# Sleep Health and Lifestyle

Statistical Analysis in R

ROSARIO PAVONE

# DATA DESCRIPTION

Dataset Overview: The Sleep Health and Lifestyle Dataset consists of 400 rows and 13 columns, encompassing various variables related to sleep and daily habits. It provides information such as gender, age, occupation, sleep duration, sleep quality, physical activity level, stress levels, BMI category.

Key Features of the Dataset: Comprehensive Sleep Metrics: Explore variables such as sleep duration, quality, and factors influencing sleep patterns. Lifestyle Factors: Analyze physical activity levels, stress levels, and BMI categories. Cardiovascular Health: Examine measurements of blood pressure and heart rate. Sleep Disorder Analysis: Identify occurrences of sleep disorders such as Insomnia and Sleep Apnea.

Dataset Columns: Person ID: An identifier for each individual. Gender: The gender of the person (Male/Female). Age: The age of the person in years. Occupation: The occupation or profession of the person. Sleep Duration (hours): The number of hours the person sleeps per day. Quality of Sleep (scale: 1-10): A subjective rating of sleep quality, ranging from 1 to 10. Physical Activity Level (minutes/day): The number of minutes the person engages in physical activity daily. Stress Level (scale: 1-10): A subjective rating of the person's stress level, ranging from 1 to 10. BMI Category: The BMI category of the person (e.g., Underweight, Normal, Overweight). Blood Pressure (systolic/diastolic): The person's blood pressure measurement, indicated as systolic pressure over diastolic pressure. Heart Rate (bpm): The resting heart rate of the person in beats per minute. Daily Steps: The number of steps the person takes per day. Sleep Disorder: The presence or absence of a sleep disorder in the person (None, Insomnia, Sleep Apnea).

## Details about Sleep Disorder Column:

None: The individual does not exhibit any specific sleep disorder. Insomnia: The individual experiences difficulty falling asleep or staying asleep, leading to inadequate or poor-quality sleep. Sleep Apnea: The individual suffers from pauses in breathing during sleep, resulting in disrupted sleep patterns and potential health risks.

```
> head(Sleep_health_and_lifestyle_dataset)
```

| | Occupation | Sleep_Duration | Quality_of_Sleep | Physical_Activity_Level | Stress_Level | Sleep_Disorder |
|---|---|---|---|---|---|---|
| 1 | 27 Software Engineer | 6.1 | 6 | 42 | 6 | None |
| 2 | 28 Doctor | 6.2 | 6 | 60 | 8 | None |
| 3 | 28 Doctor | 6.2 | 6 | 60 | 8 | None |
| 4 | 28 Sales Representative | 5.9 | 4 | 30 | 8 | Sleep Apnea |
| 5 | 28 Sales Representative | 5.9 | 4 | 30 | 8 | Sleep Apnea |
| 6 | 28 Software Engineer | 5.9 | 4 | 30 | 8 | Insomnia |

```
str(Sleep_health_and_lifestyle_dataset)
Classes ‘spec_tbl_df’, ‘tbl_df’, ‘tbl’ and 'data.frame':	374 obs. of  7 variables:
 $ Age                : num  27 28 28 28 28 28 29 29 29 29 ...
 $ Occupation         : chr  "Software Engineer" "Doctor" "Doctor" "Sales Represent
ative" ...
 $ Sleep_Duration     : num  6.1 6.2 6.2 5.9 5.9 5.9 6.3 7.8 7.8 7.8 ...
 $ Quality_of_Sleep   : num  6 6 6 4 4 4 6 7 7 7 ...
```

```
 $ Physical_Activity_Level: num  42 60 60 30 30 30 40 75 75 75 ...
 $ Stress_Level           : num  6 8 8 8 8 8 7 6 6 6 ...
 $ Sleep_Disorder         : chr  "None" "None" "None" "Sleep Apnea" ...
> dim( Sleep_health_and_lifestyle_dataset)
[1] 374   6
```

These variables collectively provide a comprehensive overview of various aspects of an individual's health, lifestyle, and well-being.

The Sleep Health and Lifestyle Dataset comprises 400 rows and 13 columns, covering a wide range of variables related to sleep and daily habits. It includes details such as gender, age, occupation, sleep duration, quality of sleep, physical activity level, stress levels, BMI category, blood pressure, heart rate, daily steps, and the presence or absence of sleep disorders.

```
 # Print current column names
 > print(names(Sleep_health_and_lifestyle_dataset))

  print(names(Sleep_health_and_lifestyle_dataset))
  [1] "Occupation"  "Sleep_Duration" "Quality_of_Sleep" Physical_Activity_Level"
  [5] "Stress_Level" "Sleep_Disorder"
```

# UNIVARIATE ANALYSIS

This is the analysis of the single variables. Let’s look at some of them.

```
>  # Main statistical indices of each variable
> summary(Sleep_health_and_lifestyle_dataset)
   summary(Sleep_health_and_lifestyle_dataset)
      Age         Occupation        Sleep_Duration  Quality_of_Sleep Physical_Activity_L
evel  Stress_Level
 Min.   :27.00   Length:374         Min.   :5.800   Min.   :4.000    Min.   :30.00          Mi
n.   :3.000
 1st Qu.:35.25   Class :character   1st Qu.:6.400   1st Qu.:6.000    1st Qu.:45.00
1st Qu.:4.000
 Median :43.00   Mode  :character   Median :7.200   Median :7.000    Median :60.00
Median :5.000
 Mean   :42.18                      Mean   :7.132   Mean   :7.313    Mean   :59.17          Me
an   :5.385
 3rd Qu.:50.00                      3rd Qu.:7.800   3rd Qu.:8.000    3rd Qu.:75.00          3r
d Qu.:7.000
 Max.   :59.00                      Max.   :8.500   Max.   :9.000    Max.   :90.00          Max.
:8.000
 Sleep_Disorder
 Length:374
 Class :character
```

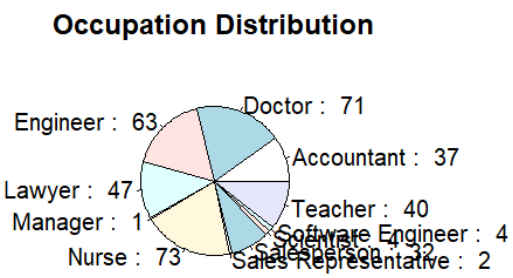
Mode  :character

## 1.OCCUPATION

the variable "Occupation" is nominal. Nominal variables are categorical variables that represent different categories or levels with no inherent order or ranking among them. In this case, the "Occupation" variable represents different occupations, and there is no inherent order or ranking among the various occupations.

Creating a pie chart for this variable is a suitable way to visualize the distribution of different occupations in the dataset. Each slice of the pie chart represents a different occupation category, and the size of each slice corresponds to the frequency or count of individuals in each occupation category.

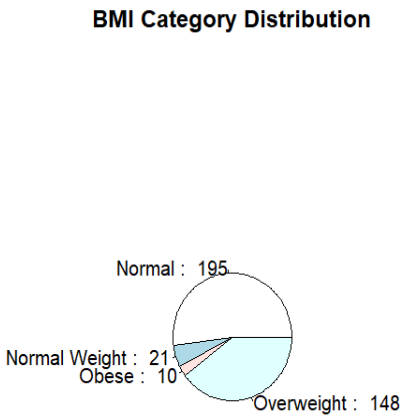
```
> # Occupation Pie Chart
> occupation_summary <- table(Sleep_health_and_lifestyle_dataset$Occupation)
> pie(occupation_summary, labels = paste(names(occupation_summary), ": ", occupation_summary), main = "Occupation Distribution")
```



## 2.BMI CATEGORY

the variable "BMI_Category" is a Nominal variables:

```
> # BMI Category Pie Chart
> bmi_summary <- table(Sleep_health_and_lifestyle_dataset$BMI_Category)
> pie(bmi_summary, labels = paste(names(bmi_summary), ": ", bmi_summary), ma
in = "BMI Category Distribution")
```



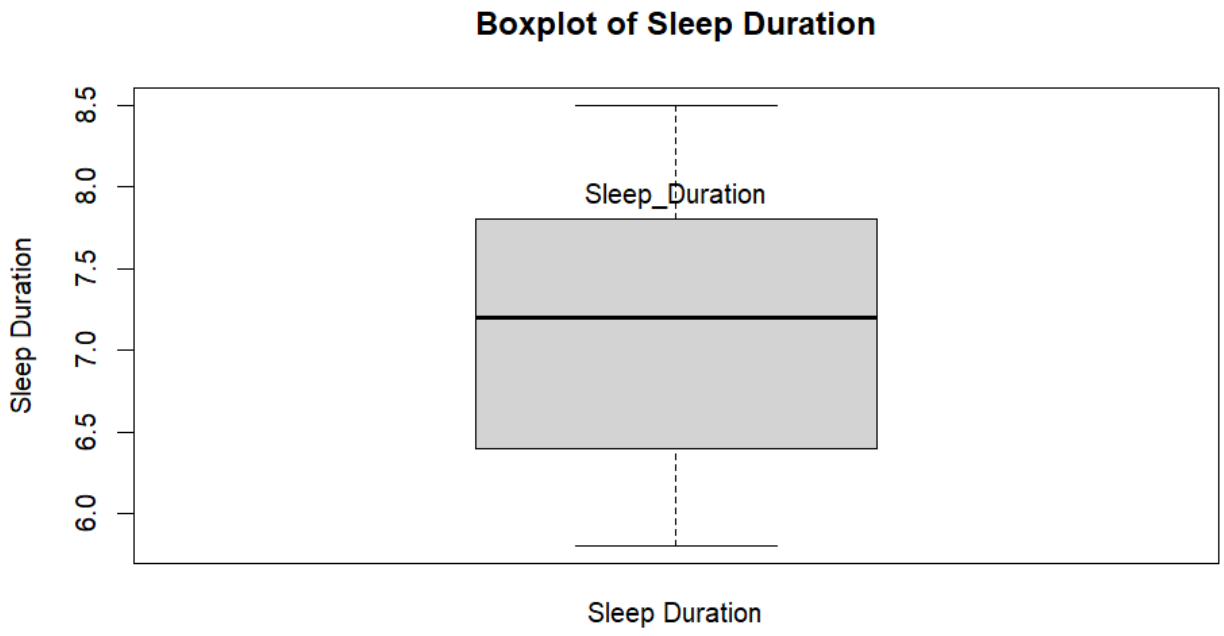

BMI stands for Body Mass Index. It is a measure of body fat based on an individual's weight and height. BMI is calculated by dividing a person's weight in kilograms by the square of their height in meters. The formula for BMI is: BMI=Weight (kg)Height (m)2BMI=Height (m)2Weight (kg)

## 3.SLEEPING DURATION

“Sleeping Duration“ ia a numeric and continous random variable .

```
> summary(Sleep_health_and_lifestyle_dataset$Sleep_Duration)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  5.800   6.400   7.200   7.132   7.800   8.500 
```

The **summary** function provides key statistical measures for the variable under consideration. Notably, it reveals the range of the variable from its minimum value to its maximum, indicating the spread of values. The mean, a central tendency measure, is computed at 7.132, offering insights into the average value. Additionally, the first and third quartiles delineate the distribution, and the interquartile gap, representing the spread between these quartiles, serves as a valuable dispersion index. Quartiles play a crucial role in constructing a box plot, aiding in visualizing the distribution characteristics of the variable.There isn’t any outlier(a data point that differs significantly from other observations) in the distribution.



