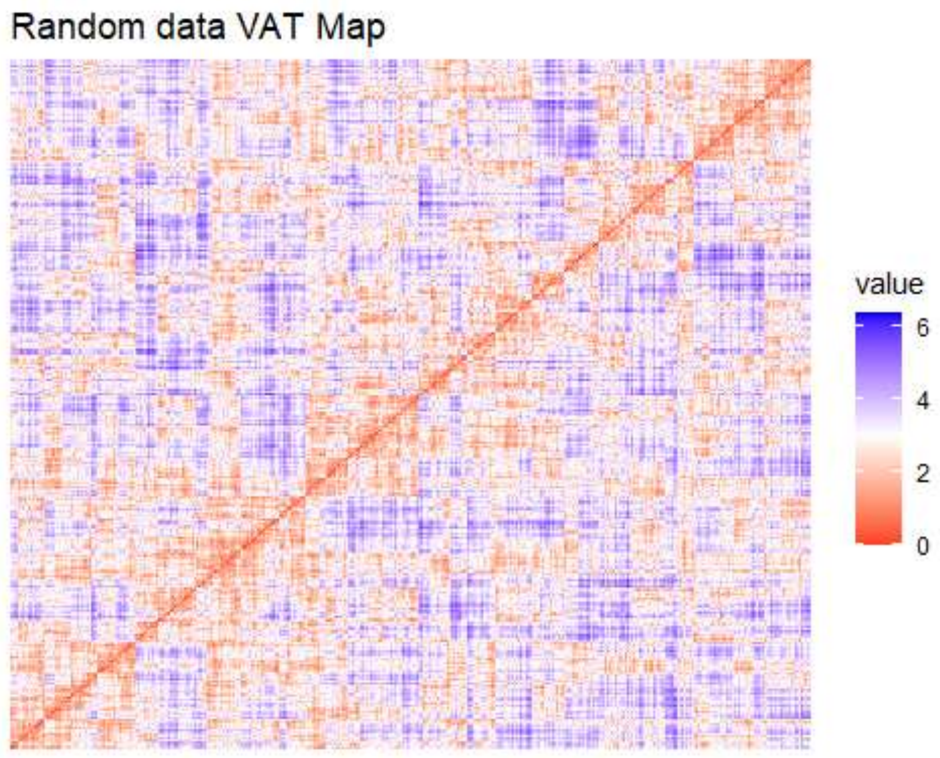


The IVAT (Interpolating Visual Assessment Tool) map enhances the VAT (Visual Assessment Tool) map by interpolating between dissimilarity matrix elements. This interpolation provides a smoother, more continuous representation of data structures, making it easier to identify clusters and hierarchical relationships, especially in complex datasets.

```
#library(seriation) to create an IVAT map, a better version of the previous plot
ivat_order <- seriate(eucldist_scaled, method = "VAT")
order <- get_order(ivat_order)
eucldist_scaled_ivat <- as.dist(as.matrix(eucldist_scaled)[order, order])
fviz_dist(eucldist_scaled_ivat, show_labels = FALSE) + labs(title = "IVAT Map")
```



```
fviz_dist(dist(random_data), show_labels = FALSE) + labs(title = "Random data VAT
Map")
```



As the graphical visualization shows, while random data's distance matrix clearly does not give space for any possible interpretation of the existence of a clustering tendency, the evidence is not so clear in the empirical data's one: it seems that a clustering structure is somehow present, but the amount of similarity within the statistical units is not so strong. That seems to be a bit contradictory with respect to what has been observed looking at the value assumed by the Hopkins statistic for the same set of data. However, a sign of existence of a clustering tendency seems to have been confirmed: the analysis can go on keeping in mind these results.

Choice of the number of clusters

Once the cluster tendency assessment has been established, the next step involves selecting the optimal number of clusters. The methods employed for clustering will include both hierarchical and partitional clustering methods. These methods utilize a hard clustering assignment approach and rely on specific distance measures between statistical units.

For distance measurement, the analysis prioritizes the Manhattan distance due to the presence of previously detected outliers, making it more robust compared to the Euclidean distance.

To implement hierarchical clustering methods in R, the installation of the "NbClust" package is required. This package facilitates the evaluation of various clustering criteria to determine the optimal number of clusters.

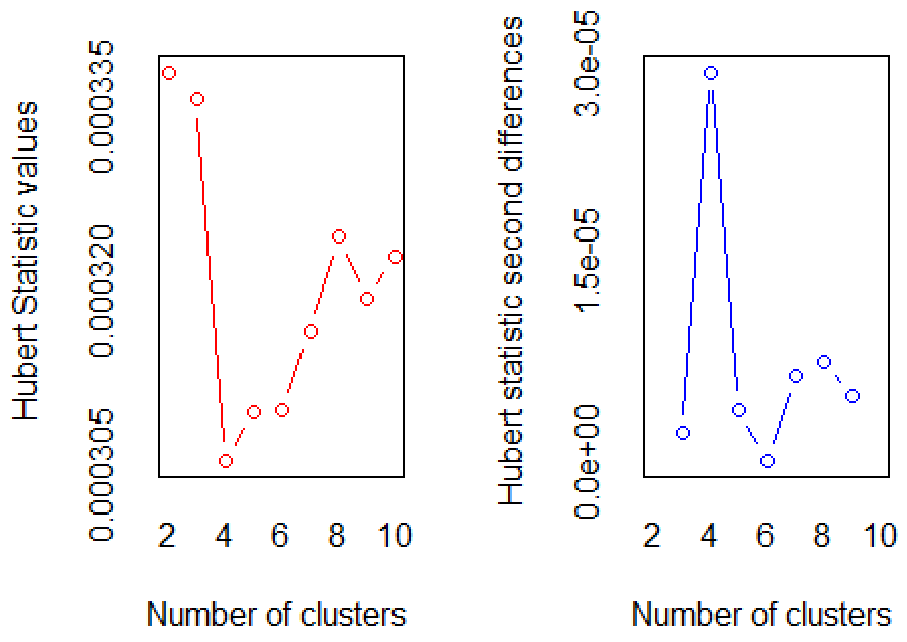
HC: Manhattan distance and single linkage criterion

The initial step essential for applying clustering techniques involves computing the distance matrix using the Manhattan distance measurement.

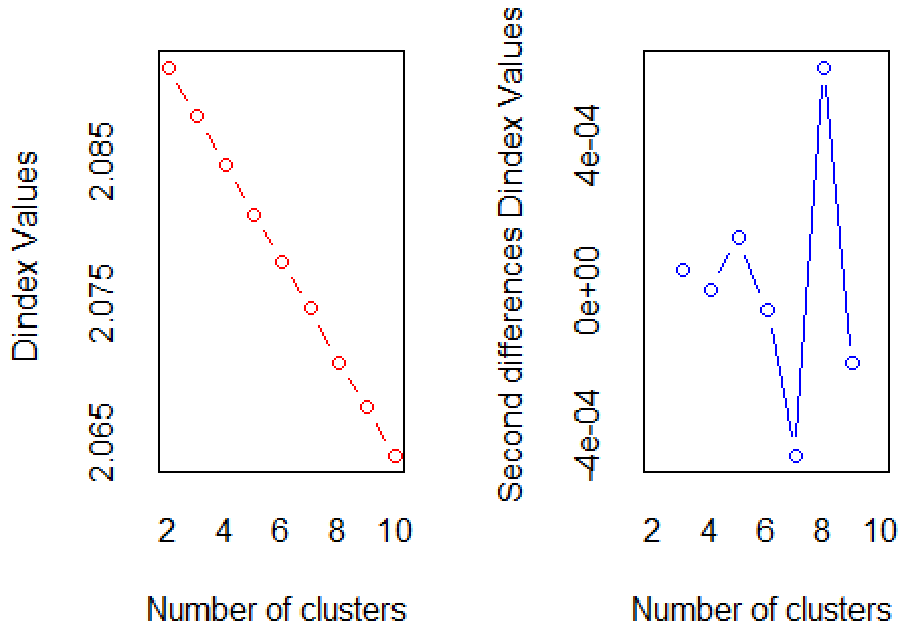
This distance is calculated as the absolute values of the distances between each statistical unit. This matrix serves as the foundation for subsequent clustering analyses, enabling the evaluation and grouping of statistical units based on their similarity according to the Manhattan distance metric.

The decision regarding the optimal number of clusters is informed by various selection criteria, such as the elbow method, silhouette width, or gap statistic. For the purpose of this analysis, the function NbClust (available in the "NbClust" package) will be utilized to provide insights from a broader range of cluster number evaluation criteria beyond just three.

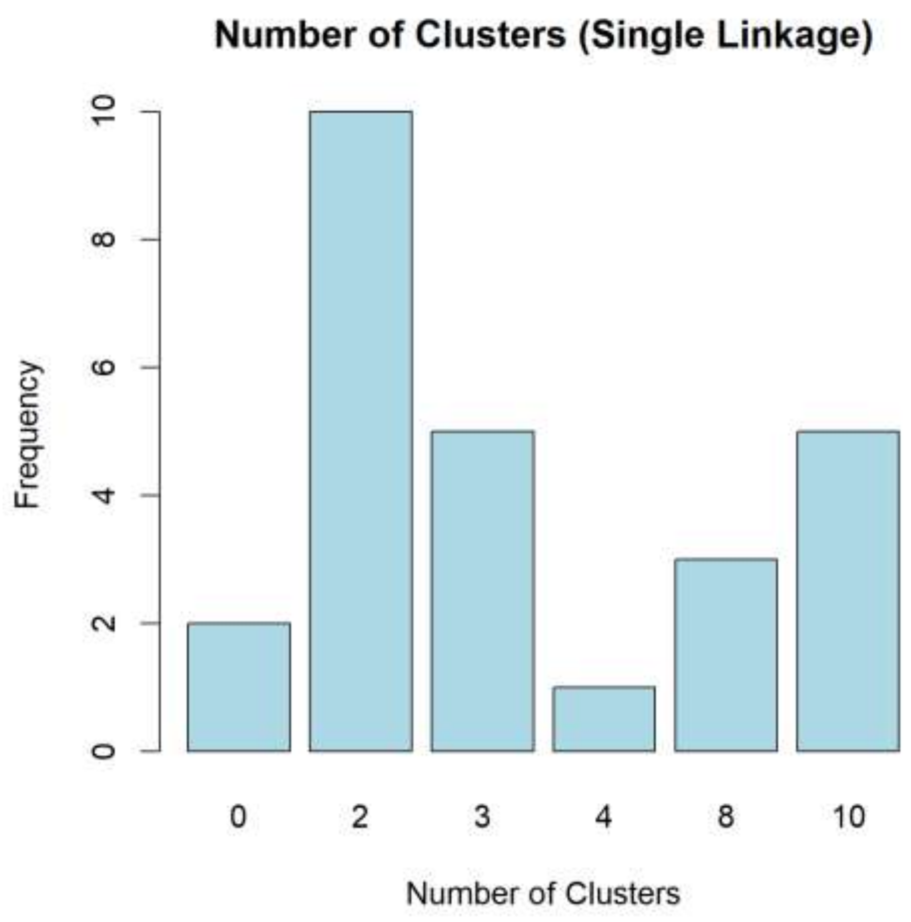
```
nb_single <- NbClust(scaled_data, distance = "manhattan",min.nc = 2, max.nc = 10,
method = "single") # Choose K
```



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



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

counts <- table(nb_single$Best.nc[1,])
barplot(counts,
        main = "Number of Clusters (Single Linkage)",
        xlab = "Number of Clusters",
        ylab = "Frequency",
        col = "lightblue")
```



Based on the evaluation criteria applied, the optimal number of clusters identified is two for the single linkage criterion. It is noteworthy that with a Hubert statistic value near zero, the clustering did not reveal a clear structure in the data.

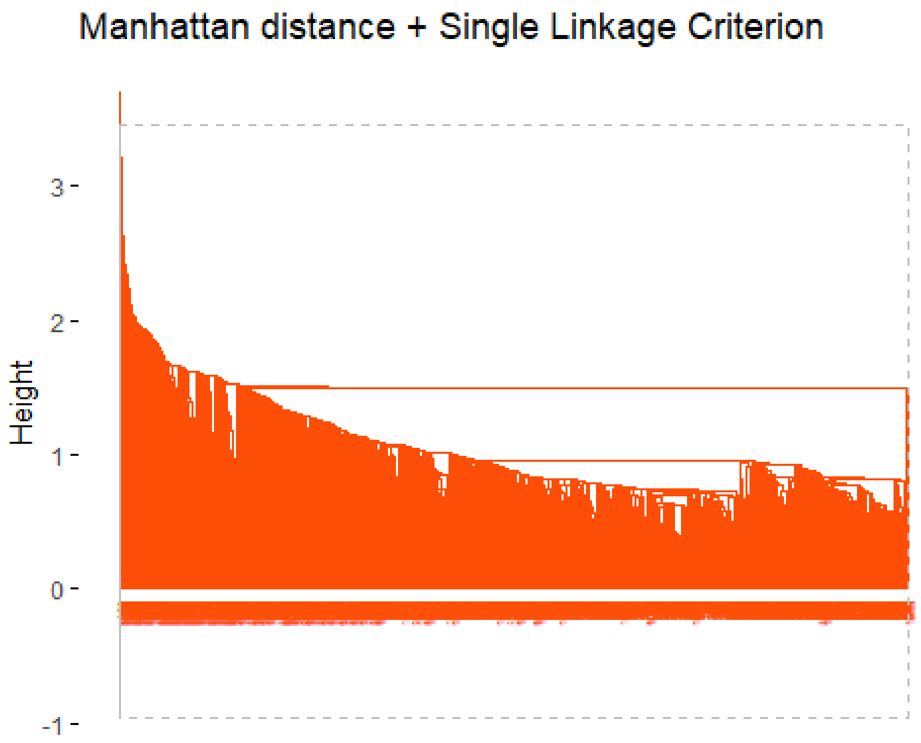
Subsequently, the hierarchical agglomerative clustering method will be applied to the distance matrix D using the Manhattan distance measure. The hclust function in R will facilitate this process, generating a dendrogram that visually represents the clustering structure. The dendrogram can be cut at the desired height corresponding to 2 clusters using the cutree function, with graphical representation facilitated by the fviz_dend function from the “factoextra” package.

```
manhdist_scaled <- dist(scaled_data, method = "manhattan")
manhdata_hc1 <- hclust(d = manhdist_scaled, method = "single")
manhdata1_tree <- cutree(manhdata_hc1, k = 2) # Identify clusters
table(manhdata1_tree)
```

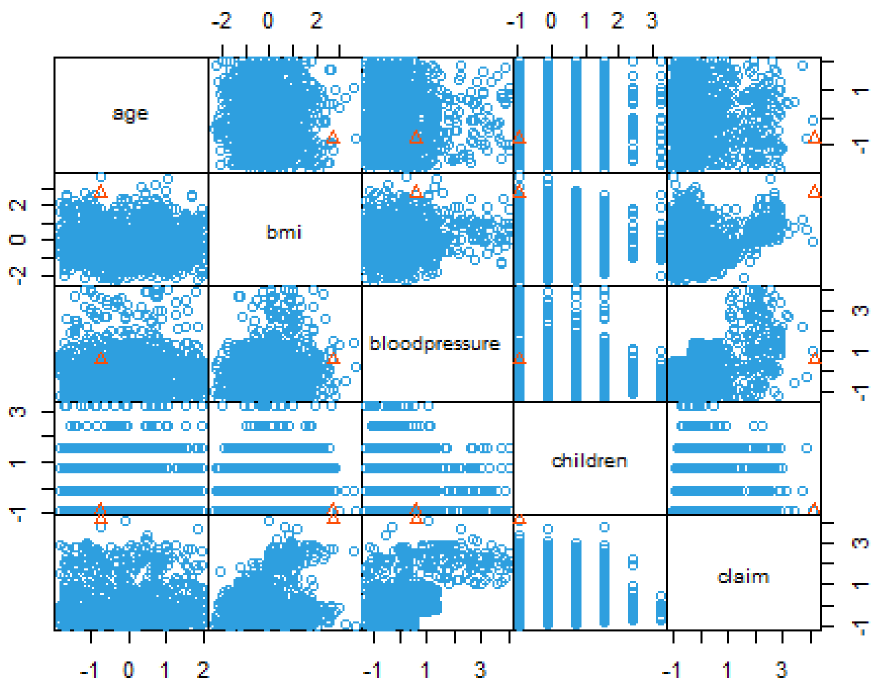
```
## manhdata1_tree
##    1    2
## 1331    1
```

```
fviz_dend(manhdata_hc1, k = 2, cex = 0.3,main = "Manhattan distance + Single
Linkage Criterion",k_colors = c("#2E9FDF", "#FC4E07"),rect = TRUE) # Dendrogram
```

```
## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none"
instead as
## of ggplot2 3.3.4.
## ℹ The deprecated feature was likely used in the factoextra package.
##   Please report the issue at <https://github.com/kassambara/factoextra/issues>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



```
pairs(scaled_data, gap = 0, pch = manhdata1_tree,col = c("#2E9FDF",
"#FC4E07")[manhdata1_tree])
```



The cluster assignment results indicate a clear differentiation between the two clusters, with one cluster containing all but one unit from the dataset, which is assigned to the second cluster.

In the final step of the hierarchical clustering method, the correlation between the original distance matrix and the "cophenetic distance" is computed. The cophenetic distance represents the similarity required for statistical units to be grouped into the same cluster during the agglomerative process, as visualized by the dendrogram's height. A higher correlation between the distance matrix and the cophenetic distance indicates greater efficiency of the clustering method, suggesting a higher degree of similarity among the statistical units.

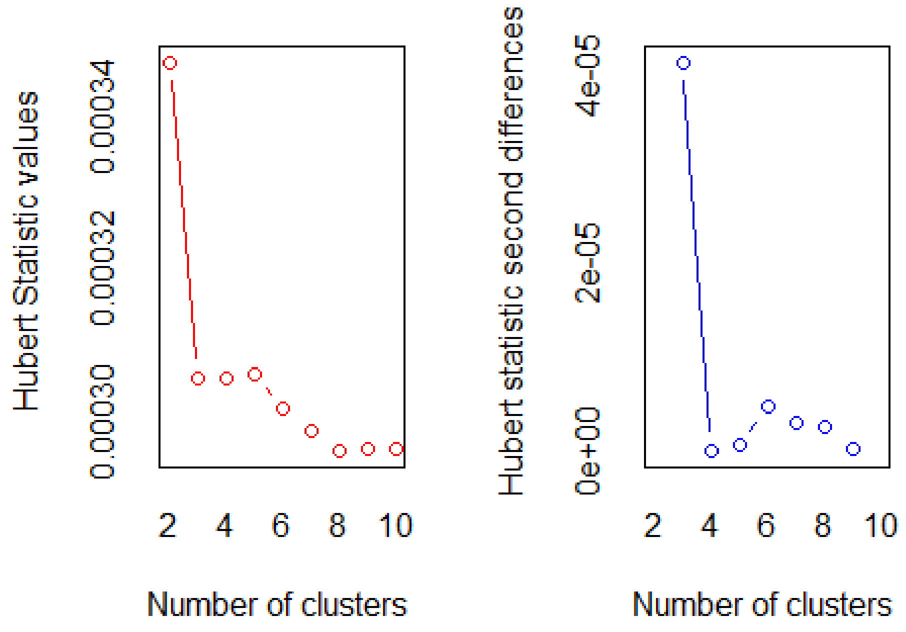
```
cor(manhdist_scaled, cophenetic(manhdata_hc1)) # Cophenetic distance

## [1] 0.6523218
```

In this case, the correlation is around 65%. The clustering method shows inefficiency, attributed to the chain-up effect caused by the single linkage criterion. Exploring alternative clustering methods is advisable.

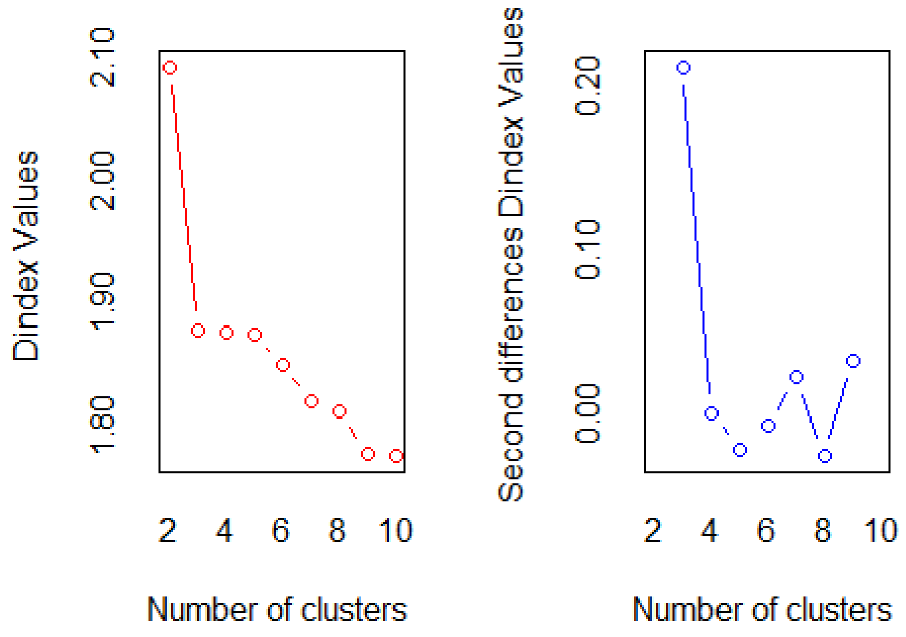
In the next steps, we will change the linkage criterion.

HC: Manhattan distance and average linkage criterion

```
nb_average <- NbClust(scaled_data, distance = "manhattan",min.nc = 2, max.nc = 10,
method = "average") # Choose K
```

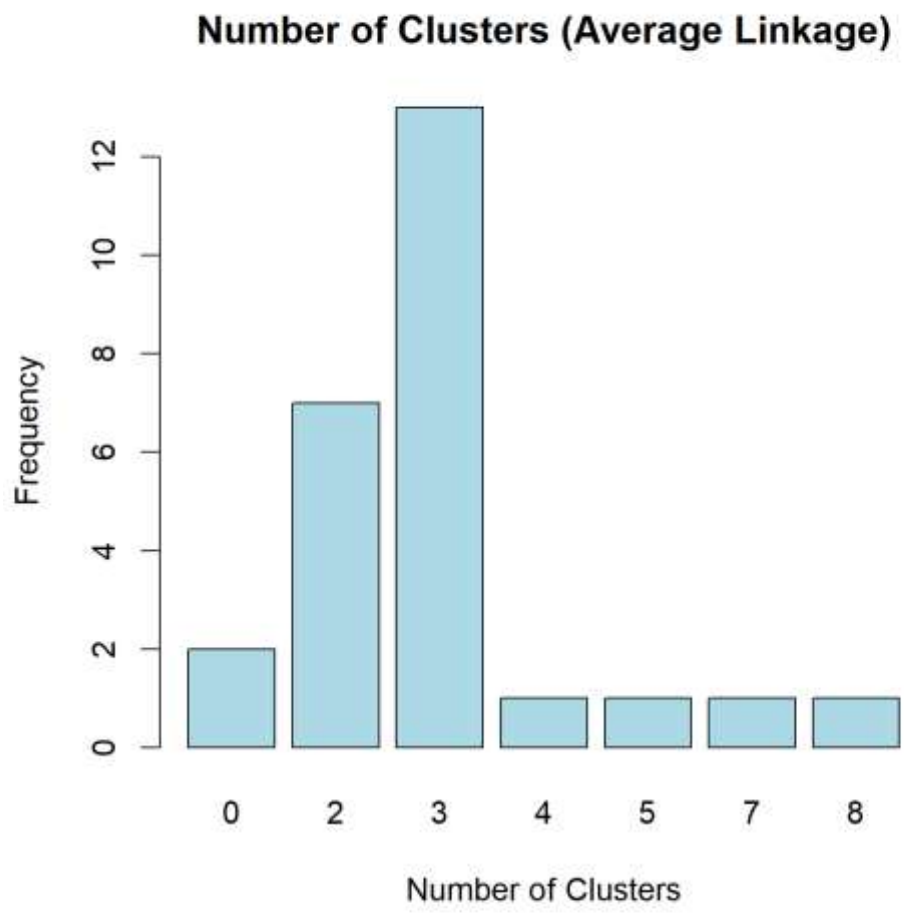


```
## *** : The Hubert index is a graphical method of determining the number of
clusters.
##                In the plot of Hubert index, we seek a significant knee that
corresponds to a
##                significant increase of the value of the measure i.e the
significant peak in Hubert
##         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



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

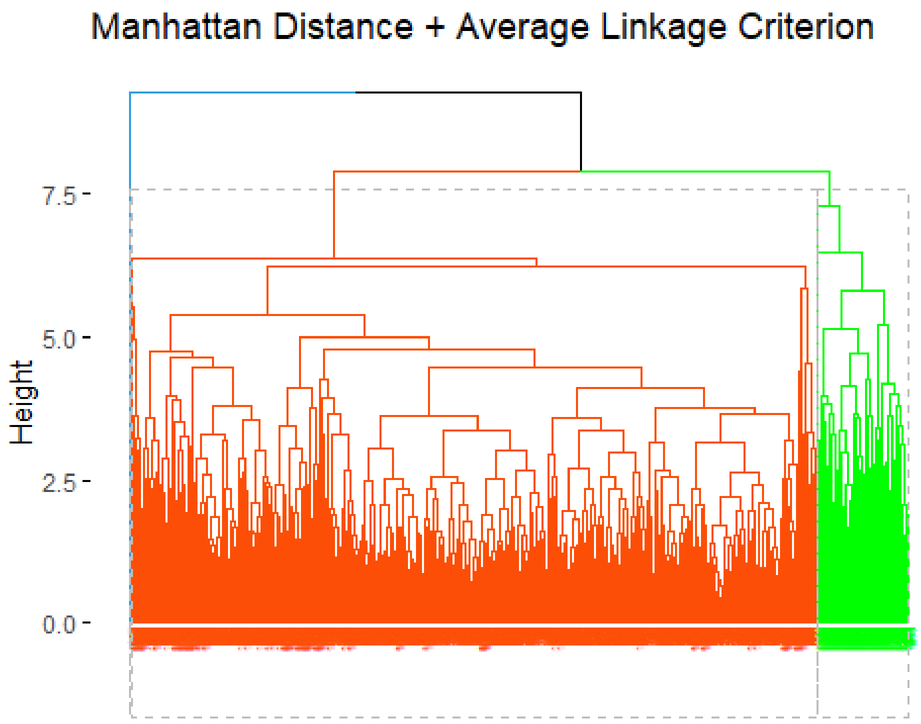
```
counts <- table(nb_average$Best.nc[1,])

barplot(counts,main = "Number of Clusters (Average Linkage)",xlab = "Number of
Clusters",ylab = "Frequency",col = "lightblue") # Barplot
```



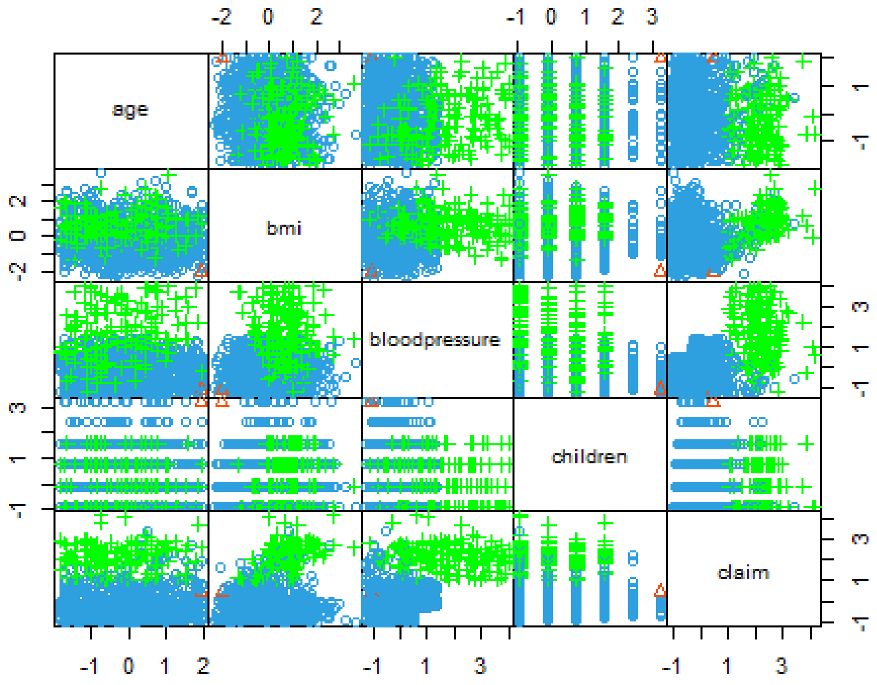
Based on the evaluation criteria applied, the optimal number of clusters identified is 3 for the average linkage criterion.

Next, we apply the agglomerative clustering algorithm.

```
manhdata_hc2 <- hclust(d = manhdist_scaled, method = "average")
manhdata2_tree <- cutree(manhdata_hc2, k = 3) # Identify clusters
table(manhdata2_tree)

## manhdata2_tree
##    1    2    3
## 1174    1  157

fviz_dend(manhdata_hc2, k = 3, cex = 0.3,
          main = "Manhattan Distance + Average Linkage Criterion",
          k_colors = c("#2E9FDF", "#FC4E07", "green"),
          rect = TRUE) # Dendrogram
```



```
pairs(scaled_data, gap = 0, pch = manhdata2_tree,
      col = c("#2E9FDF", "#FC4E07","green")[manhdata2_tree])
```



The cluster assignment results indicate a clear differentiation between the three clusters, with one cluster containing 1174 units, another containing 1 unit, and the third containing 157 units from the dataset.

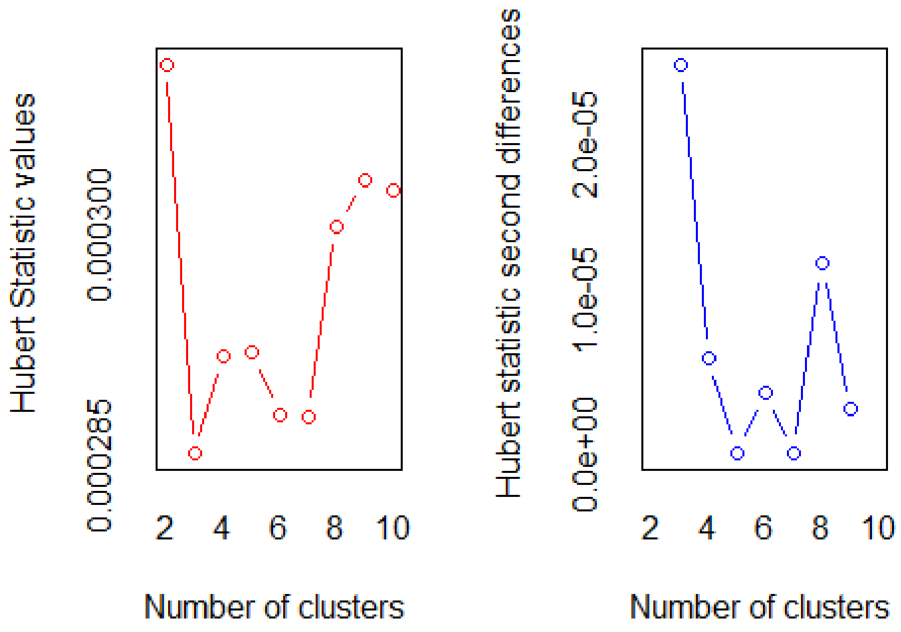
Next, the correlation between the original distance matrix and the “cophenetic distance” is computed.