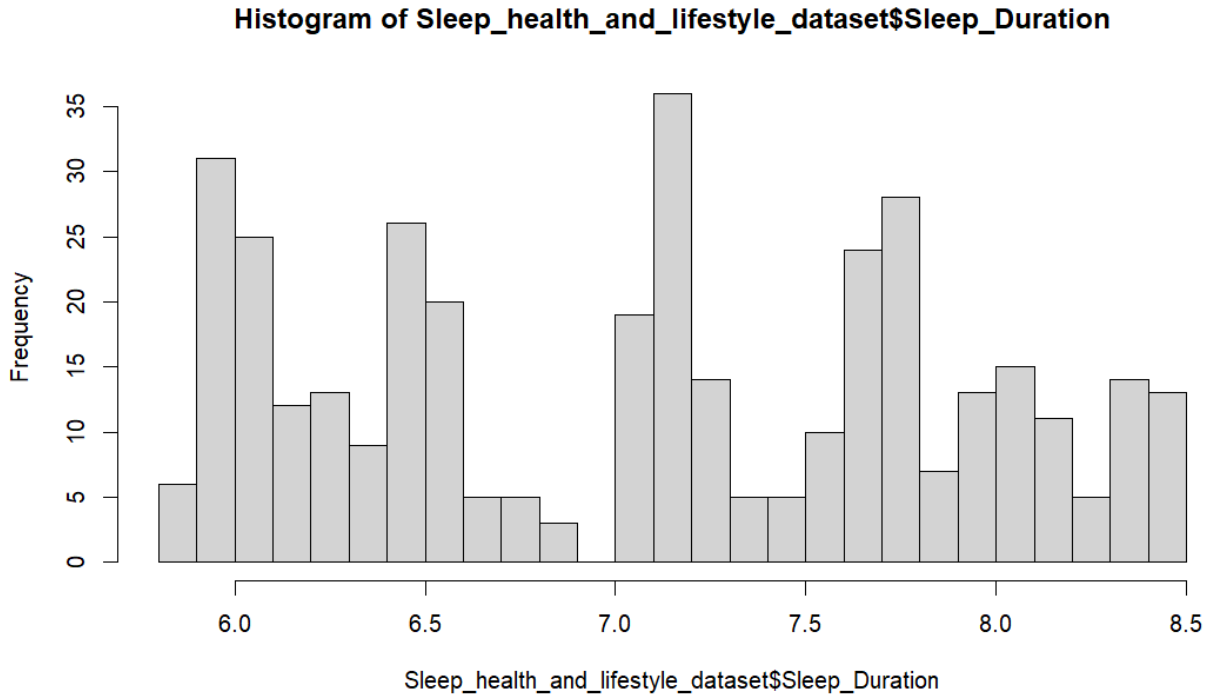

```
> frequencySleep_Duration <- table(Sleep_Duration)
> frequencySleep_Duration
Sleep_Duration
5.8 5.9   6 6.1 6.2 6.3 6.4 6.5 6.6 6.7 
  2   4  31  25  12  13   9  26  20   5 
```

```
6.8 6.9 7.1 7.2 7.3 7.4 7.5 7.6 7.7 7.8
  5   3  19  36  14   5   5  10  24  28
7.9   8 8.1 8.2 8.3 8.4 8.5
  7  13  15  11   5  14  13
```

```
length(frequencySleep_Duration)
[1] 27
> names(frequencySleep_Duration)[frequencySleep_Duration == max(frequencySle
ep_Duration)]#mode
[1] "7.2
> labstatR::cv(Sleep_Duration)
[1] 0.1114109
```

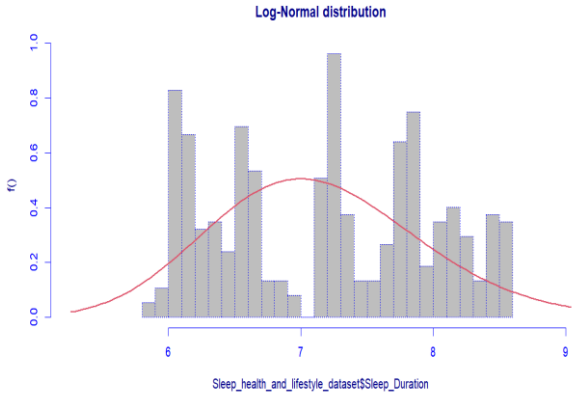
This variable takes on 27 values, ranging from 5.8 to 8.5, representing the duration of sleep. The mode is determined using the names() function with the argument frequencySleep_Duration. The square brackets indicate that when the argument within these brackets is at its maximum, the corresponding value is returned.
The coefficient of variation is approximately 0.11%.

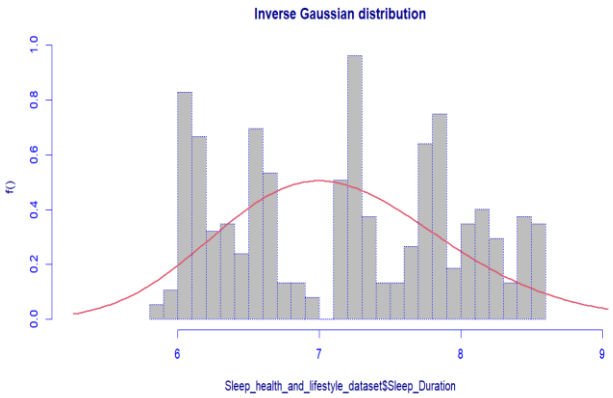
```
> hist(Sleep_health_and_lifestyle_dataset$Sleep_Duration, breaks = 27)
```

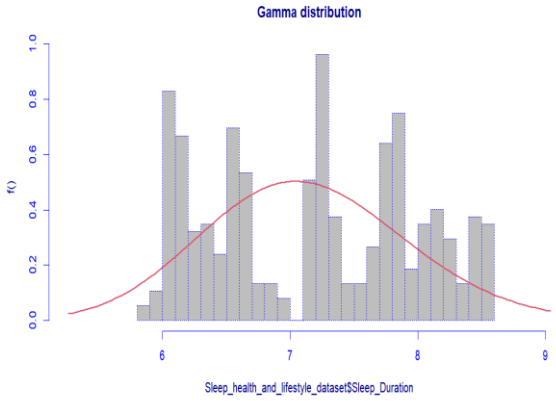


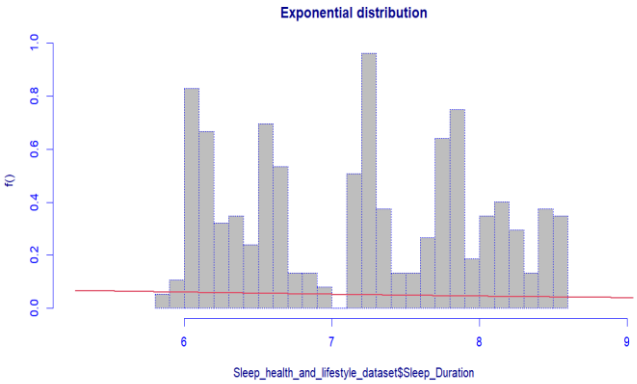

Now I show different model to fit the Sleep Duration distribution

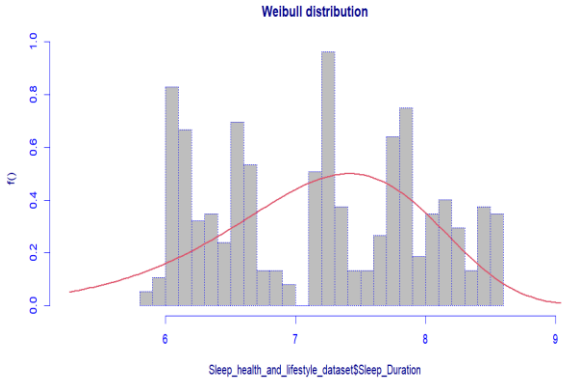
```
> fit.WEI <- histDist(Sleep_health_and_lifestyle_dataset$Sleep_Duration, family=WEI, nbins = 27, main="Weibull distribution")
> fit.EXP <- histDist(Sleep_health_and_lifestyle_dataset$Sleep_Duration, family=EXP, nbins = 27, main="Exponential distribution")
> fit.GA <- histDist(Sleep_health_and_lifestyle_dataset$Sleep_Duration, family=GA, nbins = 27, main="Gamma distribution")
> fit.IG <- histDist(Sleep_health_and_lifestyle_dataset$Sleep_Duration, family=IG, nbins = 27, main="Inverse Gaussian distribution")
> fit.LOGNO <- histDist(Sleep_health_and_lifestyle_dataset$Sleep_Duration, family=LOGNO, nbins = 27, main="Log-Normal distribution")
> fit.WEI <- histDist(Sleep_health_and_lifestyle_dataset$Sleep_Duration, family=WEI, nbins = 27, main="Weibull distribution")
```




AIC SBC

```
1      Exponential 2219.5236 2223.4478
2            Gamma  892.4165  900.2650
3 Inverse Gaussian  892.7573  900.6058
4       Log-Normal  893.2589  901.1074
5          Weibull  903.4157  911.2642
```

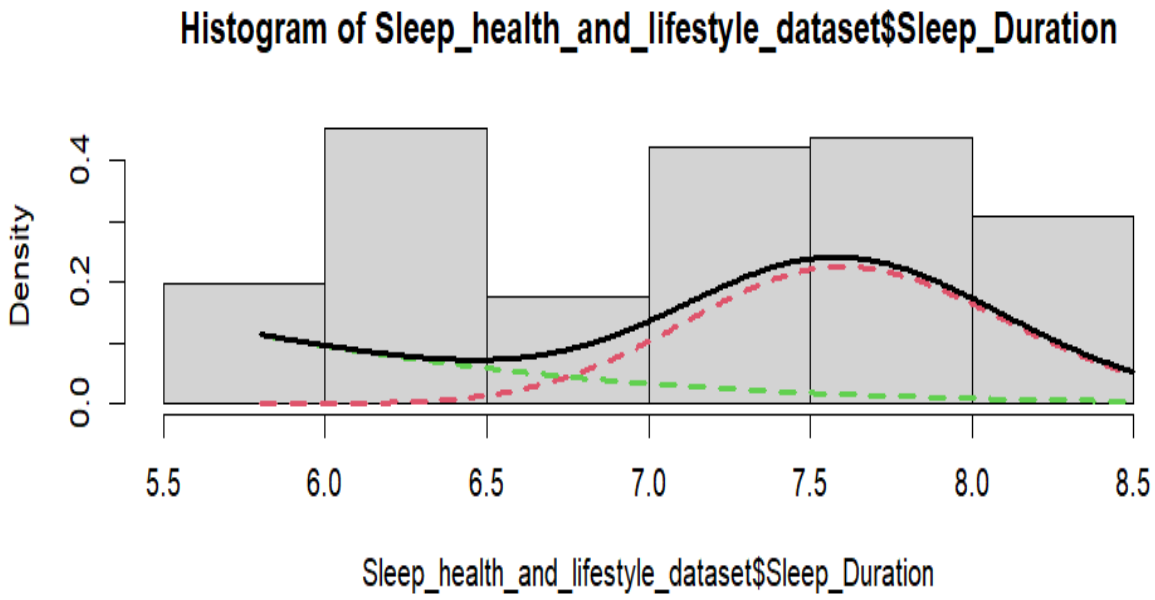
The AIC (Akaike Information Criterion) and SBC (Schwarz Bayesian Criterion) are both model selection criteria used to compare the goodness of fit of different statistical models. The lower the AIC and SBC values, the better the model is considered. When choosing a distribution for your data, you generally want to select the model with the lowest AIC or SBC, as it indicates the best balance between goodness of fit and model complexity. In this case, the Gamma distribution seems to be favored based on both AIC and SBC.

We can see that the Gamma distribution fits better the distribution. However we can try to fit the distribution using the gamma mixture model with k = 2

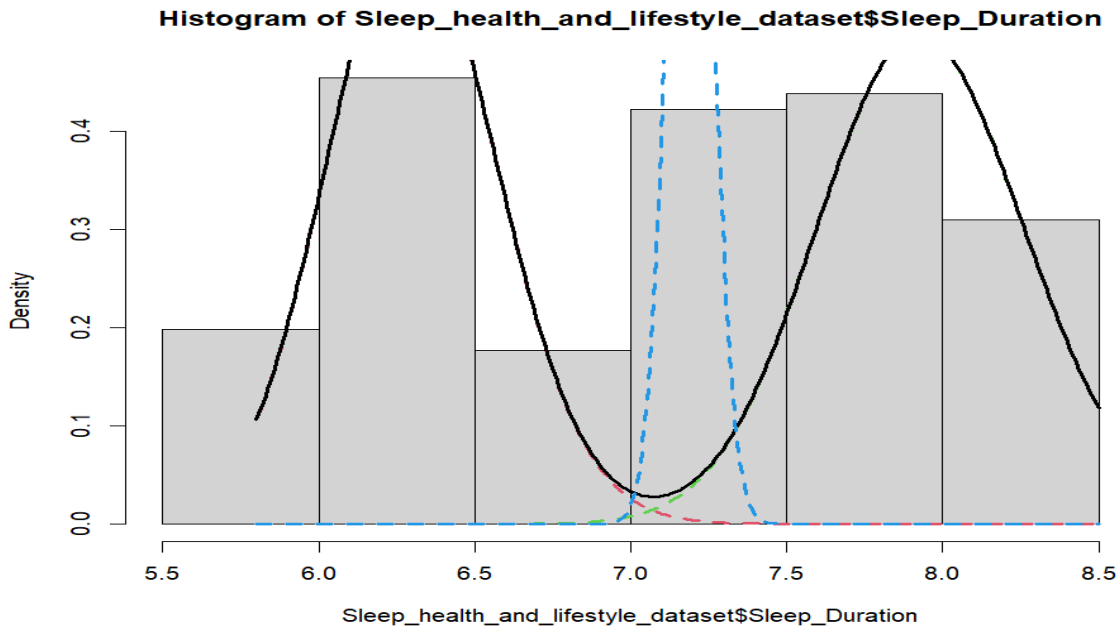
```
> fitted(fit.GA, "mu")[1]
[1] 7.132086
> fitted(fit.GA, "sigma")[1]
[1] 0.1117328
```

```
> hist(Sleep_health_and_lifestyle_dataset$Sleep_Duration, breaks = 5, freq =
FALSE)
>
> lines(seq(min(Sleep_health_and_lifestyle_dataset$Sleep_Duration), max(Sle
ep_health_and_lifestyle_dataset$Sleep_Duration), length = length(Sleep_heal
th_and_lifestyle_dataset$Sleep_Duration)),
+       fit.GA.SD.2[["prob"]][1] * dGA(seq(min(Sleep_health_and_lifestyle_datas
et$Sleep_Duration), max(Sleep_health_and_lifestyle_dataset$Sleep_Duration)
, length = length(Sleep_health_and_lifestyle_dataset$Sleep_Duration)),
+                                  mu = mu.hat1.SD, sigma = sigma.hat1.SD), lty = 2, l
wd = 3, col = 2)
>
> lines(seq(min(Sleep_health_and_lifestyle_dataset$Sleep_Duration), max(Sle
ep_health_and_lifestyle_dataset$Sleep_Duration), length = length(Sleep_heal
th_and_lifestyle_dataset$Sleep_Duration)),
+       fit.GA.SD.2[["prob"]][2] * dGA(seq(min(Sleep_health_and_lifestyle_datas
et$Sleep_Duration), max(Sleep_health_and_lifestyle_dataset$Sleep_Duration)
, length = length(Sleep_health_and_lifestyle_dataset$Sleep_Duration)),
+                                  mu = mu.hat2.SD, sigma = sigma.hat2.SD), lty = 2, l
wd = 3, col = 3)
>
> lines(seq(min(Sleep_health_and_lifestyle_dataset$Sleep_Duration), max(Sle
ep_health_and_lifestyle_dataset$Sleep_Duration), length = length(Sleep_heal
th_and_lifestyle_dataset$Sleep_Duration)),
```

```
+       fit.GA.SD.2[["prob"]][1] * dGA(seq(min(Sleep_health_and_lifestyle_datas
et$Sleep_Duration), max(Sleep_health_and_lifestyle_dataset$Sleep_Duration)
, length = length(Sleep_health_and_lifestyle_dataset$Sleep_Duration)),
+                                  mu = mu.hat1.SD, sigma = sigma.hat1.SD) +
+       fit.GA.SD.2[["prob"]][2] * dGA(seq(min(Sleep_health_and_lifestyle_dat
aset$Sleep_Duration), max(Sleep_health_and_lifestyle_dataset$Sleep_Durati
on), length = length(Sleep_health_and_lifestyle_dataset$Sleep_Duration)),
+                                  mu = mu.hat2.SD, sigma = sigma.hat2.SD),
+       lty = 1, lwd = 3, col = 1)
```