```
cor(manhdist_scaled, cophenetic(manhdata_hc2)) # Cophenetic distance

## [1] 0.6914327
```

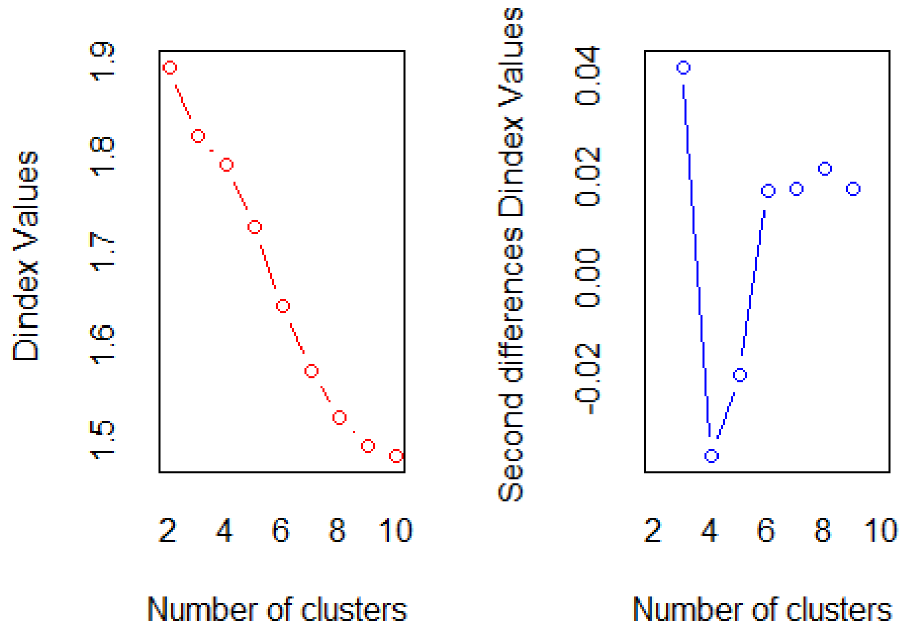
In this case, the correlation is around 69%. The clustering method is not so efficient.

HC: Manhattan distance and complete linkage criterion

```
nb_complete <- NbClust(scaled_data, distance = "manhattan",min.nc = 2, max.nc =
10, method = "complete") # Choose K
```

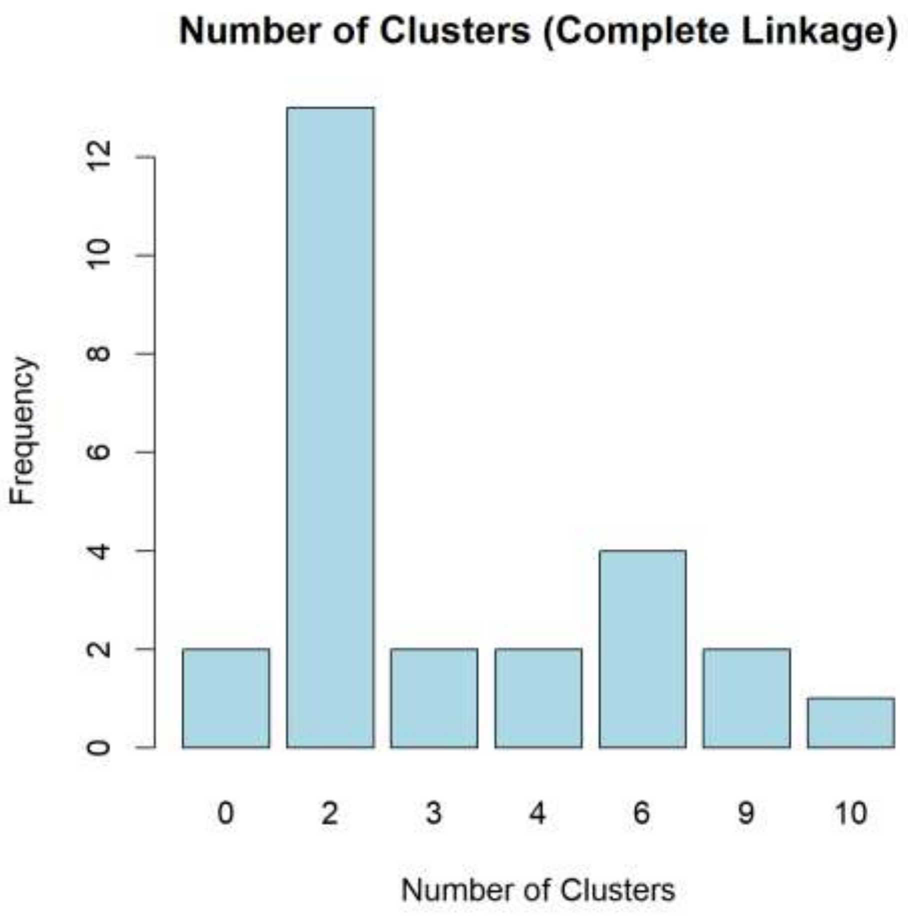


```
## *** : The Hubert index is a graphical method of determining the number of
clusters.
##                In the plot of Hubert index, we seek a significant knee that
corresponds to a
##                significant increase of the value of the measure i.e the
significant peak in Hubert
##         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



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

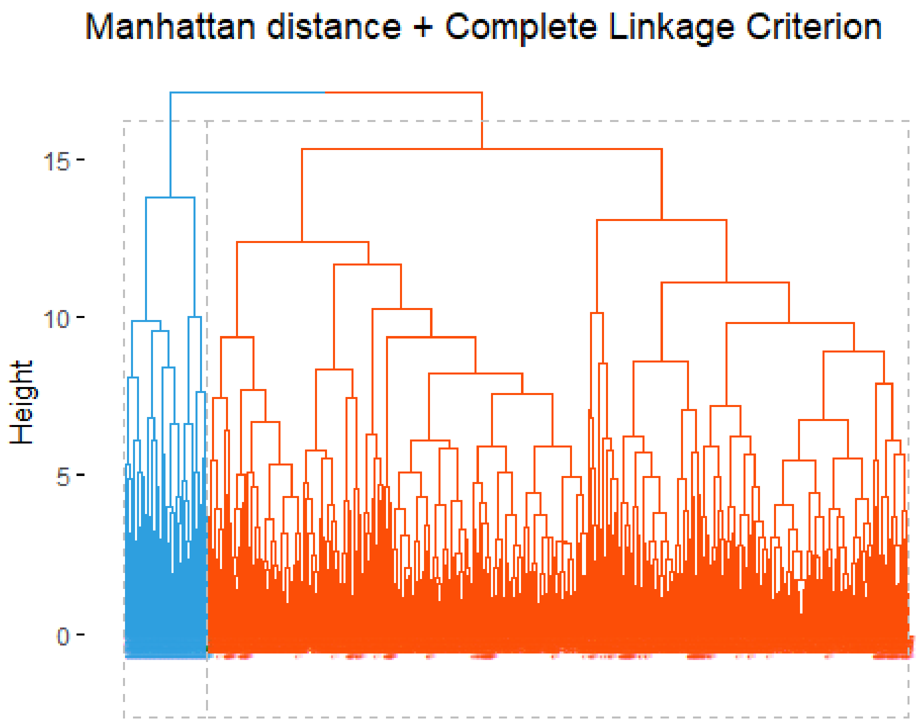
```
counts <- table(nb_complete$Best.nc[1,])

barplot(counts,main = "Number of Clusters (Complete Linkage)",xlab = "Number of
Clusters",ylab = "Frequency",col = "lightblue") # Barplot
```



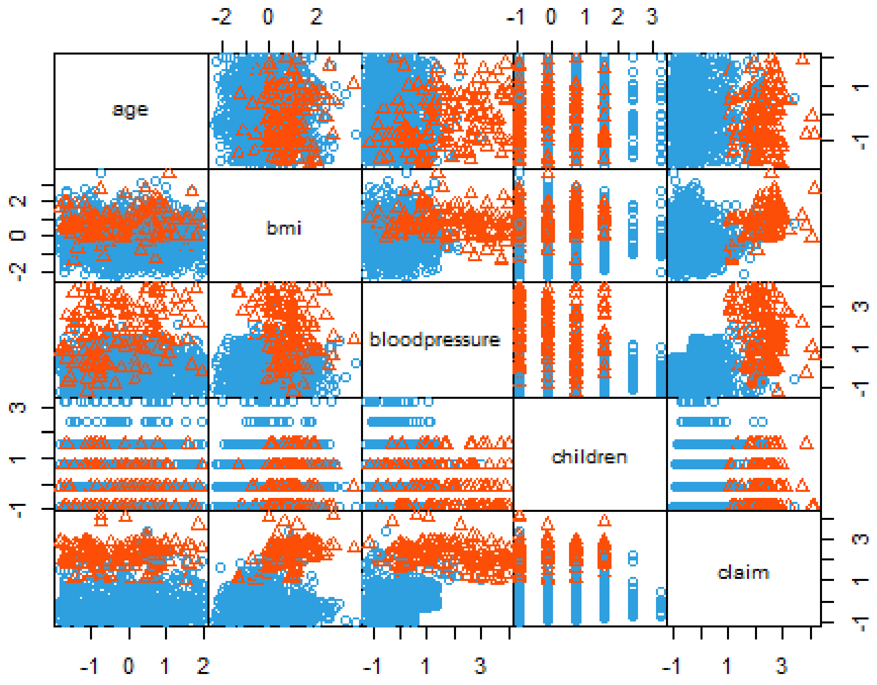
Based on the evaluation criteria applied, the optimal number of clusters identified is 2 for the complete linkage criterion.

Next, we apply the agglomerative clustering algorithm.

```
manhdata_hc3 <- hclust(d = manhdist_scaled, method = "complete")
manhdata3_tree <- cutree(manhdata_hc3, k = 2) # Identify clusters
table(manhdata3_tree)

## manhdata3_tree
##    1    2
## 1191  141

fviz_dend(manhdata_hc3, k = 2, cex = 0.3,
          main = "Manhattan distance + Complete Linkage Criterion",
          k_colors = c("#2E9FDF", "#FC4E07"),
          rect = TRUE) # Dendrogram
```



```
pairs(scaled_data, gap = 0, pch = manhdata3_tree,
      col = c("#2E9FDF", "#FC4E07")[manhdata3_tree])
```



The cluster assignment results indicate a clear differentiation between the two clusters, with one cluster containing 1191 units and the other containing 141 units from the dataset.

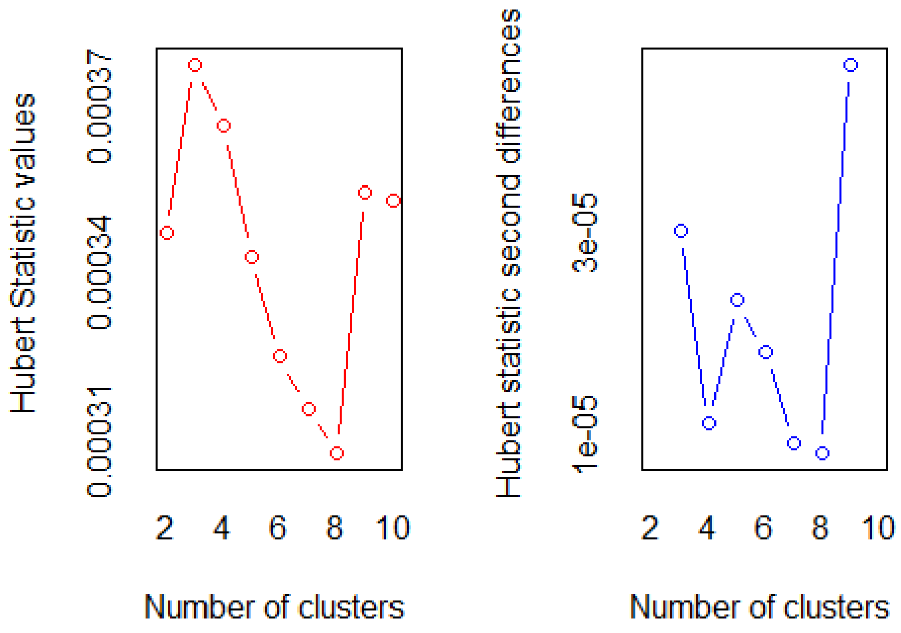
Next, the correlation between the original distance matrix and the "cophenetic distance" is computed.

```
cor(manhdist_scaled, cophenetic(manhdata_hc3)) # Cophenetic distance

## [1] 0.4924629
```

In this case, the correlation is around 49%. The clustering method is not efficient.

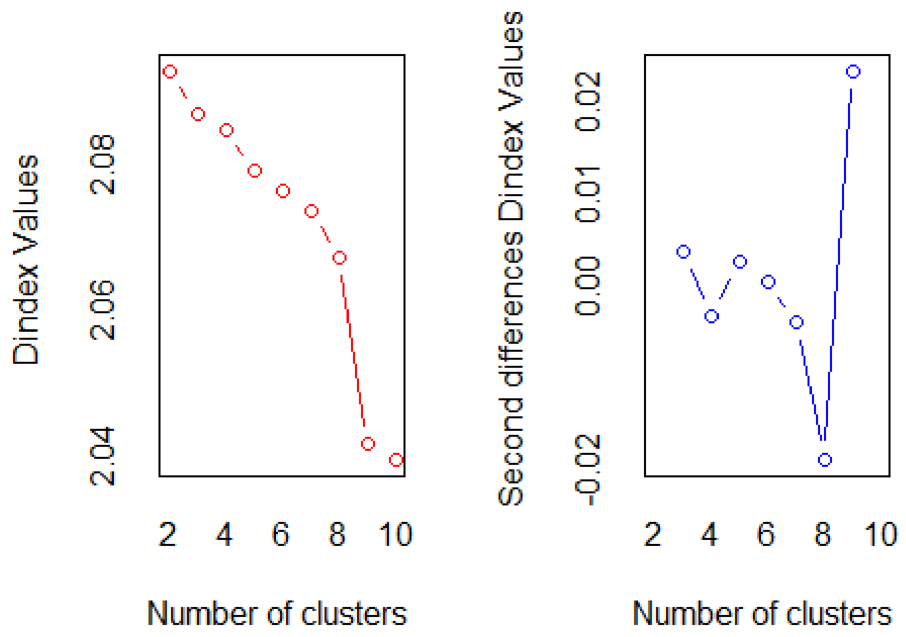
HC: Manhattan distance and centroid linkage criterion

```
nb_centroid <- NbClust(scaled_data, distance = "manhattan",min.nc = 2, max.nc =
10, method = "centroid") # Choose K

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

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



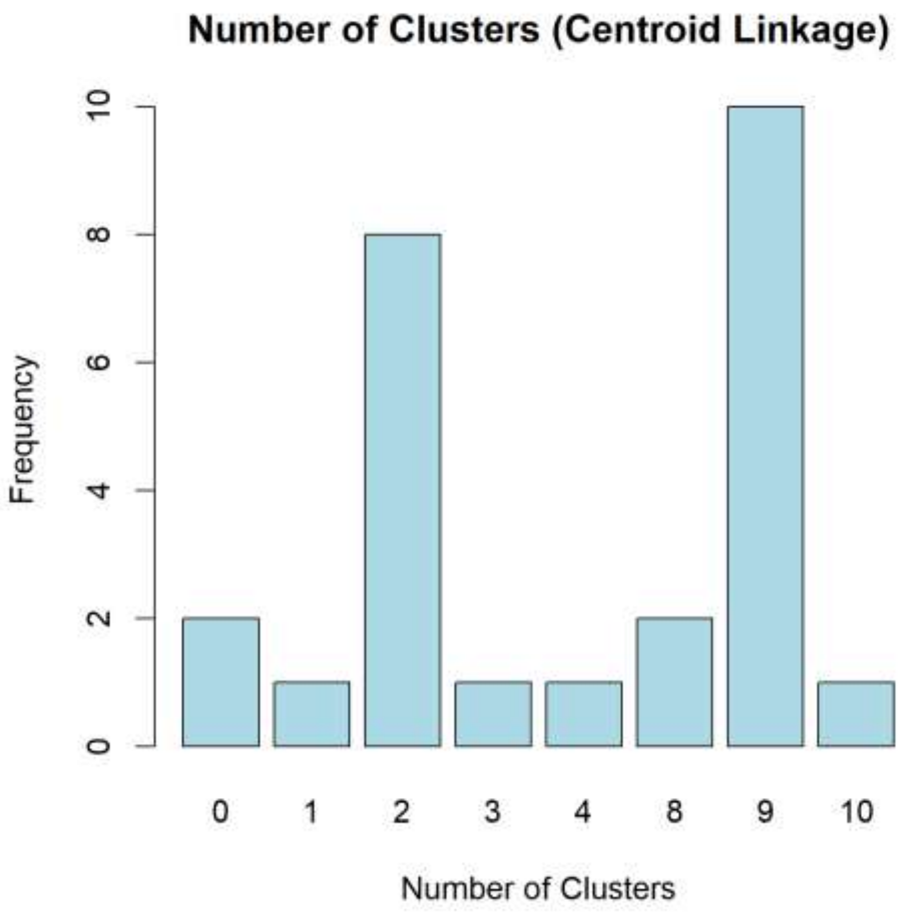
```
## *** : The Hubert index is a graphical method of determining the number of
clusters.
##                In the plot of Hubert index, we seek a significant knee that
corresponds to a
##                significant increase of the value of the measure i.e the
significant peak in Hubert
```

```
##                    index second differences plot.
##
```



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

counts <- table(nb_centroid$Best.nc[1,])
```

```
barplot(counts,main = "Number of Clusters (Centroid Linkage)",xlab = "Number of
Clusters",ylab = "Frequency",col = "lightblue") # Barplot
```

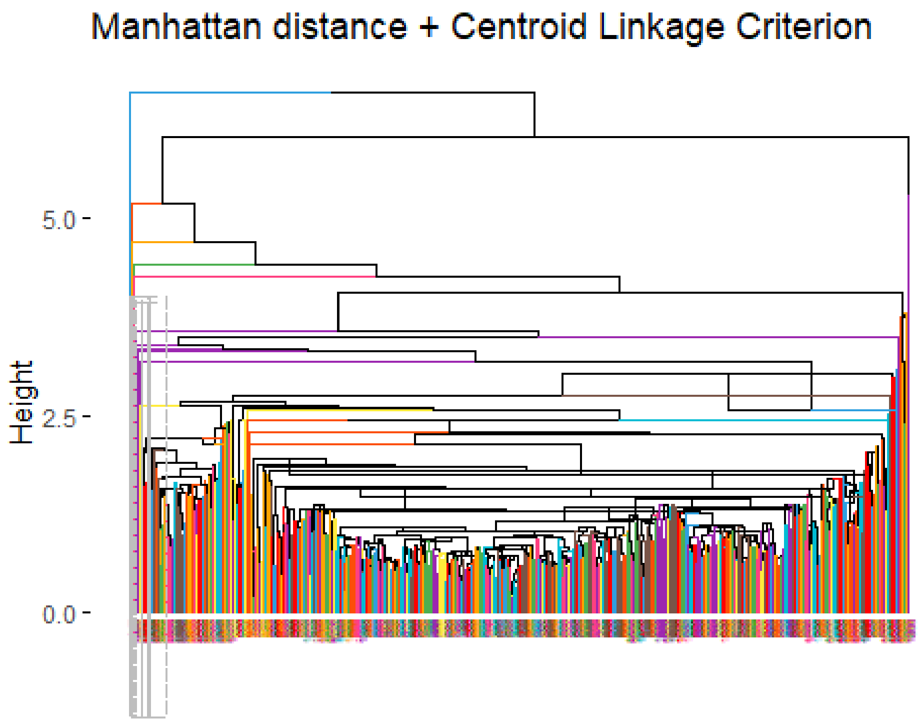


Based on the evaluation criteria applied, the optimal number of clusters identified is 9 for the centroid linkage criterion.

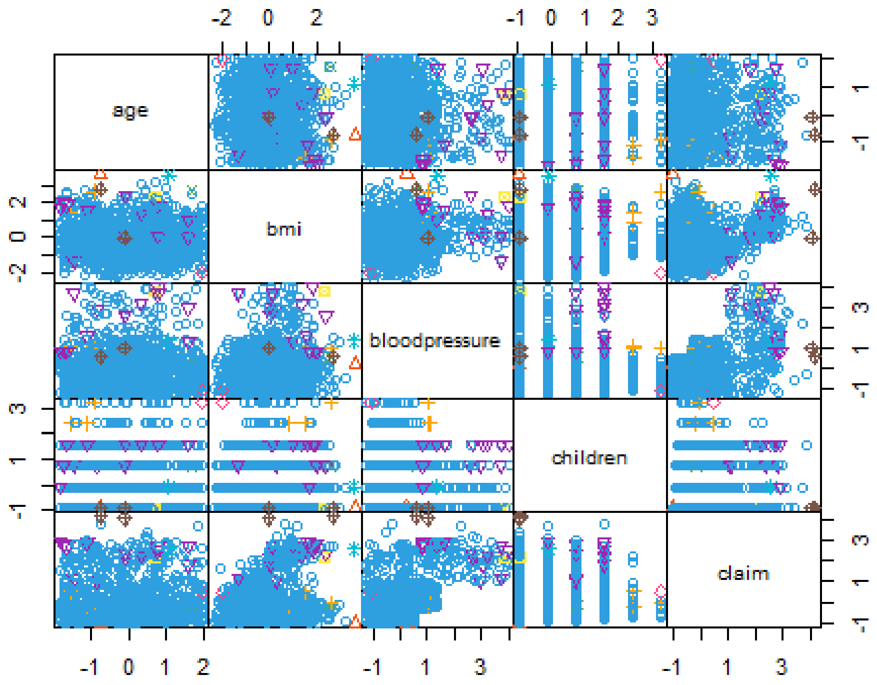
Next, we apply the agglomerative clustering algorithm.

```
manhdata_hc4 <- hclust(d = manhdist_scaled, method = "centroid")
manhdata4_tree <- cutree(manhdata_hc4, k = 9) # Identify clusters
table(manhdata4_tree)

## manhdata4_tree
##    1    2    3    4    5    6    7    8    9
## 1309    1    3    1    1   13    1    1    2

fviz_dend(manhdata_hc4, k = 9, cex = 0.3,
          main = "Manhattan distance + Centroid Linkage Criterion",
          k_colors = c("#2E9FDF", "#FC4E07", "#FFA500", "#4CAF50", "#FF4081",
"#9C27B0", "#FFEB3B", "#00BCD4", "#795548","red"),
          rect = TRUE) # Dendrogram
```

```
## Warning in get_col(col, k): Length of color vector was shorter than the number
## of clusters - color vector was recycled

## Warning in data.frame(xmin = unlist(xleft), ymin = unlist(ybottom), xmax =
## unlist(xright), : row.names ricavati da una variabile con pochi elementi e
## quindi non utlizzabili
```



```
pairs(scaled_data, gap = 0, pch = manhdata4_tree,
      col = c("#2E9FDF", "#FC4E07", "#FFA500", "#4CAF50", "#FF4081", "#9C27B0",
"#FFEB3B", "#00BCD4", "#795548")[manhdata4_tree])
```



The cluster assignment results indicate a differentiation between the nine clusters, with one cluster containing 1309 units, and the others containing fewer units distributed across the remaining clusters.

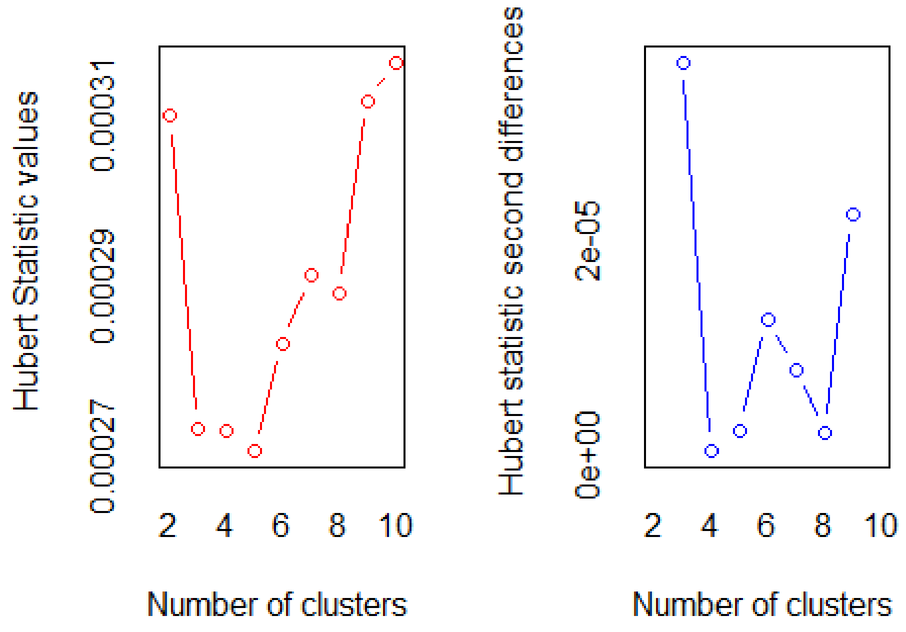
Next, the correlation between the original distance matrix and the "cophenetic distance" is computed.

```
cor(manhdist_scaled, cophenetic(manhdata_hc4)) # Cophenetic distance

## [1] 0.6430719
```

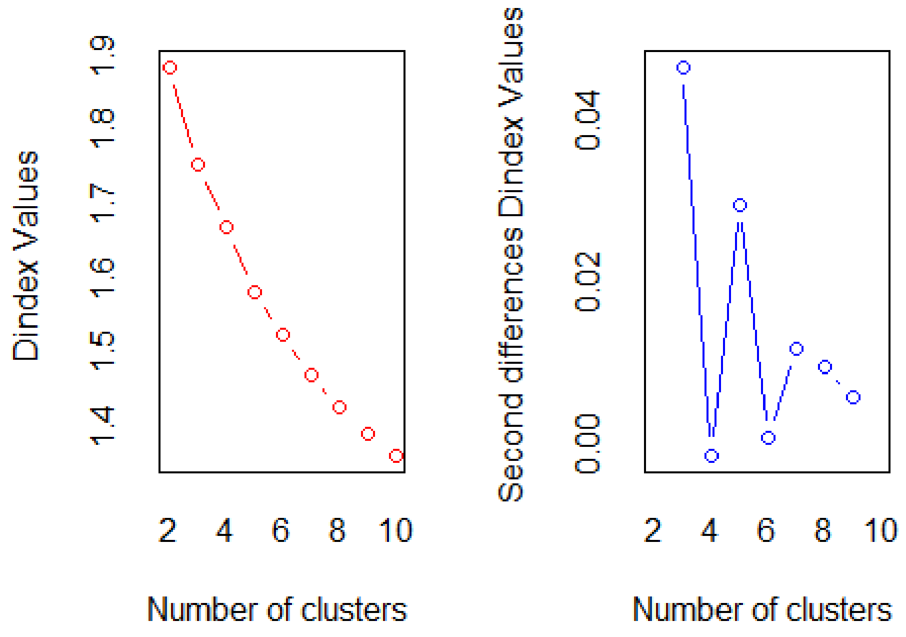
In this case, the correlation is around 64%. The clustering method is not so efficient, even though the correlation is not so low. It should be better to look for other ways.

HC: Manhattan distance and Ward's distance method

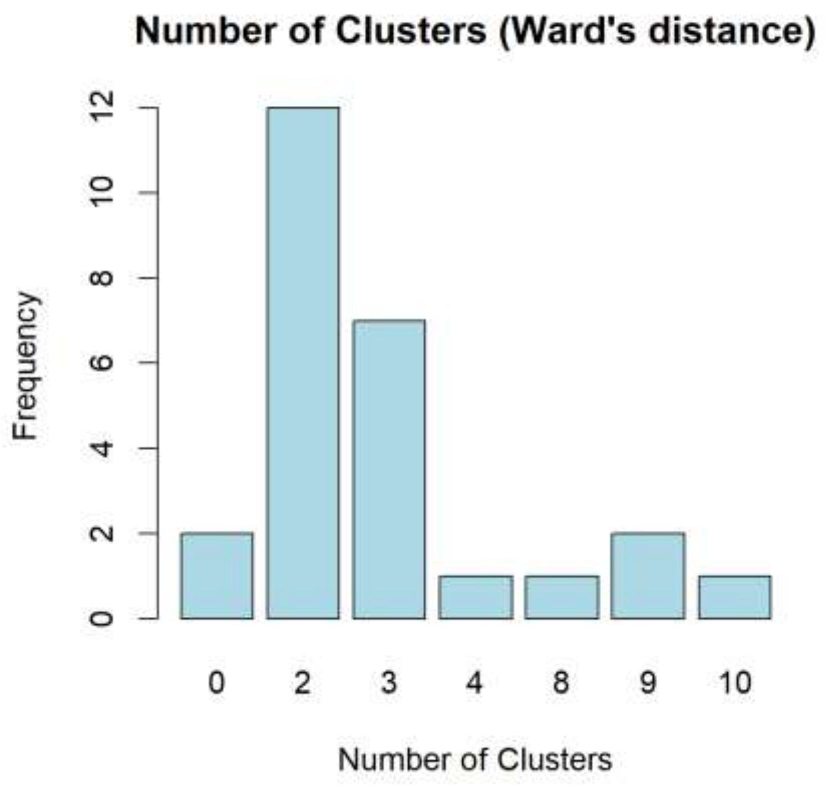
```
nb_ward <- NbClust(scaled_data, distance = "manhattan",min.nc = 2, max.nc = 10,
method = "ward.D2") # Choose K
```