Now I try with K = 3




```
> data.frame(row.names=c("Gamma mixture with K=2", "Gamma mixture with K=3
"), AIC=c(fit.GA.SD.2$aic,fit.GA.SD.2$aic), SBC=c(fit.GA.SD.2$sbc,fit.GA.SD.2$sbc))
                              AIC      SBC
Gamma mixture with K=2 1378.785 1398.406
Gamma mixture with K=3 714.8982 746.2922
```

In summary, both AIC and SBC suggest that the Gamma mixture model with three components (K=3) provides a better fit to the data compared to the model with two components (K=2).

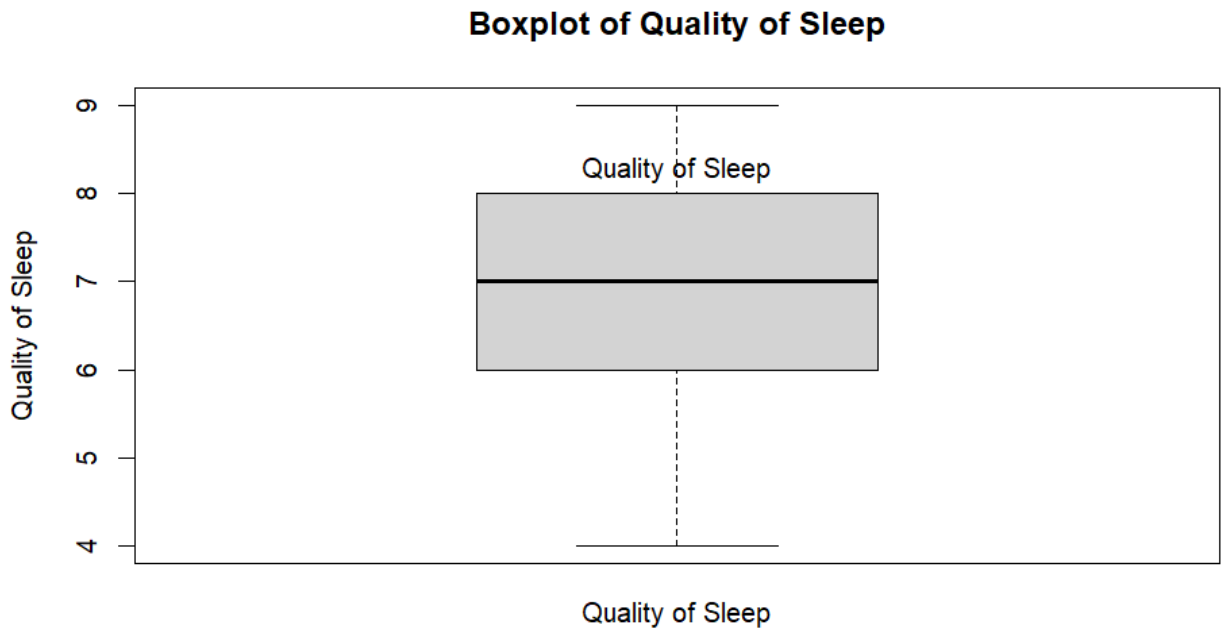
## 4. QUALITY OF SLEEP

This variable, "Quality_of_Sleep," is discrete random variable with a finite number of possible values (in this case, integers from 4 to 9). The values are not continuous but rather represent different categories or levels of sleep quality.The mean (average) of 7.313 suggests the central tendency of the variable, and the quartiles provide information about the distribution of the values. However, for discrete variables like this one, the concept of mean is a bit different from continuous variables. It represents the weighted average of the possible values.

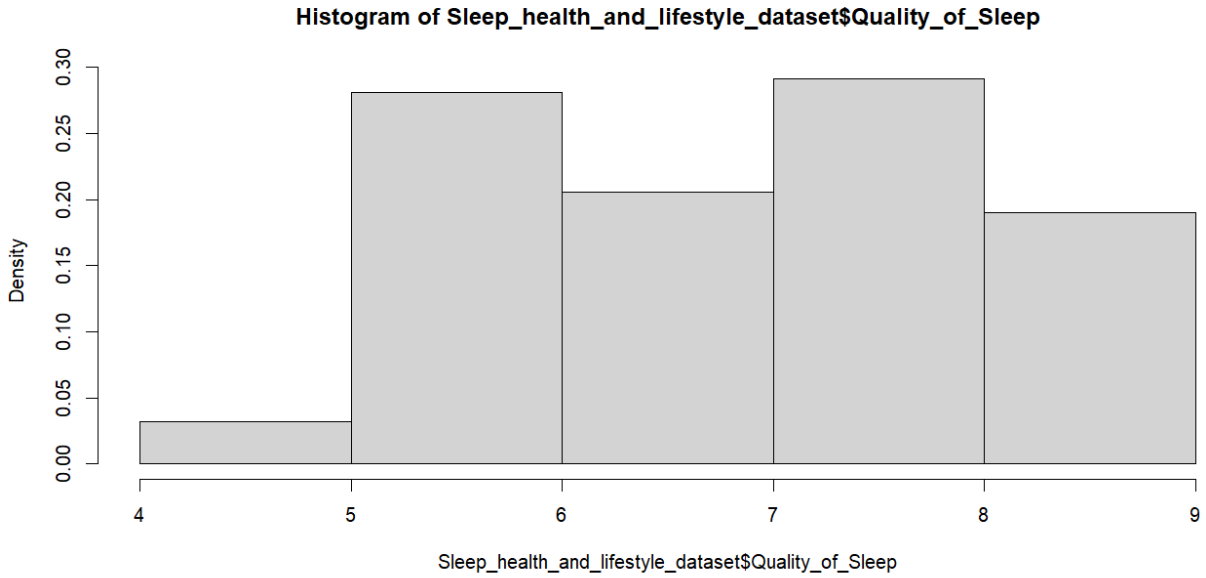
```
> summary(Sleep_health_and_lifestyle_dataset$Quality_of_Sleep)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.000   6.000   7.000   7.313   8.000   9.000
 > frequencyQuality_of_Sleep <- table(Sleep_health_and_lifestyle_dataset$Quality_of_Sleep)
 > frequencyQuality_of_Sleep

   4   5   6   7   8   9
   5   7 105  77 109  71


  > length(frequencyQuality_of_Sleep)
```

```
[1] 6


> names(frequencyQuality_of_Sleep)[frequencyQuality_of_Sleep == max (frequencyQuality_of_Sleep)]
[1] "8"
> labstatR::cv(Sleep_health_and_lifestyle_dataset$Quality_of_Sleep)
[1] 0.1634598
```

This variable takes on 6 values, ranging from 4 to 9, representing the duration of sleep.
The mode is determined using the names() function with the argument frequencySleep _ Duration.
The square brackets indicate that when the argument within these brackets is at its maximum,
the corresponding value is returned.
The coefficient of variation is approximately 0.16%.




```
> hist(Sleep_health_and_lifestyle_dataset$Quality_of_Sleep, breaks = 6, freq = FAL
SE)
```




## 5. PHYSICAL ACTIVITY LEVEL

The variable "Physical_Activity_Level" is a discrete random variable based on the provided summary statistics. The values are quantized and fall into specific categories, such as 30.00, 45.00, 60.00, etc. The fact that the values are distinct and not continuous suggests a discrete nature.

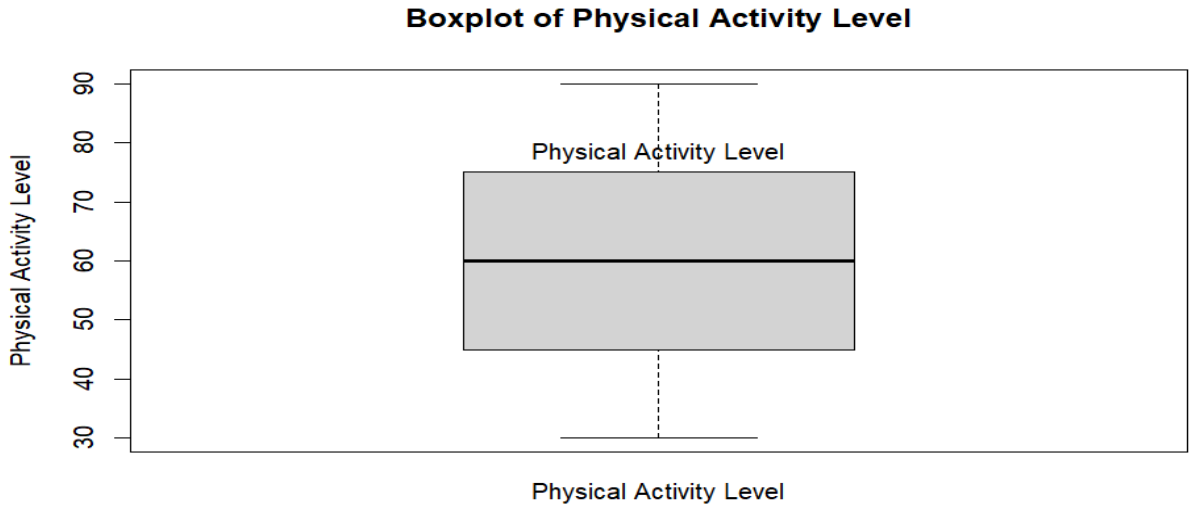
In particular, the values 30.00, 45.00, 60.00, 75.00, and 90.00 represent different levels or categories of physical activity rather than a continuous scale. Therefore, we can consider "Physical_Activity_Level" as a discrete variable in this context.

```
> summary(Sleep_health_and_lifestyle_dataset$Physical_Activity_Level)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  30.00   45.00   60.00   59.17   75.00   90.00
```

```
> frequencyPhysical <- table(Sleep_health_and_lifestyle_dataset$Physical_Activity_Level)
> frequencyPhysical

30 32 35 40 42 45 47 50 55 60 65 70 75 80 85 90
68  2  4  6  2 68  1  4  6 70  2  3 67  2  2 67
```

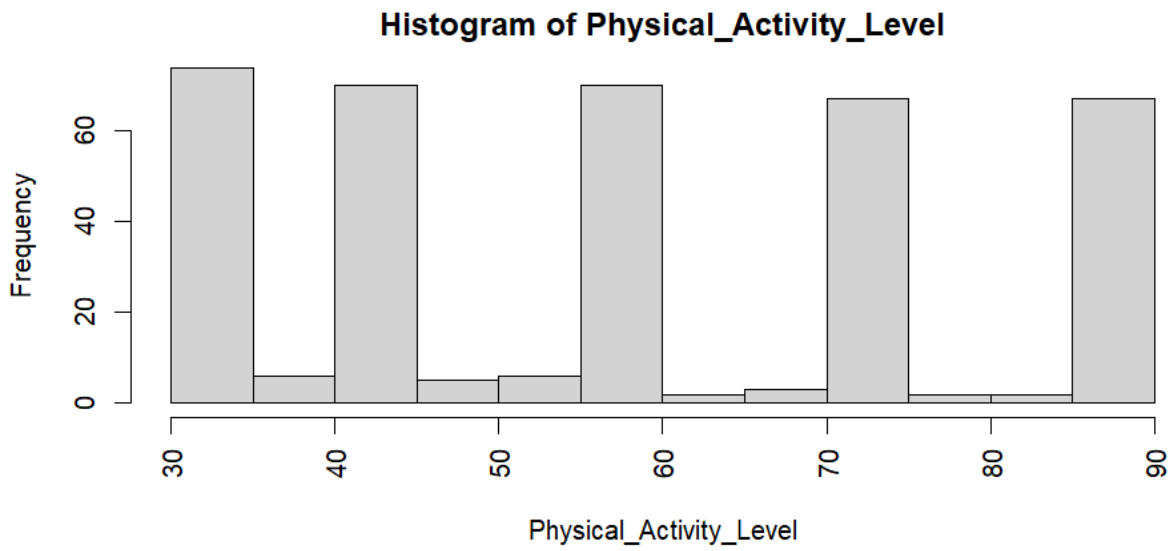
```
> length(frequencyPhysical)
[1] 16
> names(frequencyPhysical[frequencyPhysical == max(frequencyPhysical)])
[1] "60"
> labstatR::cv(Sleep_health_and_lifestyle_dataset$Physical_Activity_Level)
[1] 0.3515724
```

This variable assumes 16 values , from 30 and 90. The mode is the value 60.The modal values are found using the names()function .I passed as argument the frequencyPhysical; I used the square brackets for saying that when the argument of these brackets it's the maximum, return this value. The coefficent variation is about 0.35%.




Above I present the boxplot of Physical Activity Level . There isn't any outlier.

```
> hist(Sleep_health_and_lifestyle_dataset$Physical_Activity_Level, breaks = 16, fre
q = FALSE)
```

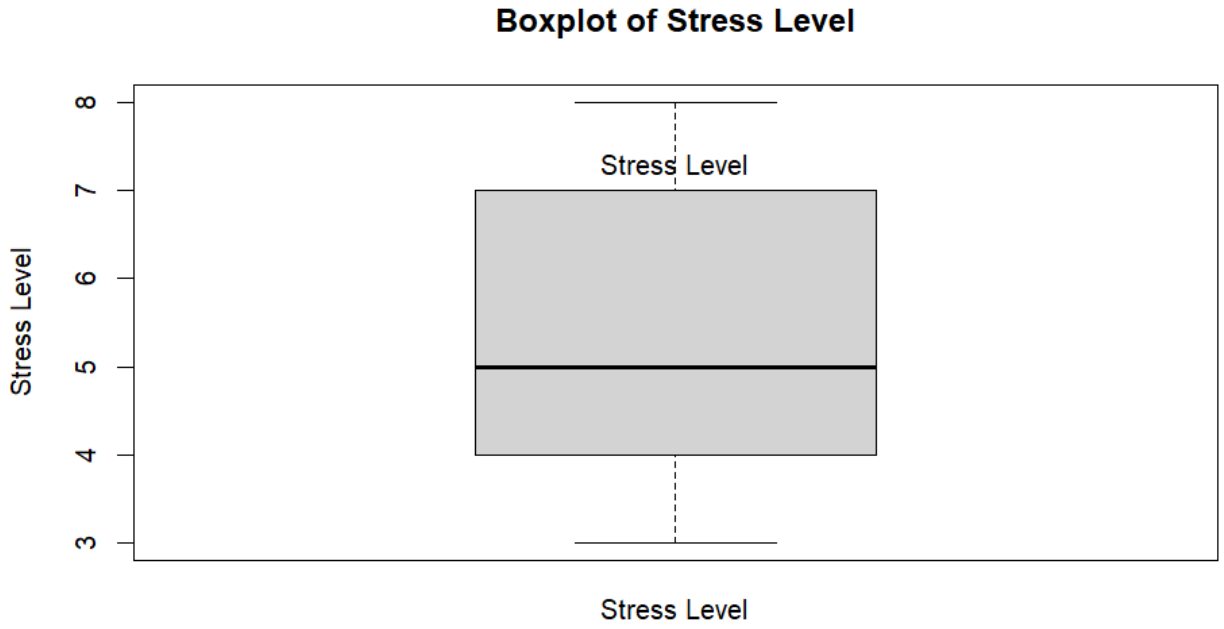



## 6. STRESS LEVEL

"Stress_Level" is a discrete variable:

```
summary(Sleep_health_and_lifestyle_dataset$Stress_Level)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.000   4.000   5.000   5.385   7.000   8.000

 > frequencyStressLevel <- table(Sleep_health_and_
 lifestyle_dataset$Stress_Level)
 > frequencyStressLevel

  3  4  5  6  7  8
 71 70 67 46 50 70
 > length(frequencyStressLevel)
 [1] 6
 > names(frequencyStressLevel)[frequencyStressLe
 vel == max(frequencyStressLevel)]
 [1] "3"
 > labstatR::cv(Sleep_health_and_lifestyle_dataset$
 Stress_Level)
 [1] 0.3290889
```

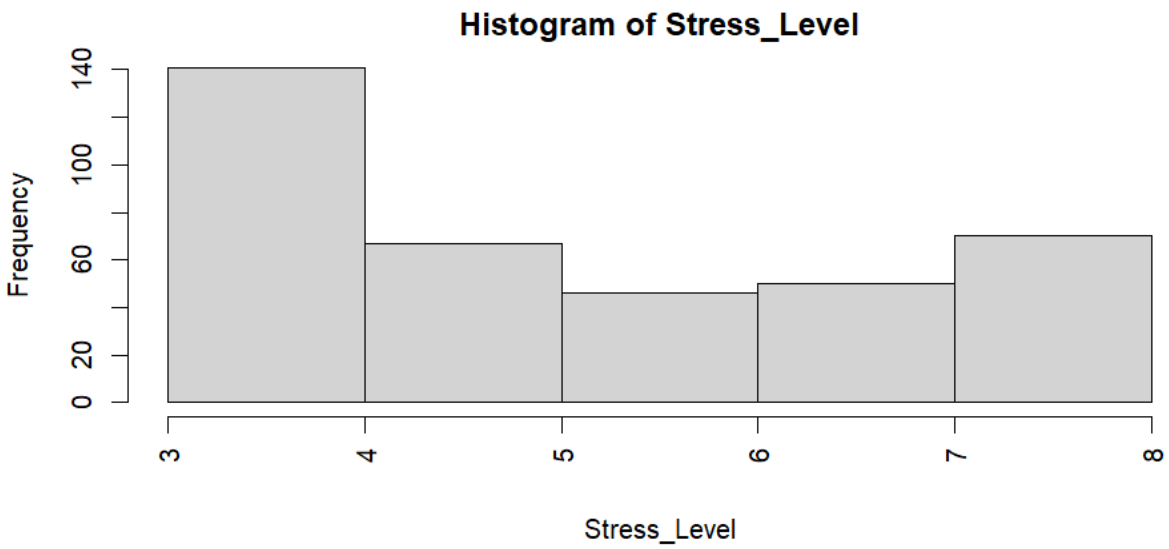
This variable assumes 6 values from 3 to 8 . The mode is equal to 3.
The coefficent of variation is about 0.32%




The boxplot representation of Stress Level says that there isn't any outliers in the distribution.

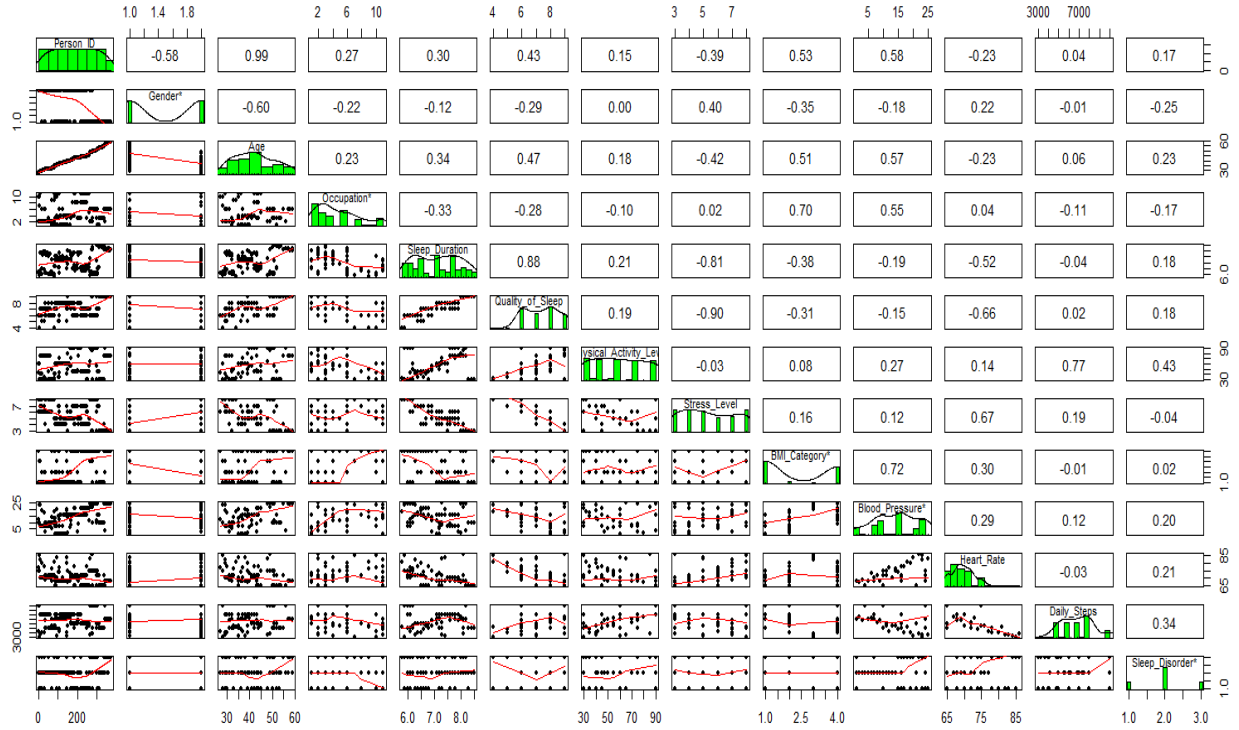
Now we can find the best model to fit ths distribution.

```
> hist(Stress_Level, breaks = 6)
```


Histogram of Stress_Level
Frequency
0
20
60
100
140
3
4
5
6
7
8
Stress_Level

# PRELIMINARY ANALYSIS

```
> plot <- pairs.panels(Sleep_subset, hist.c
ol="green", density = TRUE, ellipses = FAL
SE)
```




This serves as an initial analysis of the data within its original framework, with the goal of assessing the potential utility of conducting both Principal Component Analysis (PCA) and Cluster Analysis on the dataset. In the upper section of the matrix, one finds correlation coefficients among variables, serving to ascertain the relevance of PCA. Conversely, the lower section exhibits scatterplots of the data, with the main diagonal showcasing the non-parametric density of the data, collectively helping to evaluate the potential usefulness of Cluster Analysis. Notably, upon inspection of the plot, evident is a substantial correlation among certain variables. For instance, there exists a correlation of 0.88 between Sleep Duration and Quality of Sleep, 0.70 between Occupation and BMI Category, and 0.77 between Physical Activity and Daily Steps. This implies that PCA in this dataset holds considerable potential, allowing the creation of a linear amalgamation of variables expressed through a singular variable.

---

# PRINCIPAL COMPONENT ANALYSIS

Principal component analysis (PCA) enables us to condense and visualize the information in a dataset comprising statistical units characterized by numerous correlated quantitative variables. Each variable could be perceived as a distinct dimension. Analyzing the reference dataset thoroughly would demand a considerable amount of time, as it would involve evaluating 6 scatterplots, given the formula d(d-1)/2, where d is the number of variables (in this case, 4).Hence, PCA offers a method to discover a condensed representation of 'my_dataset,'

enabling a summary using a reduced set of key variables. These variables effectively capture the majority of the variability present in the original dataset characterized by 4 variables. Below I show the resulting in different values of mean and variance:

```
> round(apply(Sleep_subset, 2, mean), 4)
      Sleep_Duration      Quality_of_Sleep Physical_Activity_Level         Stress_Level
              7.1321               7.3128               59.1711             5.3850
> round(apply(Sleep_subset, 2, var), 4)
      Sleep_Duration      Quality_of_Sleep Physical_Activity_Level         Stress_Level
              0.6331               1.4327              433.9224             3.1489
```

Quality of Sleep,' 'Sleep Duration,' and 'Stress Level' exhibit different scales compared to 'Physical Activity Level.' In this scenario, the data related to 'Physical Activity Levels' appears to be more variable, although this observation is considerably unreliable due to the substantial difference in variance compared to the other features.