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



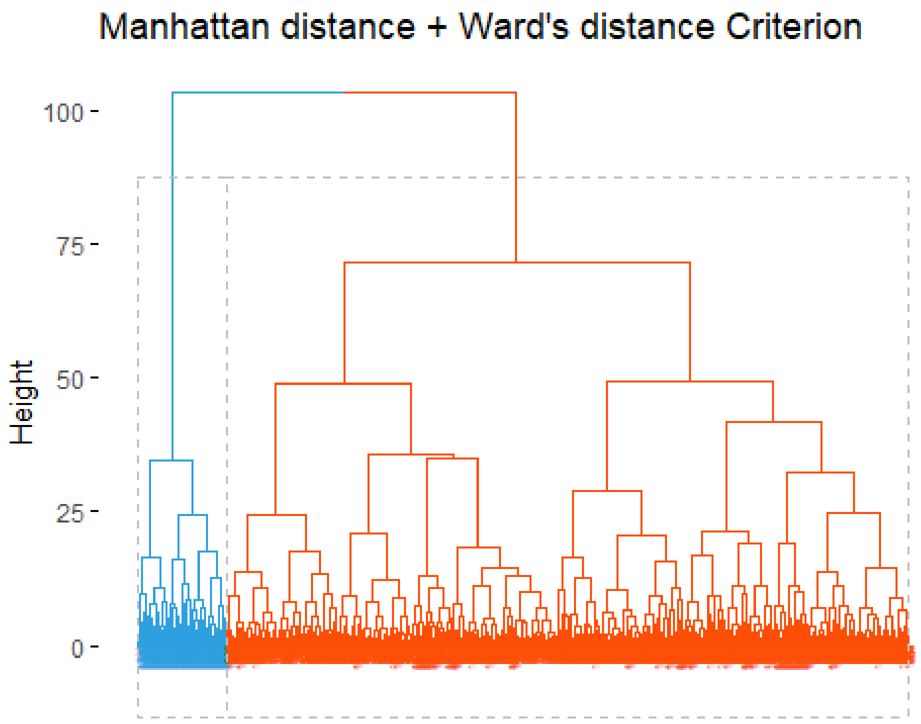
```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the
significant peak in Dindex
##                 second differences plot) that corresponds to a significant
increase of the value of
##       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

counts <- table(nb_ward$Best.nc[1,])

barplot(counts,main = "Number of Clusters (Ward's distance)",xlab = "Number of
Clusters",ylab = "Frequency",col = "lightblue") # Barplot
```



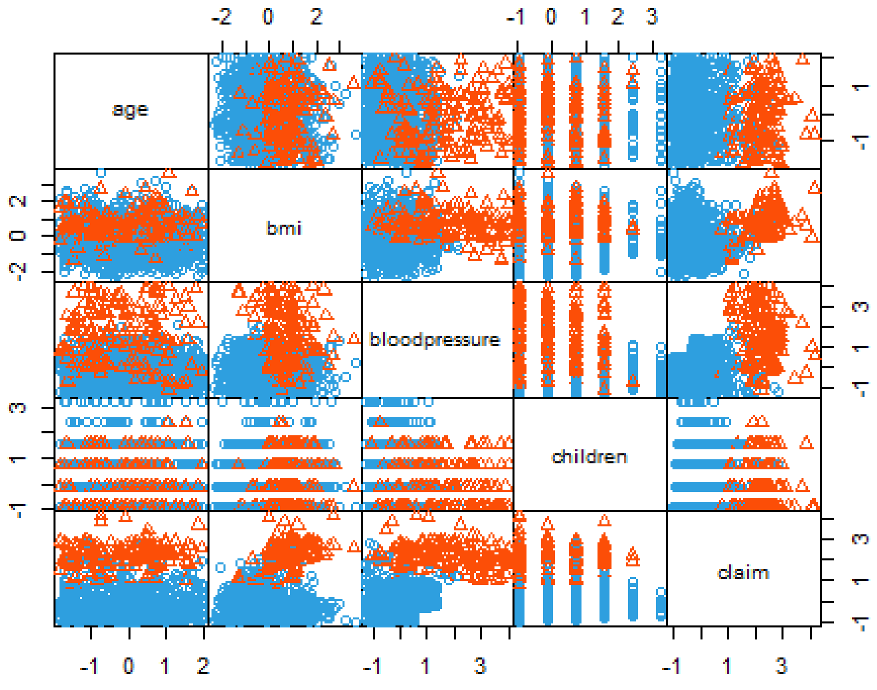
Based on the evaluation criteria applied, the optimal number of clusters identified is 2 for the Wards's distance criterion.

Next, we apply the agglomerative clustering algorithm.

```
manhdata_hc5 <- hclust(d = manhdist_scaled, method = "ward.D2")
manhdata5_tree <- cutree(manhdata_hc5, k = 2) # Identify clusters
table(manhdata5_tree)

## manhdata5_tree
##    1    2
## 1179  153

fviz_dend(manhdata_hc5, k = 2, cex = 0.3,
          main = "Manhattan distance + Ward's distance Criterion",
          k_colors = c("#2E9FDF", "#FC4E07"),
          rect = TRUE) # Dendrogram
```



```
pairs(scaled_data, gap = 0, pch = manhdata5_tree,
      col = c("#2E9FDF", "#FC4E07")[manhdata5_tree])
```



The cluster assignment results indicate a clear differentiation between the two clusters, with one cluster containing 1179 units, and the second cluster containing 153 units from the dataset.

Next, the correlation between the original distance matrix and the "cophenetic distance" is computed.

```r
cor(manhdist_scaled, cophenetic(manhdata_hc5)) # Cophenetic distance

## [1] 0.5992468
```

In this case, the correlation is around 60%. The clustering method is not so efficient, even though the correlation is not so low. It should be better to look for other ways.

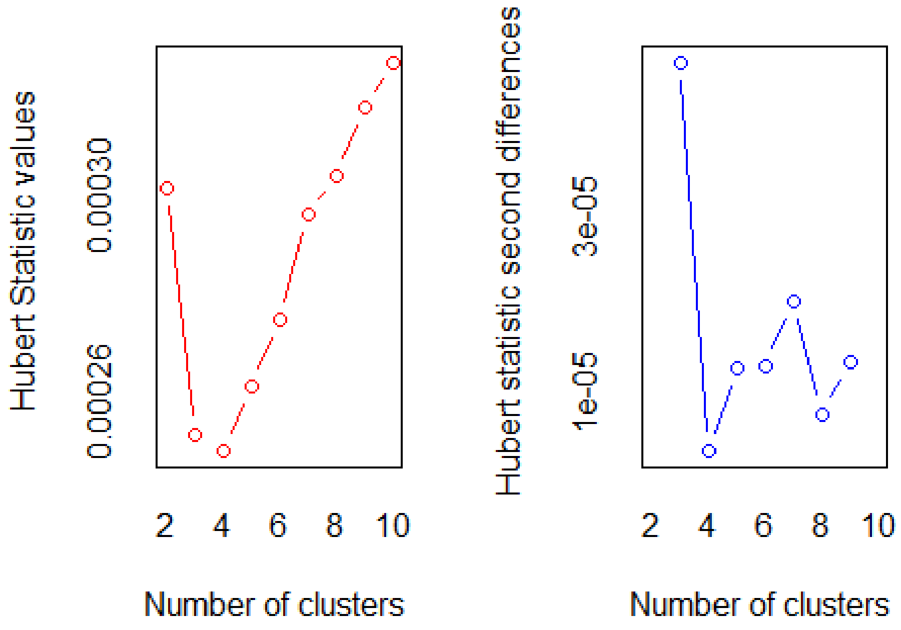
Partitional clustering: K-means

Partitional clustering methods, such as K-means and K-medoids, offer an alternative to hierarchical clustering, addressing its computational limitations. Unlike hierarchical clustering, partitional methods require the specification of the desired number of clusters before execution. These methods generate the optimal partition of units within clusters randomly, necessitating the setting of a seed to ensure reproducibility of results.

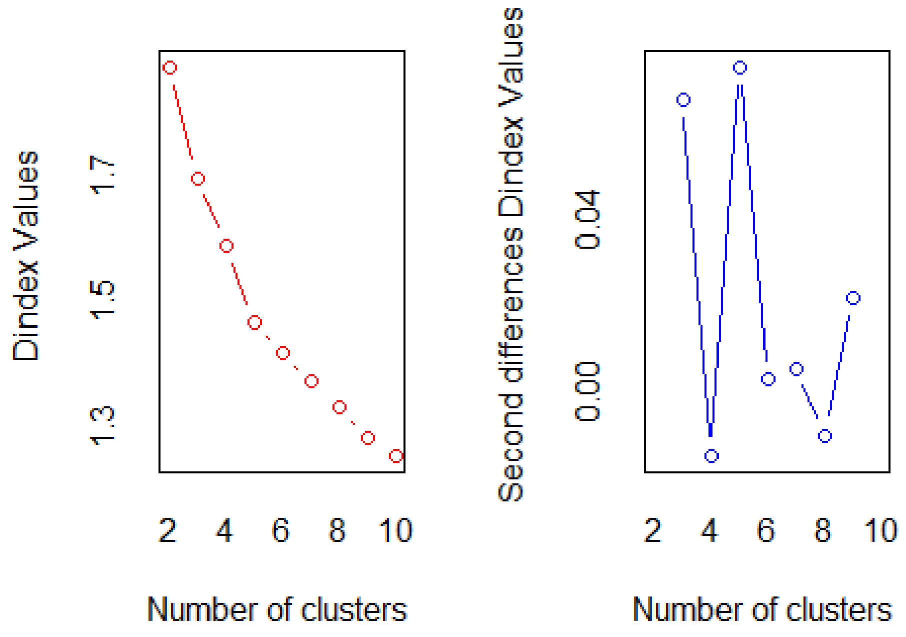
K-means clustering, one of the prominent partitional methods, merges statistical units into clusters by using the arithmetic mean of the clusters as the criterion for merging, continuing until the predefined number of clusters is achieved. Implementation of the K-means algorithm in R involves the "stats" package and requires setting an initial seed to make the outcomes replicable, as the algorithm begins with K randomly chosen units serving as temporary centroids.

Similar to hierarchical methods, determining the optimal number of clusters is a crucial preliminary step in partitional clustering.

```r
nb_kmeans <- NbClust(scaled_data, distance = "manhattan",min.nc = 2, max.nc = 10,
method = "kmeans") # Choose K
```



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



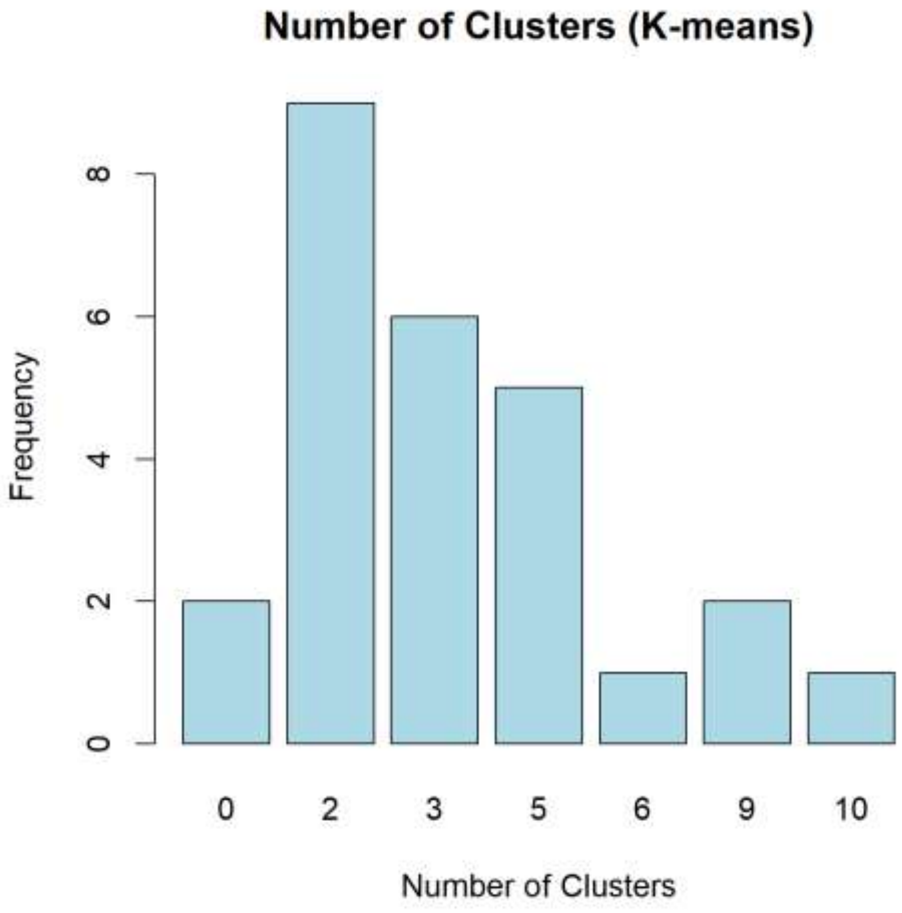
```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the
significant peak in Dindex
##                 second differences plot) that corresponds to a significant
increase of the value of
##       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

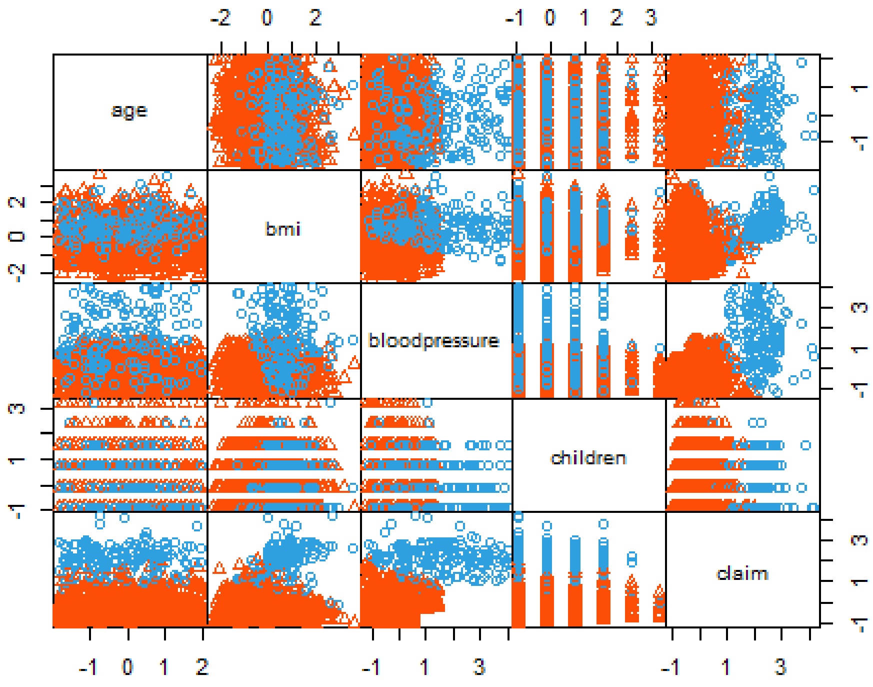
```
counts <- table(nb_kmeans$Best.nc[1,])
barplot(counts,main = "Number of Clusters (K-means)",xlab = "Number of
Clusters",ylab = "Frequency",col = "lightblue") # Barplot
set.seed(12345)
```