Because PCA is influenced by scaling and it is undesirable for the principal components to rely on the arbitrary scaling choice for individual variables, each variable was normalized to have a standard deviation equal to 1.

```
> df.scaled <- scale(Sleep_subset)
> head(round(df.scaled, digits = 3))
     Sleep_Duration Quality_of_Sleep Physical_Activity_Level Stress_Level
[1,]         -1.297          -1.097                 -0.824        0.347
[2,]         -1.171          -1.097                  0.040        1.474
[3,]         -1.171          -1.097                  0.040        1.474
[4,]         -1.549          -2.768                 -1.400        1.474
[5,]         -1.549          -2.768                 -1.400        1.474
[6,]         -1.549          -2.768                 -1.400        1.474
```

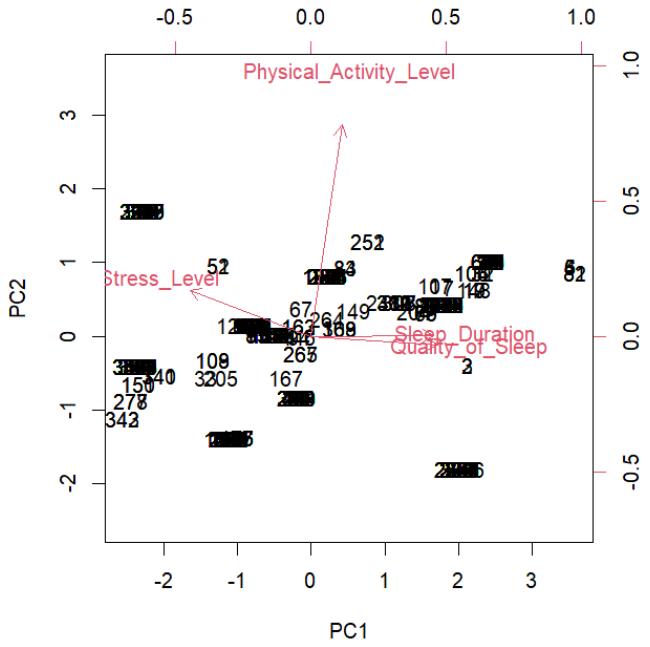
Principal components of the standardized variables can then be obtained by doing the eigen decomposition of the sample correlation matrix.

```
> cor_matrix <- cor(df.scaled)
> head(round(cor_matrix, digits = 3))
                     Sleep_Duration Quality_of_Sleep Physical_Activity_Level
Sleep_Duration                1.000          0.883                  0.212
Quality_of_Sleep              0.883          1.000                  0.193
Physical_Activity_Level       0.212          0.193                  1.000
Stress_Level                 -0.811         -0.899                 -0.034
                     Stress_Level
Sleep_Duration             -0.811
Quality_of_Sleep           -0.899
Physical_Activity_Level    -0.034
Stress_Level                1.000
> eigen <- eigen(cor_matrix)
> phi <- eigen$vectors
> head(round(phi, digits = 2))
      [,1]  [,2]  [,3]  [,4]
[1,] -0.57  0.01  0.77 -0.28
[2,] -0.59 -0.04 -0.14  0.80
```

```
[3,] -0.14  0.98 -0.14 -0.08
[4,]  0.56  0.21  0.60  0.52
> phi <- -phi
> row.names(phi) <- colnames(my_dataset) # rename rows in phi matrix
> row.names(phi) <- colnames(Sleep_subset) # rename rows in phi matrix
>  colnames(phi) <- c("PC1", "PC2", "PC3", # rename columns in phi matrix
+                    + "PC4", "PC5", "PC6")
> row.names(phi) <- colnames(Sleep_subset) # rename rows in phi matrix
>  colnames(phi) <- c("PC1", "PC2", "PC3", # rename columns in phi matrix
+                    + "PC4", "PC5")
> row.names(phi) <- colnames(Sleep_subset) # rename rows in phi matrix
>  colnames(phi) <- c("PC1", "PC2", "PC3", # rename columns in phi matrix)
+  head(round(phi, digits = 2))
+ head(round(phi, digits = 2))
> head(round(phi, digits = 2))
                             PC1   PC2   PC3   PC4
Sleep_Duration              0.57 -0.01 -0.77  0.28
Quality_of_Sleep            0.59  0.04  0.14 -0.80
Physical_Activity_Level     0.14 -0.98  0.14  0.08
Stress_Level               -0.56 -0.21 -0.60 -0.52
```

We can use the Biplot to rapresent at the same time original variables, principal components and the same unit like the PC Scores.

```
> biplot(pr.out, scale = 0)
```

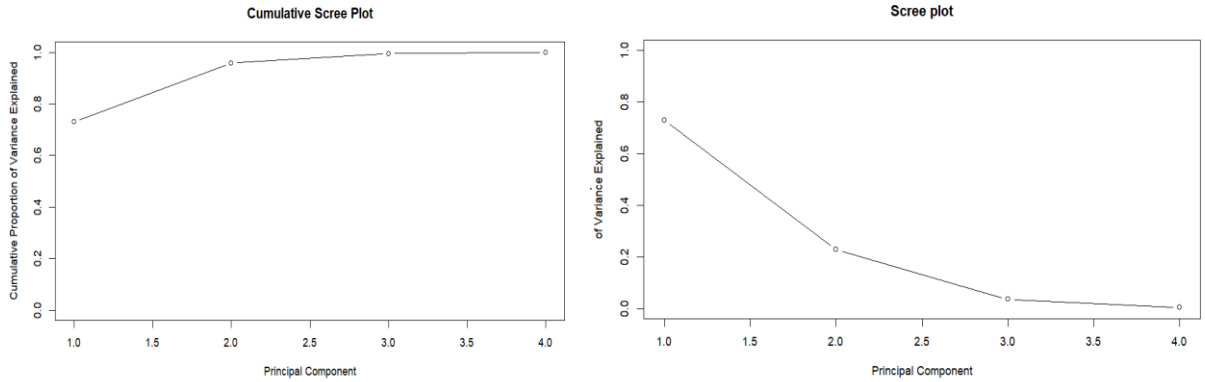



## Cumulative proportion of variance explained

```
> PVE <- eigen$values/sum(eigen$values)
>  (PVE <- round(PVE, 2))
[1] 0.69 0.25 0.04 0.02
> cumsum(PVE)
[1] 0.69 0.94 0.98 1.00
```

Computing the proportion of variance explained by each eigenvector, it results th at: - The first principal component explains the 69% of the variability; - The secon d principal component explains the 25% of the variability. - The third principal co mponent explains the 4% of the variability. It follows that, together, PC1 and PC2 a re able to explain the 94% of variability, which is a good proportion,. Therefore, al though the choice of only two principal components could also be reasonable, it i s certainly preferable reducing the dimensionality of 'Sleep_subset' selecting the fi rst two principal components. This conclusion is also reinforced by visualizing th e scree plot. In fact, looking at the scree plot on the left panel below, it is clearly n oticeable the significant jump that occurs in correspondence of the value m = 1, w hich could be easily identified in the first place as the "elbow point". However, the curve does not immediately tend to flatten itself, which instead seems to happen after m = 2

```
> plot(pve, main = "Scree plot", xlab="Principal Component", ylab="Proportion
+ of Variance Explained", ylim=c(0,1), type='b')
>  plot(cumsum(pve), xlab="Principal Component",  ylab="Cumulative Proportion of Variance Explained",
  ylim=c(0,1),type='b', main= "Cumulative Scree Plot")
```




# CLUSTER ANALYSIS

We'll use only a subset of the data by taking 15 random rows among the 374 rows in the data set. This is done by using the function sample(). Next, we standardize the data using the function scale()

```
> ss <- sample(1:50, 15)
> df <- Sleep_data[ss, ]
> df.scaled <- scale(df)
```

```
> df.scaled <- scale(df)
> df <- Sleep_data[ss, ]
> df.scaled <- scale(df)
```

Now we can compute the Euclidian compute with dist() Function

```
> #computing euclidian distance
> dist.eucl <- dist(df.scaled, method ="euclidian")
> #Reformat as a matrix
> #subset the first 5 columns and rows and round the values
>
> round(as.matrix(dist.eucl)[1:5, 1:5], 2)
                                    Nurse Doc. Doc. Doc. Engineer
Nurse    0.00 1.60 1.60 1.94 3.43
Doctor 1.60 0.00 0.00 1.44 3.74
Doctor 1.60 0.00 0.00 1.44 3.74
 Doctor  1.94 1.44 1.44 0.00 3.01
Engineer 3.43 3.74 3.74 3.01 0.00
```

In this Symmetrix matrix , each value rapresent the distanc between units. The val ues on the diagonal represent the distance between units and themeself, which is zero.

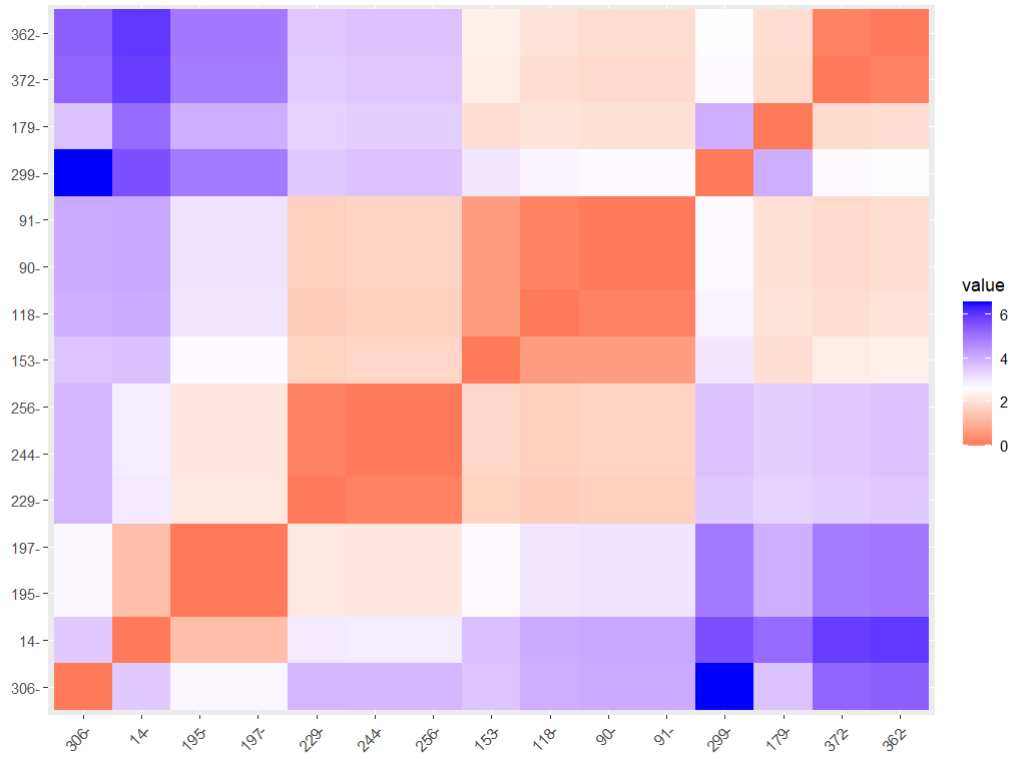
The function daisy() provides solution(Gower's distance) for computing the ddista nce matrix , In the situation when the data containing non numeric columns.

```
> library(cluster)
>
> data(Sleep__data)
> head(Sleep_data, 5)
         Occupation       Sleep_Duration Stress_Level Physical_Activity_Level Quality_of_Sleep
1    Software Engineer    6.1            6            42                      6
2              Doctor     6.2            8            60                      6
3              Doctor     6.2            8            60                      6
4 Sales Representative    5.9            8            30                      4
5 Sales Representative    5.9            8            30                      4
> #Data structure
> str(Sleep_data)
'data.frame':          374 obs. of  5 variables:
 $ Occupation            : chr  "Software Engineer" "Doctor" "Doctor" "Sales Representative" ...
 $ Sleep_Duration        : num  6.1 6.2 6.2 5.9 5.9 5.9 6.3 7.8 7.8 7.8 ...
 $ Stress_Level          : int  6 8 8 8 8 8 7 6 6 6 ...
 $ Physical_Activity_Level: int  42 60 60 30 30 30 40 75 75 75 ...
 $ Quality_of_Sleep      : int  6 6 6 4 4 4 6 7 7 7 ...
>
>
> #distance Matrix
> dd <- daisy(Sleep_data)
>
> # Display the first few rows of the new dataframe
> head(Sleep_data)
  Sleep_Duration Stress_Level Physical_Activity_Level Quality_of_Sleep
```

```
1         6.1         6                42          6
2         6.2         8                60          6
3         6.2         8                60          6
4         5.9         8                30          4
5         5.9         8                30          4
6         5.9         8                30          4
>
> dd <- daisy(Sleep_data)
> round(as.matrix(dd)[1:5, 1:5], 2)
     1     2     3     4     5
1  0.00 18.11 18.11 12.33 12.33
2 18.11  0.00  0.00 30.07 30.07
3 18.11  0.00  0.00 30.07 30.07
4 12.33 30.07 30.07  0.00  0.00
5 12.33 30.07 30.07  0.00  0.00
```

And now we can visualize the distance matrix using the function fvuz_dist()

```
> library(factoextra)
> fviz_dist(dist.eucl)
```




The color level is proportional to the value of the dissimilarity between observatio n.Red indicates high similarity and blue indicates low similarity.

```
> # Display the first few rows of the new dataframe
> head(Sleep_data)
        Sleep_Duration Stress_Level Physical_Activity_Level  Quality_of_Sleep
Engi.          6.1            6                  42                6
Doc.           6.2            8                  60                6
Doc.           6.2            8                  60                6
Sal.Rapr.      5.9            8                  30                4
Sal.Rapr.      5.9            8                  30                4
Engi.          5.9            8                  30                4
>
> dd <- daisy(Sleep_data)
> round(as.matrix(dd)[1:5, 1:5], 2)
     1     2     3     4     5
1  0.00 18.11 18.11 12.33 12.33
2 18.11  0.00  0.00 30.07 30.07
3 18.11  0.00  0.00 30.07 30.07
4 12.33 30.07 30.07  0.00  0.00
5 12.33 30.07 30.07  0.00  0.00
```

#Stanardize The Data

```
>  df <- scale(Sleep_data)
>  head(df, nrow =6)
     Sleep_Duration Stress_Level Physical_Activity_Level Quality_of_Sleep
[1,]     -1.297149    0.3465563          -0.82431400       -1.096811
[2,]     -1.171467    1.4736175           0.03979093       -1.096811
[3,]     -1.171467    1.4736175           0.03979093       -1.096811
[4,]     -1.548514    1.4736175          -1.40038394       -2.767716
[5,]     -1.548514    1.4736175          -1.40038394       -2.767716
[6,]     -1.548514    1.4736175          -1.40038394       -2.767716
```

#Compute the dissimilarity Matrix

```
> res.dist <- dist(df, method ="euclidian")
> as.matrix(res.dist)[1:6, 1:6]
        1        2        3        4        5        6
1 0.000000 1.425742 1.425742 2.111216 2.111216 2.111216
2 1.425742 0.000000 0.000000 2.237899 2.237899 2.237899
3 1.425742 0.000000 0.000000 2.237899 2.237899 2.237899
4 2.111216 2.237899 2.237899 0.000000 0.000000 0.000000
5 2.111216 2.237899 2.237899 0.000000 0.000000 0.000000
6 2.111216 2.237899 2.237899 0.000000 0.000000 0.000000
```

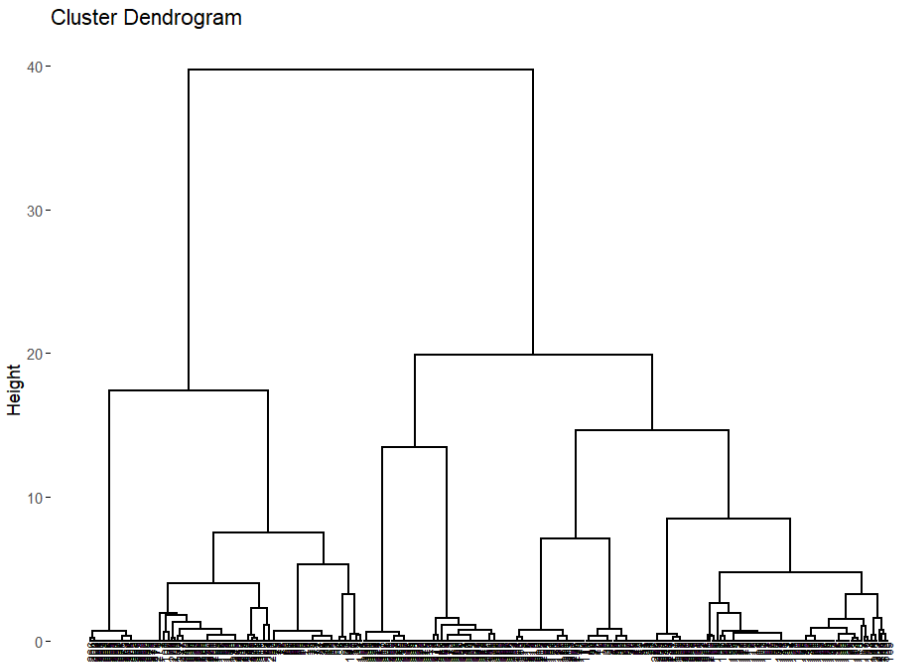
If you have a distance matrix stored in the "res.dist" object generated by the dist() function, you can utilize the R function hclust() to construct the hierarchical tree. The usage of hclust() is outlined below:

- **d**: A dissimilarity structure generated by the dist() function.
- **method**: The agglomeration (linkage) method used for computing distances between clusters. The allowed values include "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median", or "centroid". As mentioned earlier, average linkage and Ward's method are generally preferred.

A noteworthy point is that the only distinction between ward.D and ward.D2 lies in the distance matrix provided as an input to hclust(). Specifically, hclust(dist(x)2, method = "ward.D") is equivalent to hclust(dist(x), method = "ward.D2"). Therefore, using method = "ward.D2" is typically recommended!

The dendrogram can be produced in R using the base function plot(res.hc), where res.hc is the output of hclust(). Here, we'll use the function fviz dend() (in the fact oextra package) to produce a beautiful dendrogram.

```
> #Dendogram
> library("factoextra")
> fviz_dend(res.hc, cex = 0.5)
```



The cophenetic dissimilarity or cophenetic distance of two units is a measure of h ow similar those two units have to be in order to be grouped into the same cluster . From a practical point of view, The cophenetic distance between two units is the height of the dendrogram where the two branches that include the two units merg e into a single branch (height of the fusion).
The higher the height of the fusion, the less similar the units are.

```
> #Compute cophenetic distance
>
> res.coph <- cophenetic(res.hc)
>
> #Correltion between cophenetic and original distance
>
> cor(res.dist, res.coph)
[1] 0.7799441
>
>
>
```

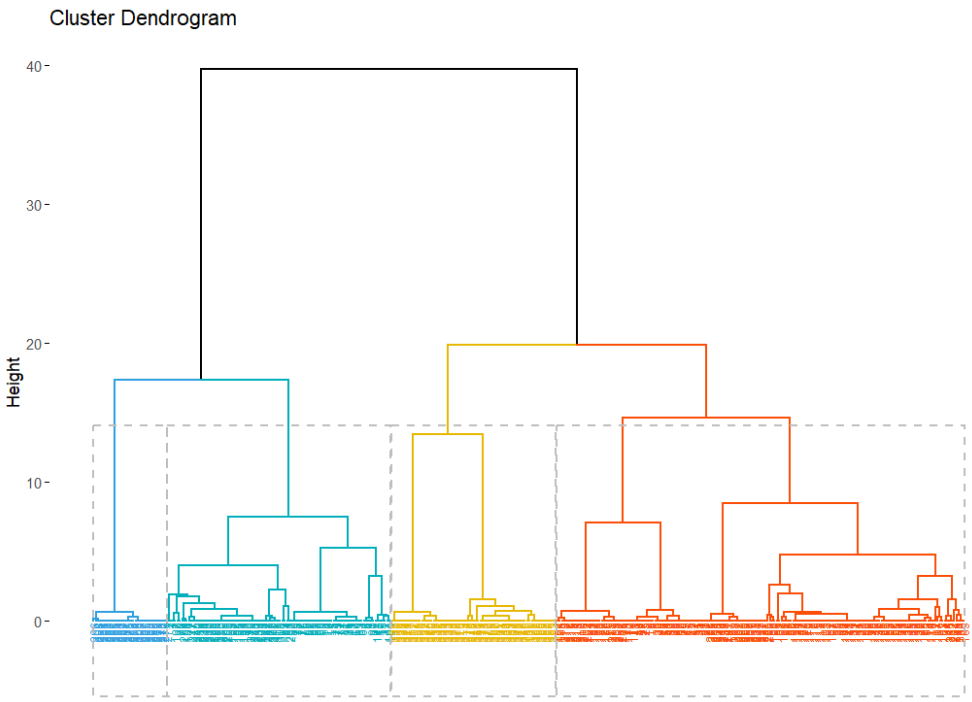
We can execute the hclust() function again using the average linkage method. Next , call cophenetic() to evaluate the clustering solution.

```
> #Execute again with the average linkage mathod
> res.hc2 <- hclust(res.dist, method = "average")
> cor(res.dist, cophenetic(res.hc2))
[1] 0.8195883
```

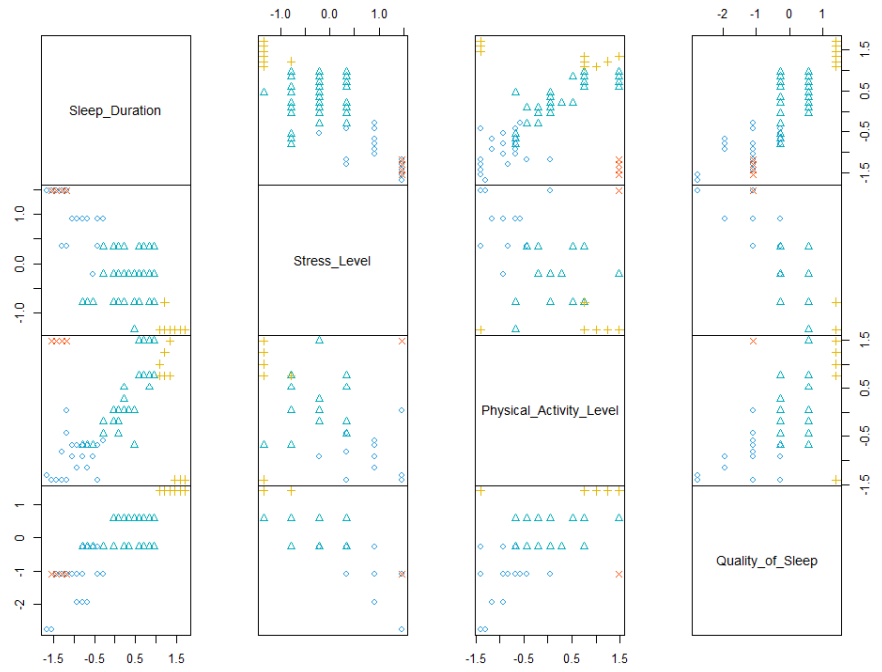
The correlation coefficient shows that using a different linkage method creates a t ree that represents the original distances slightly better.

The result of the cuts can be visualized easily using the function fviz_dend():

```
> table(grp)
grp
  1   2   3   4
 96 175  71  32
> grp <- cutree(res.hc, k = 4)
>
> # Assuming 'data' is the original data used for clustering
> grp <- cutree(res.hc, k = 4)
>
> fviz_dend(res.hc, k = 4,
+           cex = 0.5,                # label size
+           k_colors = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),
+           color_labels_by_k = TRUE, # color label by group
+           rect = TRUE               # add rectangle around groups
```

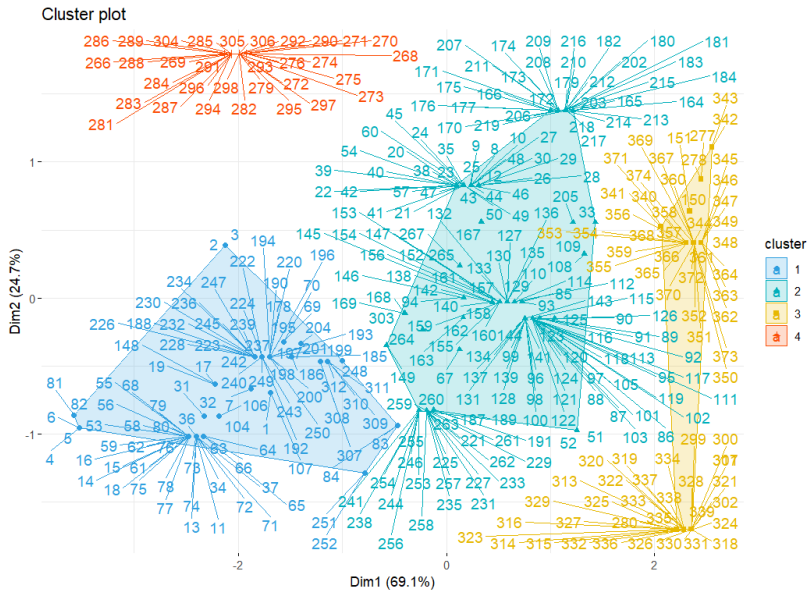



And we can also visualize these clustering result in the original space, via the mat rix of pairwise catterplots, using the pairs() function .