```r
kmeans_r <- kmeans(scaled_data, centers = 2, nstart = 25)
kmeans_r$size
```

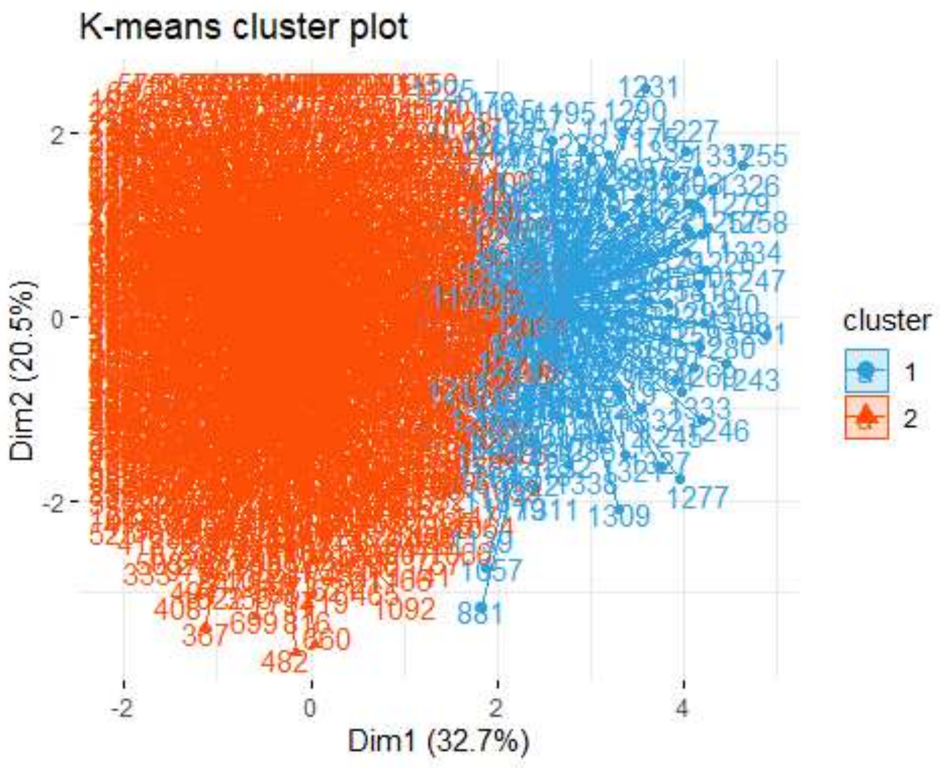
```
## [1]  182 1150
```

```r
pairs(scaled_data, gap = 0, pch = kmeans_r$cluster, col = c("#2E9FDF",
"#FC4E07")[kmeans_r$cluster])
```


Number of Clusters (K-means)
Frequency
0
2
4
6
8
0
2
3
5
6
9
10
Number of Clusters



```
# Visualize the K-means cluster plot
fviz_cluster(kmeans_r, data = scaled_data,
             palette = c("#2E9FDF", "#FC4E07"),
             ellipse.type = "euclid", star.plot = TRUE,
             repel = TRUE, ggtheme = theme_minimal(),
             main = "K-means cluster plot")
```



According to the K-means algorithm, the best partition of the statistical units into 2 clusters is given by the assignment of 1,150 units to the first cluster and 182 to the second one.

In order to achieve a clearer visualization of the clusters, we utilized the space defined by the first two principal components identified in the PCA. This approach allows for a graphical representation in a 2-dimensional space.

Generally, we can distinguish a group characterized by high values of “claim” and “blood pressure” from the remaining group in the dataset.

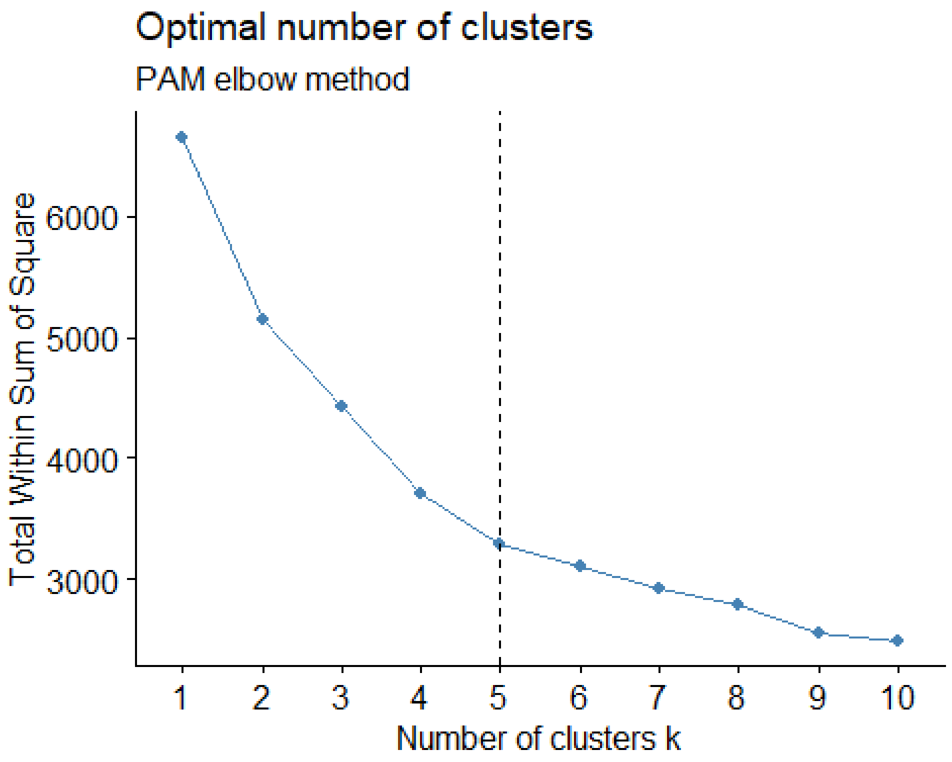
Partitional clustering: K-medoids (PAM)

The K-medoids (PAM) algorithm clusters statistical units using the “medoid” of each cluster as the merging criterion, continuing until the desired number of clusters is achieved. The medoid, the most central unit in a cluster, minimizes the average Euclidean (or Manhattan) distance to all other units, making this method more robust to outliers compared to K-means. Unlike the arithmetic mean, which is sensitive to outliers, the medoid serves as a representative unit within the data set.

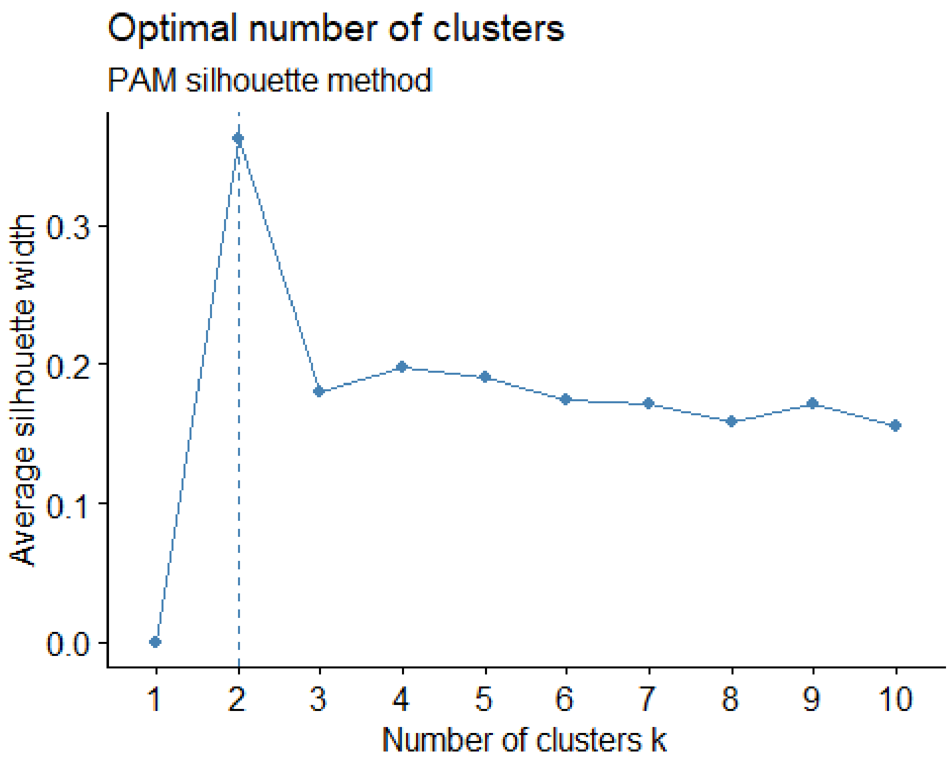
To implement the K-medoids algorithm in R, the “fpc” and “cluster” packages are required. Similar to K-means, an initial seed must be set to ensure reproducibility, as the algorithm begins with K randomly chosen units as temporary centroids.

Determining the optimal number of clusters is a crucial preliminary step. Since the NbClust function does not support PAM, alternative methods are used: the elbow method, the average silhouette width method, and the gap statistic method. The elbow method identifies the optimal number of clusters by analyzing the within sum of squares (WSS) and looking for an “elbow” point where the decrease in WSS becomes less significant.

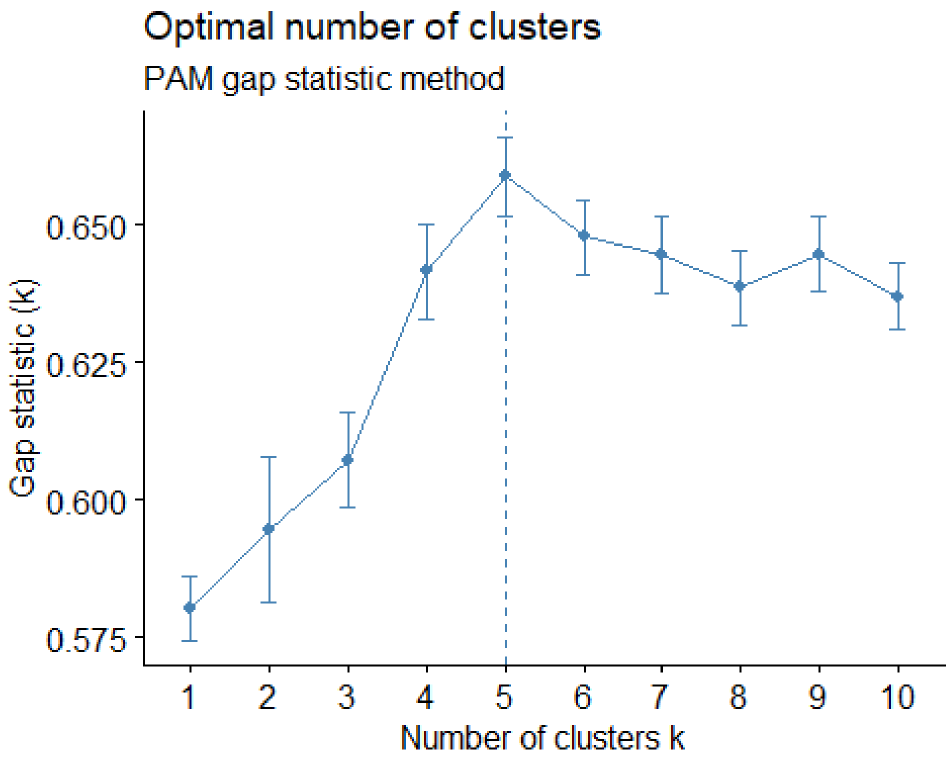
```r
#Partitional clustering: K-medoids (PAM)

# Use the elbow method to determine the number of clusters using PAM
fviz_nbclust(scaled_data, FUNcluster = pam, method = "wss") +
  labs(subtitle = "PAM elbow method") +
  geom_vline(xintercept = 5, linetype = 2)
```



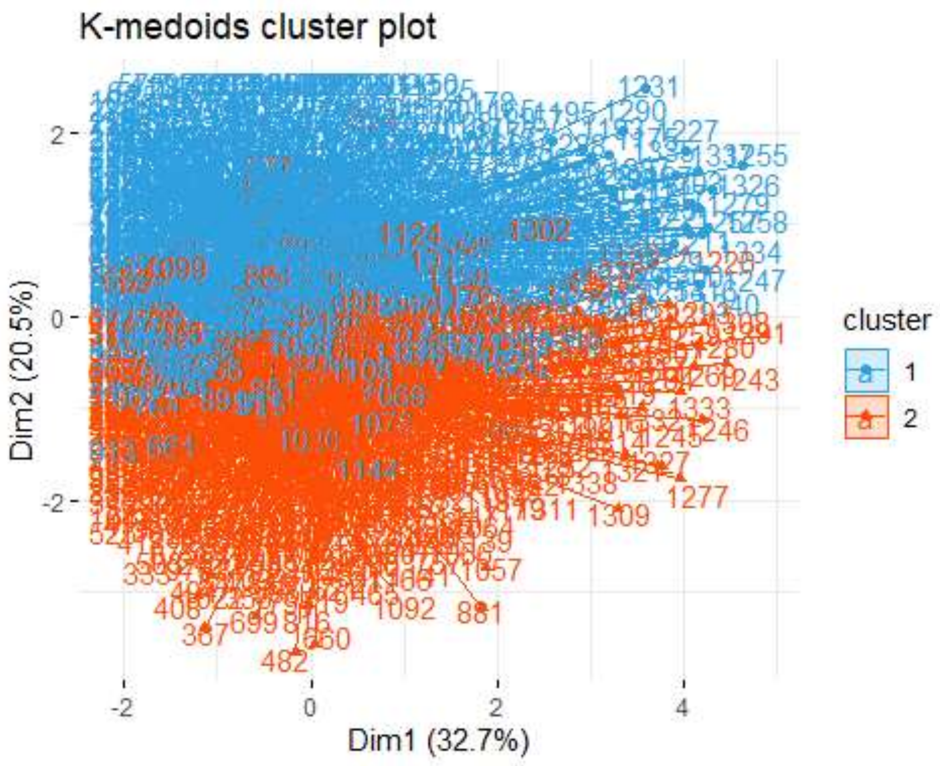
```r
# Use the silhouette method to determine the number of clusters using PAM
fviz_nbclust(scaled_data, FUNcluster = pam, method = "silhouette") +
  labs(subtitle = "PAM silhouette method")
```



```
# Use the gap statistic method to determine the number of clusters using PAM
fviz_nbclust(scaled_data, FUNcluster = pam, method = "gap_stat") +
  labs(subtitle = "PAM gap statistic method")
```



```r
# Perform K-medoids clustering with k = 2 and Manhattan metric
kmedoids <- pam(scaled_data, k = 2, metric = "manhattan")

# Display a table of cluster assignments
table(kmedoids$clustering)

## 
##   1   2 
## 743 589

# Visualize the K-medoids cluster plot
fviz_cluster(kmedoids, data = scaled_data,
             palette = c("#2E9FDF", "#FC4E07"),
             ellipse.type = "euclid", star.plot = TRUE,
             repel = TRUE, ggtheme = theme_minimal(),
             main = "K-medoids cluster plot")
```



The graphical representation differs significantly from the previous one. I opted to select the number of clusters suggested by the silhouette method, which matched the result obtained from the previous algorithm. However, the resulting clusters are notably different. This distinction is also evident when comparing the number of points in each cluster: Cluster 1 contains 743 points and Cluster 2 contains 589 points, indicating a more balanced distribution compared to the previously computed clusters.

Cluster Validation

The final step in cluster analysis involves the application of clustering validation techniques, which are categorized into internal, external, and relative measures. These techniques aim to provide a final evaluation of the clustering algorithms, identifying the most effective one.

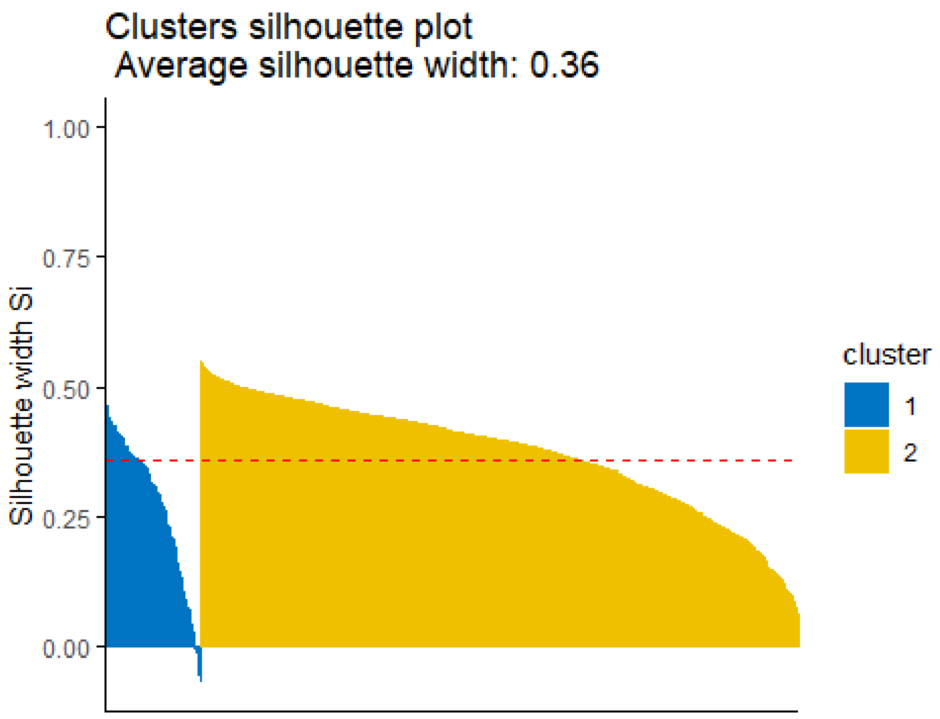
Internal Measures of Clustering Validation

Internal measures assess the performance of clustering algorithms based on internal cohesion and separation without requiring external information. The primary internal measures include:

```
Silhouette Width: This measure, already discussed in the context of estimating the
optimal number of clusters, should be maximized for effective clustering. It
evaluates how similar an object is to its own cluster compared to other clusters.

Dunn Index: This measure combines separation and cohesion, and like the silhouette
width, should also be maximized. It evaluates the smallest distance between
observations not in the same cluster and the largest intra-cluster distance.
```

The silhouette method will be applied to both K-means and K-medoids partitions. Units with negative silhouette widths indicate that these units may be better suited to a neighboring cluster rather than the one to which they have been assigned.

```
set.seed(123)
kmeans.data <- eclust(scaled_data, "kmeans", k = 2, nstart = 25, graph = FALSE)
table(kmeans.data$cluster)

##
##    1    2
##  182 1150

fviz_silhouette(kmeans.data, palette = "jco", ggtheme = theme_classic())

##   cluster size ave.sil.width
## 1       1  182          0.27
## 2       2 1150          0.37
```



he average silhouette width for the K-means partition is 0.36, indicating that, on average, the assignment of each unit to its respective cluster is well-defined.

All units in the second cluster exhibit a positive silhouette width, averaging 0.37, indicating they have been correctly assigned to their respective clusters. In contrast, the assignment in the first cluster shows variability, with some units incorrectly assigned (negative silhouette width averaging 0.27).

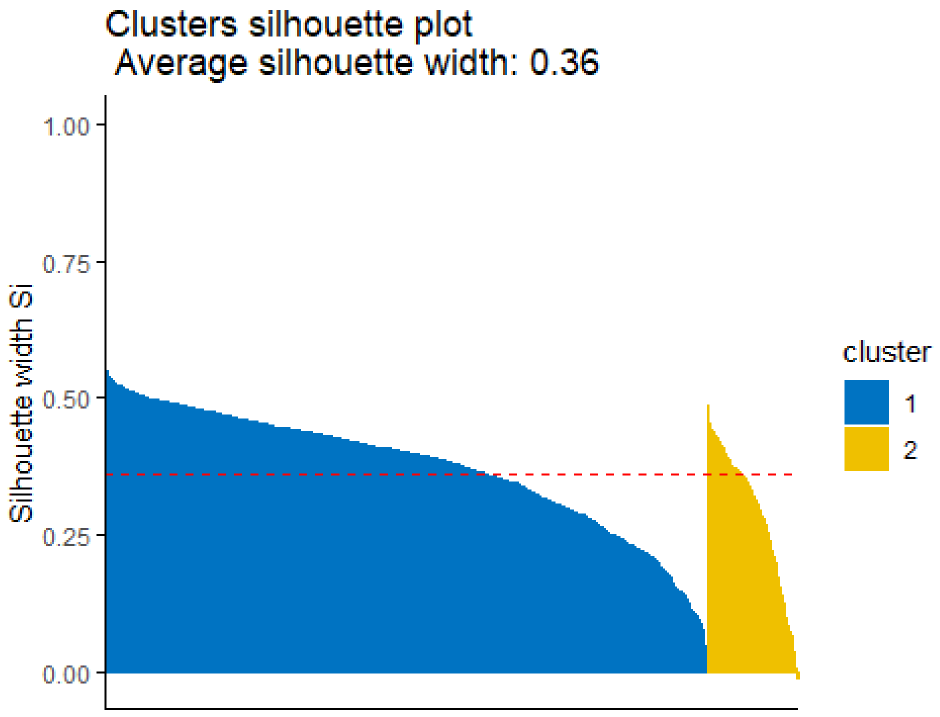
Next, an additional step is taken to identify units characterized by negative silhouette widths.

```
sil_kmeans <- silhouette(kmeans.data$cluster, manhdist_scaled)
neg_sil_kmeans <- which(sil_kmeans[, "sil_width"] < 0)
neg_sil_kmeans # Units characterized by negative silhouette widths

## [1]  795  873 1069 1074 1076 1124 1151
```

Seven points have been incorrectly assigned to clusters according to the K-means algorithm, with the majority of these allocated towards the upper end of the distribution, ranging from 795 to 1151.

```
set.seed(123)
kmedoids.data <- eclust(scaled_data, "pam", k = 2, nstart = 25, graph = FALSE)
table(kmedoids.data$clustering)

##
##    1    2
## 1156  176
```

```
fviz_silhouette(kmedoids.data, palette = "jco", ggtheme = theme_classic())

##   cluster size ave.sil.width
## 1       1 1156          0.37
## 2       2  176          0.28
```



The average silhouette width for the K-medoids partition is 0.36, indicating that, on average, the assignment of each unit to its respective cluster has been moderately discrete, albeit more optimal compared to the K-means algorithm.

In the first cluster, all assignments are correct with a silhouette width of 0.37. However, in the second cluster, there is a misplaced points, resulting in a lower silhouette width of 0.28.

Next, the units characterized by negative silhouette widths will be identified.

```
sil_kmedoids <- silhouette(kmedoids.data$clustering, manhdist_scaled)
neg_sil_kmedoids <- which(sil_kmedoids[, "sil_width"] < 0)
neg_sil_kmedoids # Units characterized by negative silhouette widths

## [1] 1151
```

This time only the point 1151 was misplaced.

Now the Dunn index will be computed alternatively to the silhouette width: the result will tell information about the goodness of both separation and cohesion operated by the two different algorithms. The usage of Dunn index requires the installation of the package "stats"

```
kmeans_stats <- cluster.stats(manhdist_scaled, kmeans.data$cluster)
kmeans_stats$dunn
```

```
## [1] 0.02710086
```

```
kmedoids_stats <- cluster.stats(manhdist_scaled, kmedoids.data$clustering)
kmedoids_stats$dunn
```

```
## [1] 0.07577049
```

As it can be seen, the Dunn index value generated by the K-medoids partition is higher than the same generated by the K-means partition: this confirms again that the K-medoids algorithm has performed a better assignment than the K-means one.

The evaluation of the clustering algorithms' effectiveness is summarized using the "clValid" package, which offers a range of internal and stability measures to assess their quality. Internal measures utilize inherent data characteristics to evaluate clustering quality, whereas stability measures assess the consistency of clustering results when each variable is sequentially removed and re-evaluated.

```
cluster_methods <- c("hierarchical", "kmeans", "pam")
internal <- clValid(scaled_data, nClust = 2, clMethods =
cluster_methods,validation = "internal", maxitems = nrow(scaled_data) + 1)

summary(internal)
```

```
##
## Clustering Methods:
##  hierarchical kmeans pam
##
## Cluster sizes:
##  2
##
## Validation Measures:
##                                  2
##
## hierarchical Connectivity  49.5052
##              Dunn           0.0799
##              Silhouette     0.3638
## kmeans       Connectivity  71.6425
##              Dunn           0.0384
##              Silhouette     0.3588
## pam          Connectivity  53.2921
##              Dunn           0.0931
##              Silhouette     0.3623
##
## Optimal Scores:
##
##              Score   Method       Clusters
## Connectivity 49.5052 hierarchical 2
## Dunn          0.0931 pam          2
## Silhouette    0.3638 hierarchical 2
```

Based on the internal evaluation metrics, hierarchical methods appear to provide the most favorable partitioning results, as indicated by higher Dunn index (0.0931) and silhouette width (0.3638) values compared to the PAM method (Dunn index: 49.5052, silhouette: 0.3638). Therefore, from an internal evaluation standpoint, hierarchical algorithms demonstrate superior performance in clustering effectiveness.

```r
cluster_methods <- c("hierarchical","kmeans","pam")
stability <- clValid(scaled_data, nClust = 2, clMethods =
cluster_methods,validation = "stability", maxitems = nrow(scaled_data) + 1)

summary(stability)

##
## Clustering Methods:
##  hierarchical kmeans pam
##
## Cluster sizes:
##  2
##
## Validation Measures:
##                           2
##
## hierarchical APN  0.0088
##              AD   2.8952
##              ADM  0.5685
##              FOM  0.9945
## kmeans       APN  0.1873
##              AD   2.7632
##              ADM  0.5060
##              FOM  0.9952
## pam          APN  0.1131
##              AD   2.7015
##              ADM  0.2883
##              FOM  0.9652
##
## Optimal Scores:
##
##     Score  Method       Clusters
## APN 0.0088 hierarchical 2
## AD  2.7015 pam          2
## ADM 0.2883 pam          2
## FOM 0.9652 pam          2
```

The summary of stability measures are unanimous into identifying the pam algorithm as the one capable of generating the most resistant clustering partitions. So, this is the best algorithm from the stability's point of view.

Model-Based Clustering

This section of this report addresses the soft-clustering approach, encompassing clustering techniques that assign each unit to all clusters with varying degrees of intensity. This intensity is expressed as "degree of membership" in fuzzy clustering techniques or "level of probability" in model-based clustering. This approach mitigates the limitations of hard clustering, acknowledging that the assignment of a unit to a specific cluster can be ambiguous.

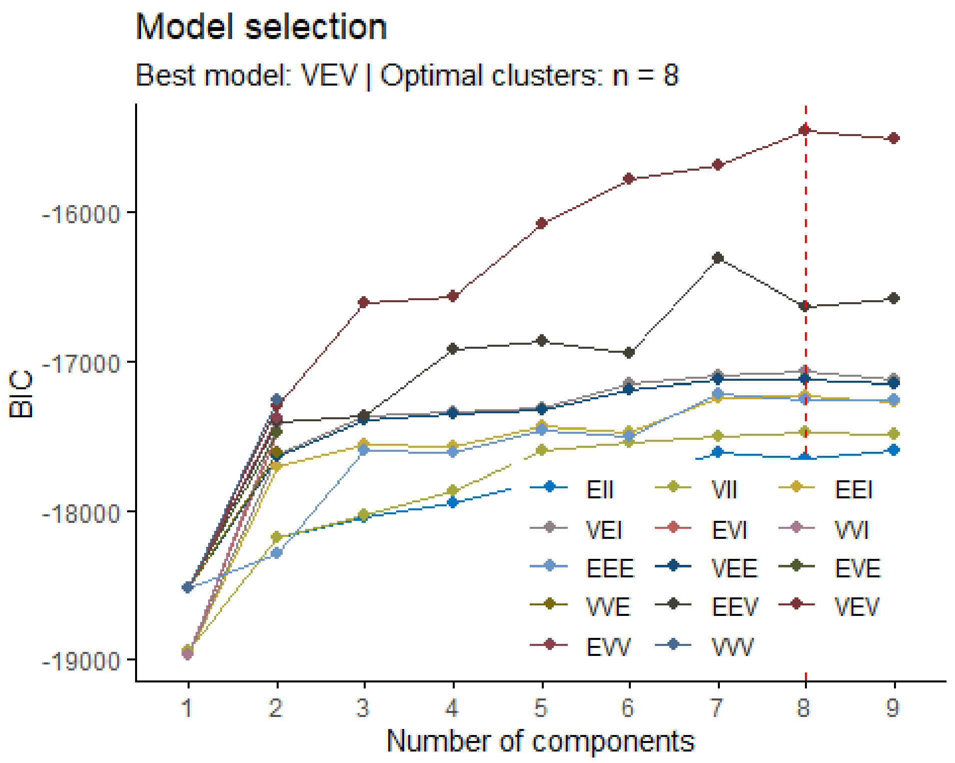
Model-Based Clustering

In R, the application of model-based clustering techniques requires the "mclust" package. Model-based clustering relies on parameterized finite Gaussian mixture models, which can be seen as mixtures of probability distribution functions corresponding to the units assigned to each component. Each component represents a cluster constructed under a specific probability distribution assumption.

The models are estimated using the Expectation-Maximization (EM) algorithm, which seeks the best set of parameters for the mixture. This process involves an initial guess of these parameters to determine the probabilities of each unit being assigned to each cluster. The Mclust function assumes that the probability distribution of each component follows a normal distribution, characterized by the mean vector (defining the distribution's barycenter) and the sample covariance matrix (defining the volume, shape, and orientation of the distribution). The optimal model is selected based on the Bayesian Information Criterion (BIC) value, among all possible combinations of normal mixtures.

```r
model_based <- Mclust(scaled_data)
fviz_mclust(model_based, "BIC", palette = "jco")
```

```
## Warning: `gather_()` was deprecated in tidyr 1.2.0.
## ℹ Please use `gather()` instead.
## ℹ The deprecated feature was likely used in the factoextra package.
##   Please report the issue at <https://github.com/kassambara/factoextra/issues>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



The function provides 14 different possible combinations of finite mixture models obtained by the Eigen decompositions on 3 groups (K=3) and on 2 dimensions (d = 2). The best model to be chosen will be the one characterized by the highest value of BIC (Bayesian Information Criterion), an index useful for data fitting evaluation which identifies, if it is high enough, a good trade-off between parsimony (number of parameters) and accuracy (fit) of the model. Another aspect is related to the shape of the clusters that the model is able to generate

```
summary(model_based$BIC)

## Best BIC values:
##              VEV,8       VEV,9       VEV,7
## BIC      -15464.19 -15509.5018 -15684.4906
## BIC diff      0.00    -45.3076   -220.2964
```

In this case, the best model according to BIC values is VEV with BIC = -15464.19. The model identifier 'VEV' indicates a mixture model assuming variable volume and shape, suggesting the presence of 8 clusters characterized by varying distributions across the coordinate axes.

```
summary(model_based)

## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VEV (ellipsoidal, equal shape) model with 8 components:
##
##  log-likelihood    n  df       BIC       ICL
##       -7232.084 1332 139 -15464.19 -15616.55
```

```
##
## Clustering table:
##   1   2   3   4   5   6   7   8
##  74 102  53 276 492 162  82  91
```

The summary provides details of a Gaussian finite mixture model fitted using the EM algorithm, specifically the Mclust VEV (ellipsoidal, equal shape) model with 8 components.

The clustering table shows the distribution of observations across the 8 clusters: 74 in Cluster 1, 102 in Cluster 2, 53 in Cluster 3, 276 in Cluster 4, 492 in Cluster 5, 162 in Cluster 6, 82 in Cluster 7, and 91 in Cluster 8.

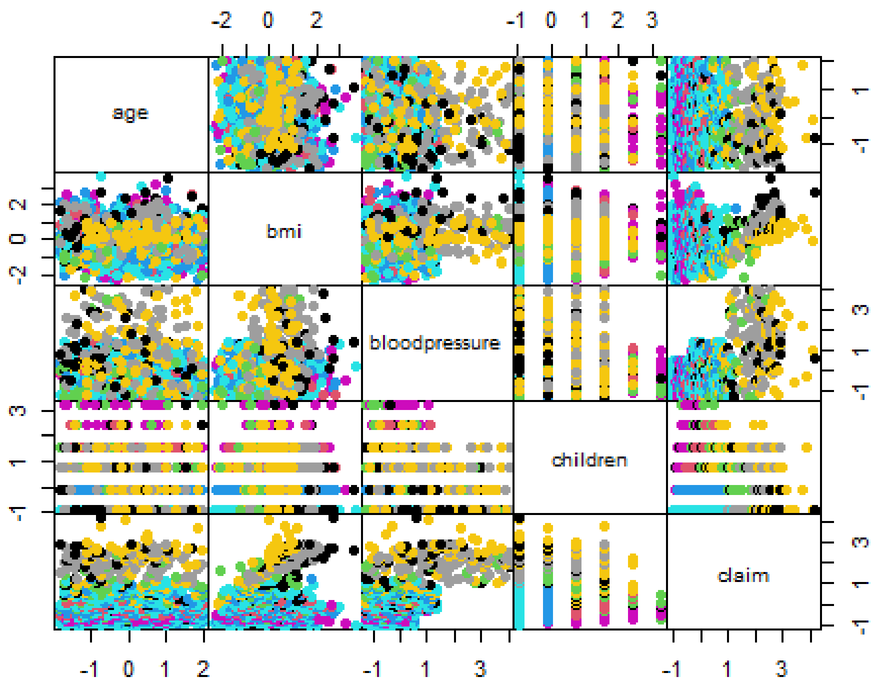
The classification is derived from a probabilistic approach where each unit is assigned to clusters based on maximum a posteriori probability. Initially, soft assignment probabilities are calculated for each unit across clusters. The transition to hard assignment occurs by assigning a value of 1 to the cluster with the highest posterior probability for each unit.

```
head(model_based$z, 30) # Soft assignment
```

```
##              [,1]         [,2]          [,3]         [,4]      [,5]        [,6]
## 1   6.645204e-13 1.084755e-04 2.175355e-141 1.476443e-55 0.9961723 0.003719259
## 2   2.716262e-69 7.716215e-06  1.837944e-67 1.590776e-55 0.9884948 0.011497534
## 8  2.904342e-172 4.384077e-04  8.440428e-29 9.466706e-56 0.9799518 0.019609758
## 9  6.039544e-194 1.241366e-05  2.399479e-16 1.096628e-55 0.9261815 0.073806098
## 10 4.914257e-258 1.959681e-03  6.150515e-08 1.303001e-56 0.6705620 0.327478286
## 11  1.351931e-08 3.752802e-05 2.369696e-155 9.675210e-56 0.9902582 0.009704302
## 12  1.209563e-08 7.048174e-05 3.046225e-157 1.121088e-55 0.9928360 0.007093527
## 13  1.091266e-30 2.413909e-07 9.973513e-105 5.803507e-56 0.9539979 0.046001898
## 17  2.839590e-65 5.139038e-05  1.042393e-64 1.800542e-55 0.9940495 0.005899156
## 18  4.563401e-54 2.522449e-05  1.199322e-63 1.864474e-55 0.9916787 0.008296095
## 19  2.331962e-39 5.572825e-04  4.216570e-85 1.706809e-55 0.9970369 0.002405832
## 20  4.434859e-33 2.427844e-04  2.598524e-87 2.008817e-55 0.9948010 0.004956171
## 21  1.505136e-08 2.571030e-05 2.159130e-139 1.319293e-55 0.9925927 0.007381552
## 22  3.101027e-48 1.102345e-04  9.121967e-86 1.874471e-55 0.9964282 0.003461605
## 23 1.277718e-121 2.045878e-04  2.174880e-47 1.492311e-55 0.9846406 0.015154823
## 24  5.319628e-63 1.820262e-04  7.698024e-62 1.654179e-55 0.9930570 0.006760972
## 25  6.210355e-05 5.782450e-05 1.862784e-167 1.147249e-55 0.9893036 0.010576464
## 26 6.205539e-181 1.677848e-05  3.842151e-18 1.123239e-55 0.9416038 0.058379400
## 27  6.857122e-25 1.416883e-04 7.793508e-118 1.893278e-55 0.9969532 0.002905154
## 28  1.594498e-07 5.653676e-05 1.412098e-168 9.102365e-56 0.9811313 0.018811361
## 29  7.937874e-13 4.737427e-04 3.192236e-109 2.268042e-55 0.9875743 0.011951922
## 30  3.602637e-48 5.268469e-05  4.420491e-81 1.885201e-55 0.9952104 0.004736946
## 31  8.788741e-16 1.411999e-07 2.064422e-120 6.275105e-56 0.9839846 0.016015299
## 32  2.062644e-05 8.190899e-06 1.440146e-147 9.990355e-56 0.9890584 0.010912825
## 33  4.466764e-32 5.869518e-05 9.346579e-118 1.443654e-55 0.9818856 0.018055704
## 34  1.053541e-42 5.457306e-05  1.137871e-59 2.020688e-55 0.9802335 0.019711966
## 35  7.614349e-70 4.206541e-05  3.110845e-49 1.732291e-55 0.9861742 0.013783726
## 36  1.230423e-58 3.583521e-04  1.304771e-60 1.561872e-55 0.9897316 0.009910004
## 37  2.477285e-08 4.566391e-04 1.121057e-152 2.020622e-55 0.9976062 0.001937091
## 38  4.788016e-26 1.249883e-06  1.184766e-99 1.285124e-55 0.9925835 0.007415291
##              [,7]          [,8]
## 1   2.784104e-20  2.866458e-74
## 2   7.414053e-51 9.511401e-119
## 8  1.859244e-126 7.691616e-216
## 9  2.275058e-141 1.198679e-242
## 10 5.535892e-241  0.000000e+00
## 11  2.458862e-11  6.036188e-54
## 12  2.081285e-12  1.707911e-56
## 13  5.251622e-14  6.641497e-55
## 17  2.890639e-71 5.670615e-156
## 18  3.617624e-71 5.549805e-163
## 19  6.761565e-76 4.227613e-172
## 20  5.229037e-63 8.068058e-153
## 21  7.235997e-19  7.276197e-75
## 22  9.037844e-54 1.505453e-125
## 23  2.223643e-83 1.696163e-155
## 24  5.703194e-87 1.292980e-182
## 25  8.383810e-12  1.891596e-61
## 26 1.682933e-140 2.081447e-244
## 27  1.664833e-32  6.521992e-92
## 28  6.476923e-07  5.735324e-39
## 29  8.421144e-48 1.853137e-136
## 30  6.098777e-55 1.409161e-128
## 31  2.246347e-12  2.244955e-54
## 32  5.339087e-14  1.584951e-67
## 33  2.482181e-19  1.892032e-61
## 34  1.151814e-82 9.525922e-190
## 35  6.261989e-95 1.556761e-197
## 36  6.251247e-95 2.699351e-198
## 37  9.826387e-25  5.263132e-85
## 38  4.158224e-28  1.132382e-86
```

```r
head(model_based$classification, 200) # Hard assignment
```

```
##   1   2   8   9  10  11  12  13  17  18  19  20  21  22  23  24  25  26  27  28
##   5   5   5   5   5   5   5   5   5   5   5   5   5   5   5   5   5   5   5   5
##  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48
##   5   5   5   5   5   5   5   5   5   5   5   5   5   5   5   5   5   5   5   5
##  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68
##   5   5   5   5   5   5   5   4   5   4   4   4   4   5   5   5   5   5   5   5
##  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88
##   5   5   5   5   5   4   4   5   4   5   5   5   5   5   5   5   4   5   5   5
##  89  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
##   5   4   5   5   5   5   5   5   5   5   5   5   4   5   5   5   5   5   5   5
## 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128
##   5   5   5   5   5   5   5   5   5   4   5   5   5   5   5   5   5   4   4   5
## 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148
##   5   5   5   5   1   5   4   5   4   5   5   5   4   6   5   5   5   5   5   5
## 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168
##   5   5   5   5   5   5   5   6   4   5   4   5   4   4   5   5   5   5   4   4
## 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188
##   4   4   5   5   4   5   5   5   4   4   1   6   5   5   5   5   5   5   5   4
## 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208
##   4   1   6   5   5   5   5   6   5   5   5   1   5   4   5   5   5   5   6   5

table(model_based$classification)

##
##   1   2   3   4   5   6   7   8
##  74 102  53 276 492 162  82  91

pairs(scaled_data, gap = 0, pch = 16, col = model_based$classification)
```



The matrix of scatterplots shows the clustering structure of the data according to the partition done by the hard-assignment correction of the model-based partitioning. It can be seen that even the model-based approach suggests the presence of clusters in the data.

Fuzzy Clustering

Fuzzy clustering allows observations to belong to several clusters simultaneously, with each membership expressed as a degree of belonging. The fuzzy K-means approach, an extension of the traditional K-means method, assigns membership values to observations for each cluster. This approach is particularly useful when the boundaries between clusters are not clear-cut, reflecting the inherent uncertainty in cluster assignments.

Fuzzy clustering K-Means

```
# Perform fuzzy clustering with all numerical columns
fkm_result <- FKM(X = n_data, stand = 1, RS = 10, seed = 264)

# Display the number of clusters chosen and the criterion
fkm_result$k

## Number of clusters
##                  2

fkm_result$criterion

## SIL.F k=2 SIL.F k=3 SIL.F k=4 SIL.F k=5 SIL.F k=6
## 0.4293872 0.2764008 0.4150422 0.3328404 0.3326483
```

The silhouette width (SIL.F) values indicate the quality of clustering for different numbers of clusters (k) tested. For k=2, the silhouette width is 0.4293872, suggesting that the clustering at this number of clusters has relatively well-separated clusters.

```
# Perform fuzzy clustering with the XB index
fkm_result_XB <- FKM(X = n_data, stand = 1, RS = 10, seed = 264, index = "XB")

# Display the criterion for fuzzy clustering with XB index
fkm_result_XB$criterion

##       XB k=2       XB k=3       XB k=4       XB k=5       XB k=6
## 9.252053e+00 1.234929e+01 2.053855e+00 1.995944e+17 4.324456e+18
```

The XB (Xie-Beni) index values provide a measure of clustering quality for different numbers of clusters (k). For k=2, the XB index is 9.252053e+00, indicating the compactness and separation of clusters. The lowest values give us the best numer of cluster.

```
# Perform fuzzy clustering with the PC index
fkm_result_PC <- FKM(X = n_data, stand = 1, RS = 10, seed = 264, index = "PC")

# Display the criterion for fuzzy clustering with PC index
fkm_result_PC$criterion

##    PC k=2    PC k=3    PC k=4    PC k=5    PC k=6
## 0.5103162 0.3439063 0.3106381 0.2523595 0.2081049
```

The presented results show the Partition Coefficient (PC) values for fuzzy clustering across different numbers of clusters (k=2 to k=6). The PC index measures the average membership degree of data points to the clusters.

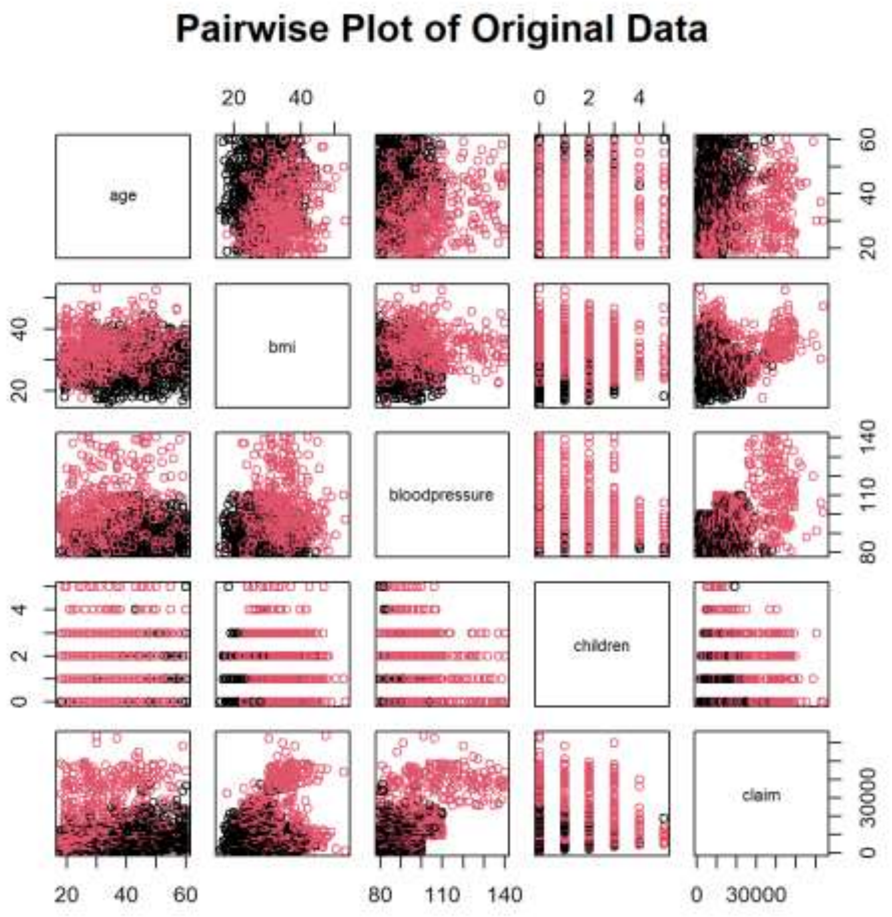
Higher PC values indicate a stronger degree of membership certainty for each data point across the specified number of clusters.

```
# Perform fuzzy clustering with K=2
fkm_result_2 <- FKM(X = n_data, k = 2, stand = 1, RS = 10, seed = 264)

# Display the first results of the membership degree for each cluster
head(fkm_result_2[["clus"]])
```

```
##    Cluster Membership degree
## 1        1         0.6146911
## 2        1         0.5467616
## 8        2         0.5228304
## 9        2         0.5059330
## 10       2         0.5147643
## 11       1         0.5848860

# Display the size of the clusters
cl.size(fkm_result_2$U)

## Clus 1 Clus 2
##    761    571
```

In the initial iteration, no specific value of k is imposed. Instead, the function autonomously determines the optimal number of clusters based on the standard Fuzzy silhouette index. Subsequently, the function is executed using various indices to ascertain the optimal number of clusters. Both the Xie and Beni index and the Partition Coefficient index recommend $k = 2$. Following the majority rule approach, $k = 2$ is chosen as the optimal number of clusters.

```
# Determine the max membership cluster for each observation
max_membership_cluster <- apply(fkm_result_2$clus, 1, which.max)

# Pairwise plot of the original data colored by the max membership cluster
pairs(n_data, col = max_membership_cluster, main = "Pairwise Plot of Original
Data")
```



Appendix

Scott's Rule for Optimal Number of Histogram Bins

Scott's rule is a method used to determine the optimal number of bins for a histogram. The goal is to find a bin width that accurately represents the distribution of the data without over-simplifying or creating too much detail.

Formula

The formula for calculating the bin width h using Scott's rule is:

$$h = \frac{3.5 \times \sigma}{n^{1/3}}$$

Where:

- σ is the standard deviation of the data.
- n is the sample size (number of data points).