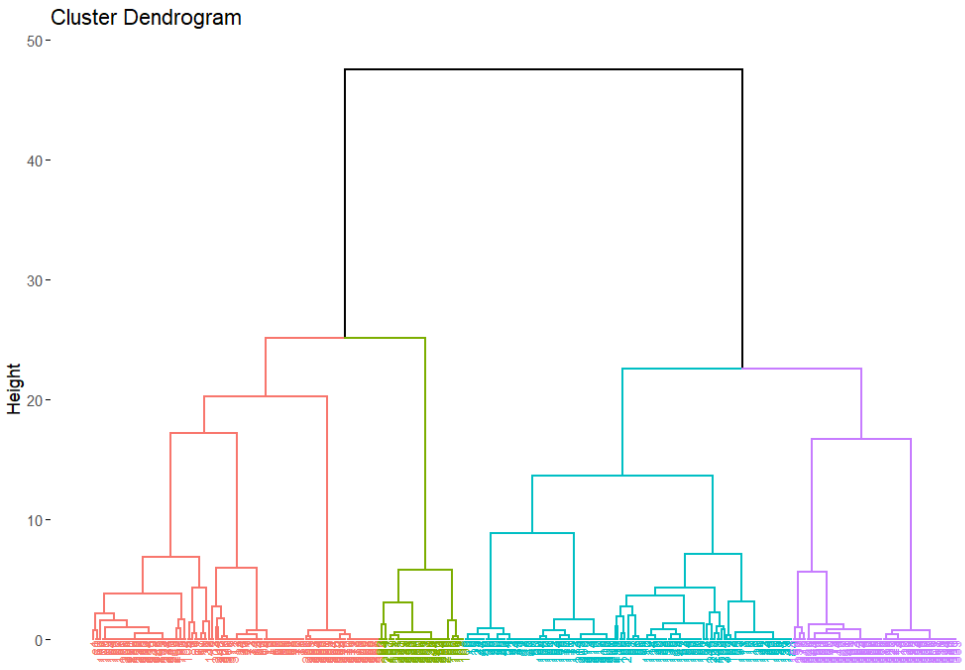
Using the function fviz_cluster(), we can also visualize the result in the catterplt(s pace) of the first 2 PCS:

```
> fviz_cluster(list(data=df, cluster = grp),
+ palette = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),
+ ellipse.type = "convex",
+ repel =TRUE, #Avoid label overplotting(slow)
+ show.clust.cent =FALSE, ggtheme =theme_minimal())
```




The package cluster makes it easier to perform cluster analysis in R. It provides th e functions agnes() and diana() for computing agglomerative and divisive clusterin g, respectively.

```
> library("cluster")
>
>
> #Agglomerative Nesting (Hierarchical Clustering)
> res.agnes <- agnes(x = Sleep_data, #data matrix
+ stand = TRUE, #Standardize the data
+ metric = "euclidian", #metric for distance matrix
+ method ="ward" #Linkage method
+ )
> # Divisive Analysis Clustering
> res.diana <- diana(x = Sleep_data, #data matrix)
+ res.diana <- diana(x = Sleep_data, #data matrix
+ stand = TRUE, #Standardize the data
+ metric = "euclidian" #metric for distance matrix
+ )
+
+
+ #Visualize th output
+ fviz_dend(res.agnes, cex = 0.6, k = 4)
```

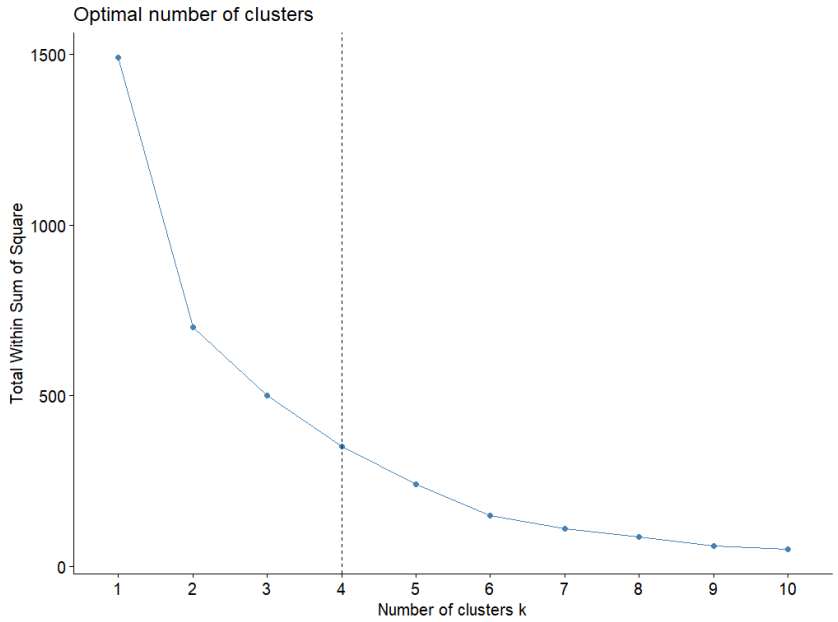



```
>
```

## K-Means:

The K-means clustering requires the users to specify the number of clusters K to be generated. Fundamental question How to select K? Various methods can be con sidered for this purpose. A straightforward approach Conduct K-means clustering using various values for K. Then, for each K, calculate the Within-Sum-of-Squares ( WSS). The position of a bend (knee or elbow) in the plot is typically regarded as an indicator of the suitable number of clusters.

```
> # Elbow method for k-means (look at the knee)
>
> library(factoextra)
> fviz_nbclust(df, kmeans, nstart=25, method = "wss")+
+ geom_vline(xintercept = 4, linetype = 2)
```




For ensuring reproducibility in the results of the K-means clustering algorithm, it is advised to utilize the set.seed() function to set a seed for R's random number generator. This practice guarantees that others can obtain precisely the same results presented in the article or report.

Since the outcome of K-means clustering is influenced by the initial random centroids, setting a seed is crucial. In this context, we set nstart = 25, signifying that R will experiment with 25 different random starting assignments and select the optimal outcome based on the lowest within-cluster variation (WSS).

Note: Although the default value is nstart = 1, it is strongly recommended to perform K-means clustering with a higher value for nstart, such as 25 or 50, to achieve a more robust and stable result.

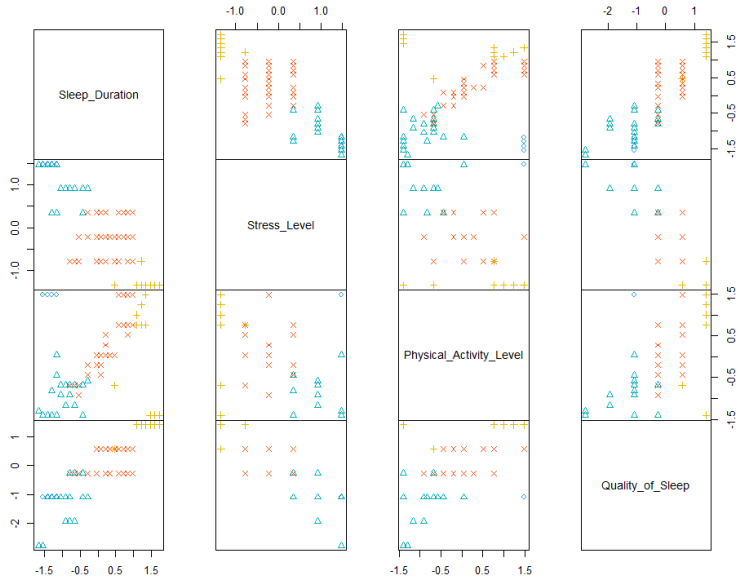
```
> set.seed(123)
> km.res <- kmeans(df, 4, nstart = 25)

> print(km.res)
K-means clustering with 4 clusters of sizes 32, 94, 73, 175

Cluster means:
  Sleep_Duration Stress_Level Physical_Activity_Level Quality_of_Sleep
1     -1.3403526    1.4736175               1.4799658       -1.0968108
2     -1.0618292    1.1019271              -0.9816097       -1.1679132
3      1.3714483   -1.3285964              -0.1870859        1.3866580
4      0.2433571   -0.3071393               0.3346839        0.2494614

Clustering vector:
  [1] 2 2 2 2 2 2 2 4 4 4 2 4 2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 4 4 2 2 4 2 4 2 2
 [38] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 3 2 4 2 2 4 2 2 4 2 2 2 2 2 2 4 2 2 2 2 2 2 2
 [75] 2 2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 4 2 2 4 4 4 4
[112] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2
[149] 4 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 4 4 4 4 4 4 2
[186] 2 4 2 4 2 4 2 2 2 2 2 2 2 2 2 2 4 4 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 4 2
[223] 2 2 4 2 4 2 4 2 4 2 4 2 4 2 2 4 2 2 4 2 2 4 2 4 2 2 2 2 2 2 4 4 4 4 4 4 4
[260] 4 4 4 4 4 4 1 4 1 1 1 1 1 1 1 1 1 3 3 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[297] 1 1 3 3 3 3 4 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[334] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[371] 3 3 3 3

Within cluster sum of squares by cluster:
[1]   0.2719882  58.0122958  96.9657014 195.8240458
 (between_SS / total_SS =  76.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

The kmeans() function returns a list of components, including: cluster: A vector of integers (from 1 to K) indicating the cluster to which each point is allocated; centers: A matrix of cluster centers (cluster means); totss: The total sum of squares (TSS) measuring the total deviance in the data; withinss: Vector of within-cluster sum of squares (SS), one component per cluster; tot.withinss: Total within-cluster sum of squares (WSS), i.e. sum(withinss); betweenss: The between-cluster sum of squares (BSS), i.e. totss - tot.withinss; size: A vector of number of observations in each cluster.

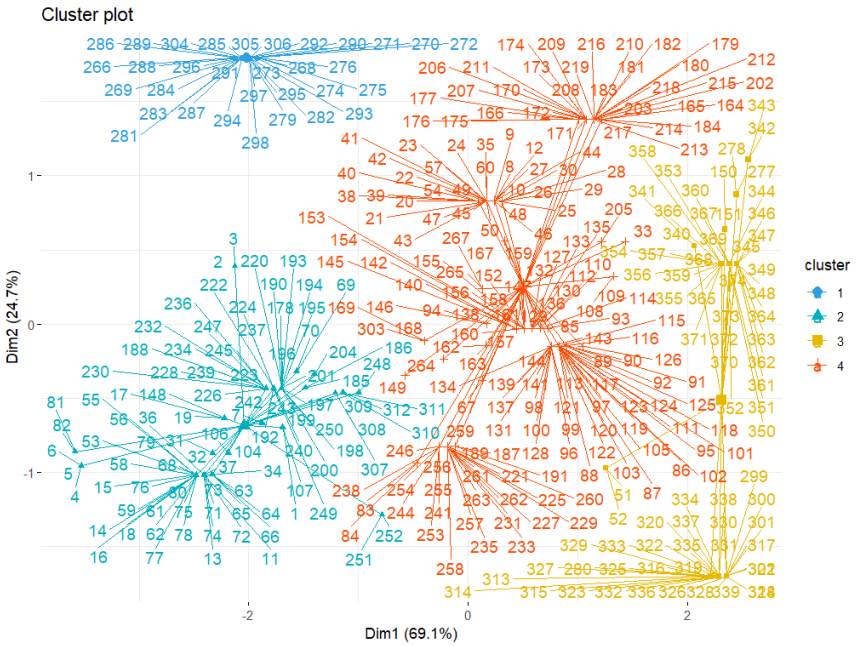
```
> #Accessing to the result of kmean()function
>
> #cluster number of each of the observations
>
> km.res$cluster
  [1] 2 2 2 2 2 2 2 4 4 4 2 4 2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 4 4 2 2 4 2 4 2 2
 [38] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 3 2 4 2 2 4 2 2 4 2 2 2 2 2 2 4 2 2 2 2 2 2 2
 [75] 2 2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 4 2 2 4 4 4 4
[112] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2
[149] 4 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 4 4 4 4 4 4 2
[186] 2 4 2 4 2 4 2 2 2 2 2 2 2 2 2 2 4 4 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 4 2
[223] 2 2 4 2 4 2 4 2 4 2 4 2 4 2 2 4 2 2 4 2 2 4 2 4 2 2 2 2 2 2 4 4 4 4 4 4 4 4
[260] 4 4 4 4 4 4 1 4 1 1 1 1 1 1 1 1 1 3 3 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[297] 1 1 3 3 3 3 4 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[334] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[371] 3 3 3 3
>
>
> #cluster size
>
> km.res$size
[1]  32  94  73 175
>
> #cluster means
>
> km.res$centers
  Sleep_Duration Stress_Level Physical_Activity_Level Quality_of_Sleep
1     -1.3403526    1.4736175               1.4799658       -1.0968108
2     -1.0618292    1.1019271              -0.9816097       -1.1679132
3      1.3714483   -1.3285964              -0.1870859        1.3866580
4      0.2433571   -0.3071393               0.3346839        0.2494614
```

Visualizing cluster results can be useful to assess the choice of the number of clu sters as well as to compare two different cluster analyses. In the original space we have:

```
> #Visualizing k-means clusters in the original space
>
> cl <- km.res$cluster
> pairs(df, gap = 0, pch = cl, col = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E0
7")[cl])
```



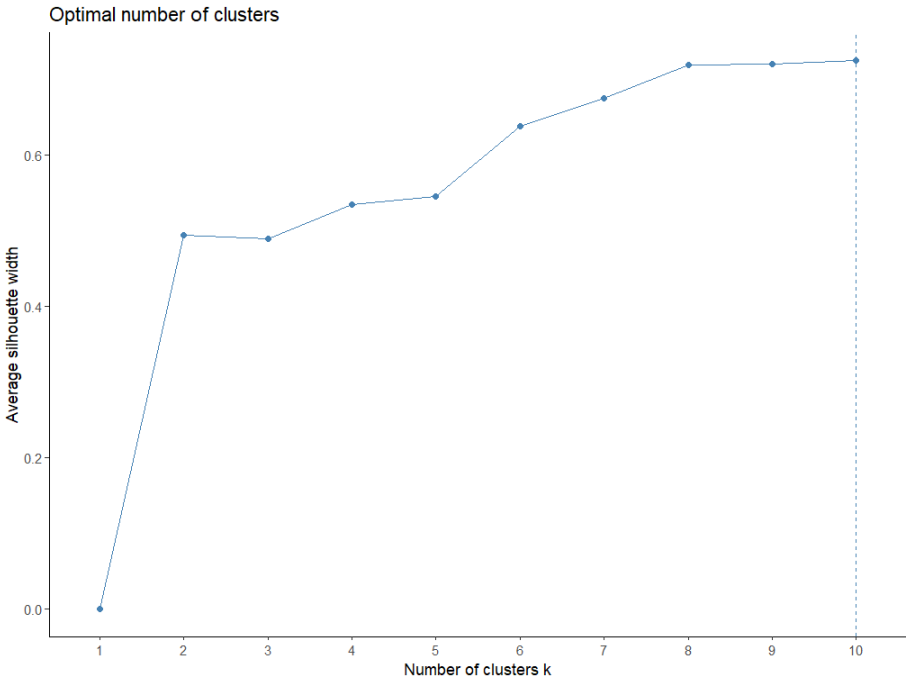
```
> # Visualizing K-means clusters
> fviz_cluster(
+     km.res,
+     data = df,
+     palette = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),
+     ellipse.type = "euclidian",  # Concentration Ellipse
+     star.plot = TRUE,  # Add segments from centroids to items
+     repel = TRUE,  # Avoid label overplotting (slow)
+     ggtheme = theme_minimal()
+ )
```



## K-Medoids

Choosing K: the silhouette method The K-medoids algorithm requires the user to specify K, the number of clusters to be generated (like in K-means clustering). A u seful approach to determine the optimal number of clusters is the silhouette meth od. Aim The silhouette value, for each unit, is a measure of how similar a unit is t o its own cluster (cohesion) compared to other clusters (separation).

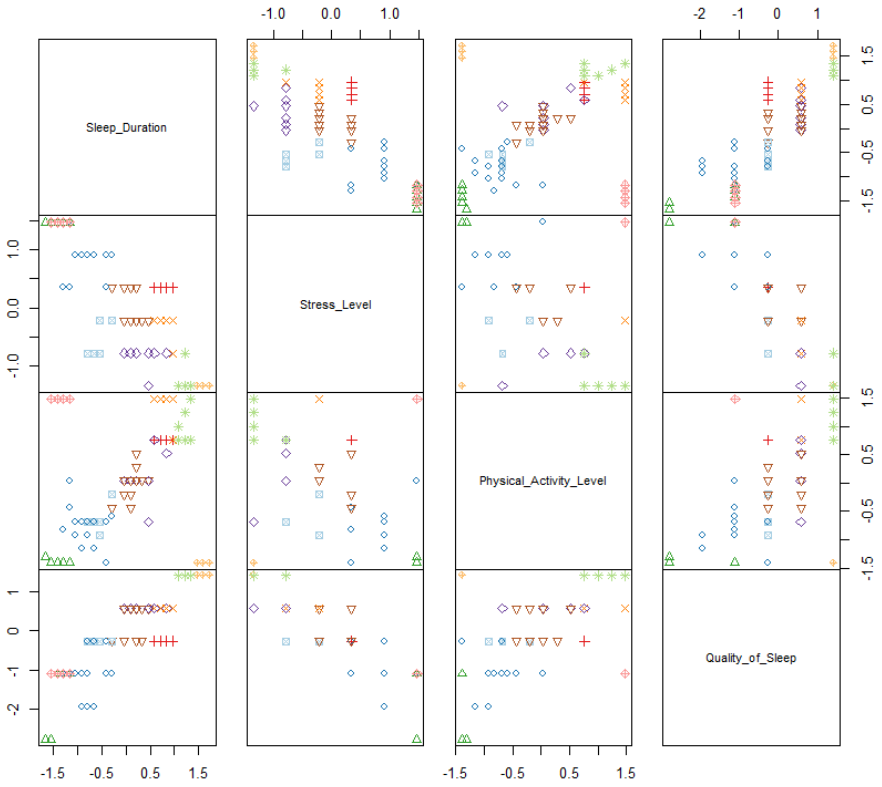
```
> fviz_nbclust(df, pam, method = "silhouette")+
+ theme_classic()
> # Silhouette method for PAM (look at the maximum)
> library(cluster)
> library(factoextra)
>
> fviz_nbclust(df, pam, method = "silhouette") +
+     theme_classic()
```



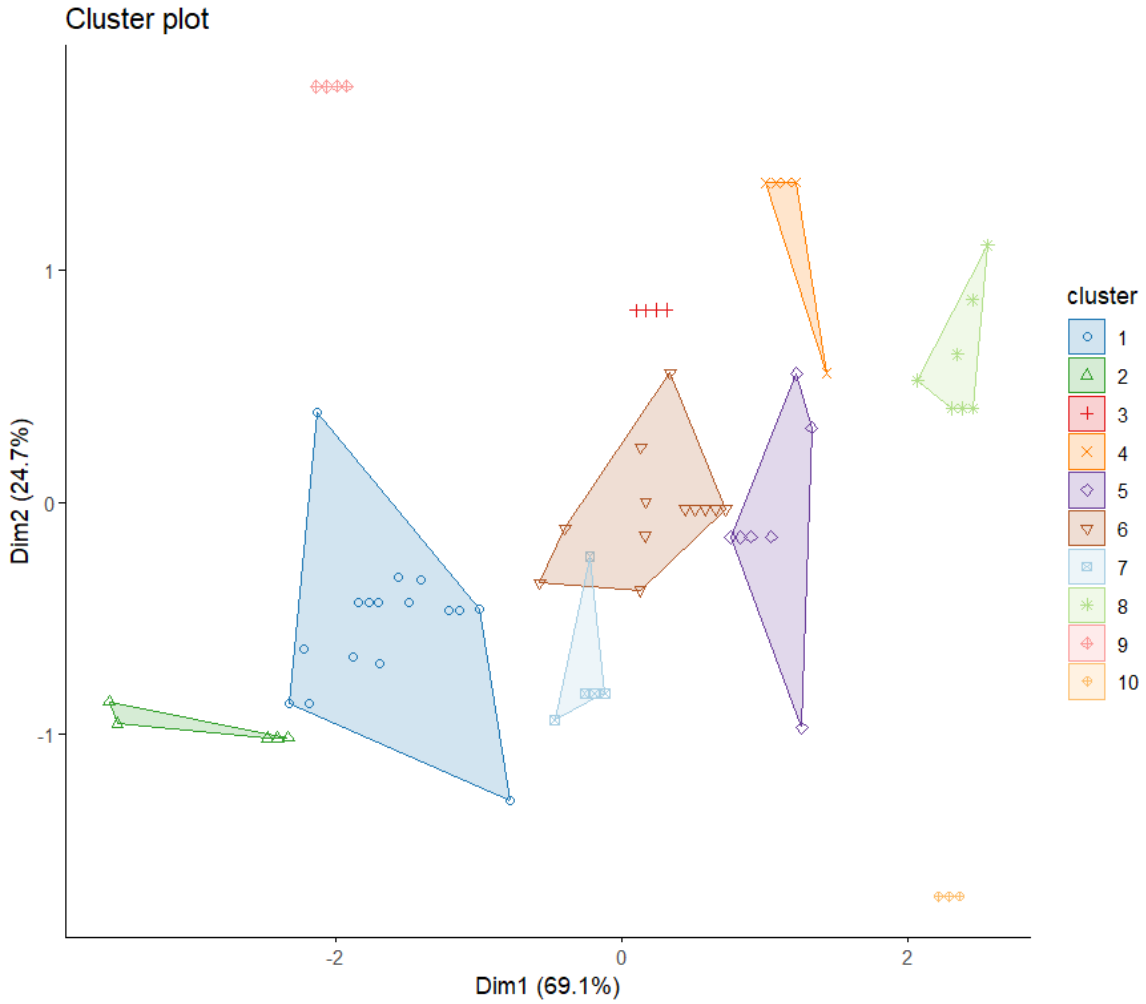
From the plot , the suggested number of cluster is k =10.
So now we compute the R code with the PAM algorithm with K =10.

```
> #Adding the point classifications to the original data
> dd <- cbind(Sleep_data, cluster =pam.res$cluster)
> head(dd, n = 8)
> head(dd, n = 8)
            Occupation Sleep_Duration Stress_Level Physical_Activity_Level Quality_of_Sleep c
1    Software Engineer            6.1            6                      42                6
2               Doctor            6.2            8                      60                6
3               Doctor            6.2            8                      60                6
4 Sales Representative            5.9            8                      30                4
5 Sales Representative            5.9            8                      30                4
6    Software Engineer            5.9            8                      30                4
7              Teacher            6.3            7                      40                6
8               Doctor            7.8            6                      75                7
```

```
> ##Visualizing PAM cluster
>
> #on the original space
>
> cl <- pam.res$clustering
> pairs(df, gap = 0, pch =cl, col=c("#1f78b4", "#33a02c", "#e31a1c", "#ff7f00", "#6a3d9a", "#b
>
```

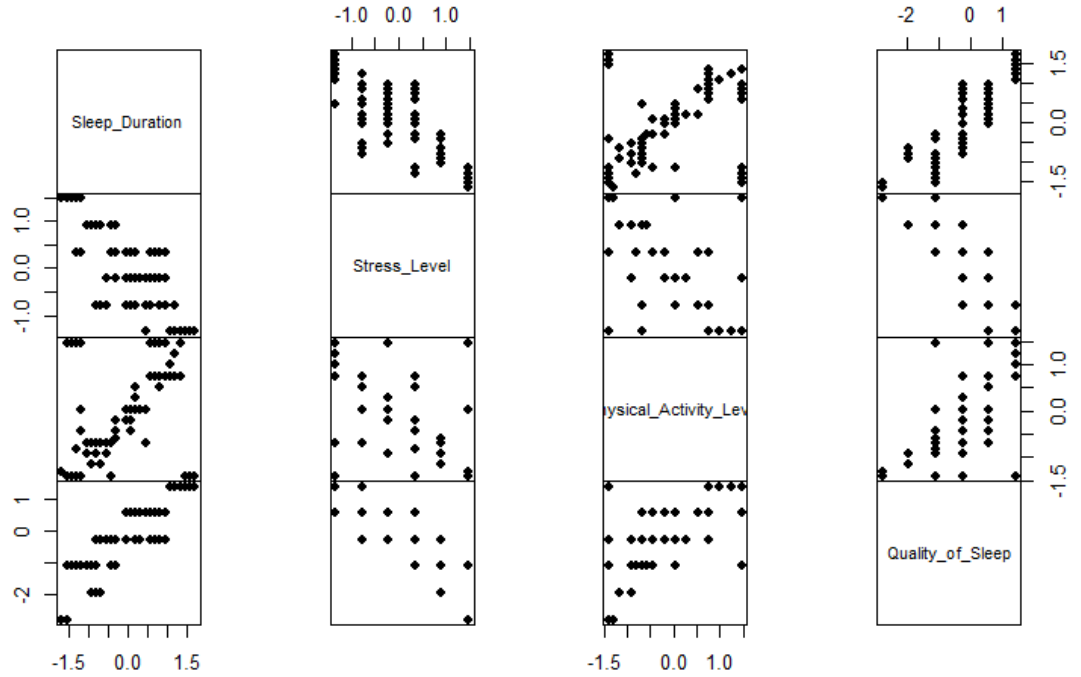


```
> # Visualize PAM clusters on the space of the first 10 PCs
> fviz_cluster(pam.res,
+              geom = "point",
+              palette = c("#1f78b4", "#33a02c", "#e31a1c", "#ff7f00", "#6a3d9a
", "#b15928", "#a6cee3", "#b2df8a", "#fb9a99", "#fdbf6f"),
+              repel = TRUE,  # Avoid label overplotting (slow)
+              ggtheme = theme_classic()
+ )
```



# CLUSTER VALIDATION

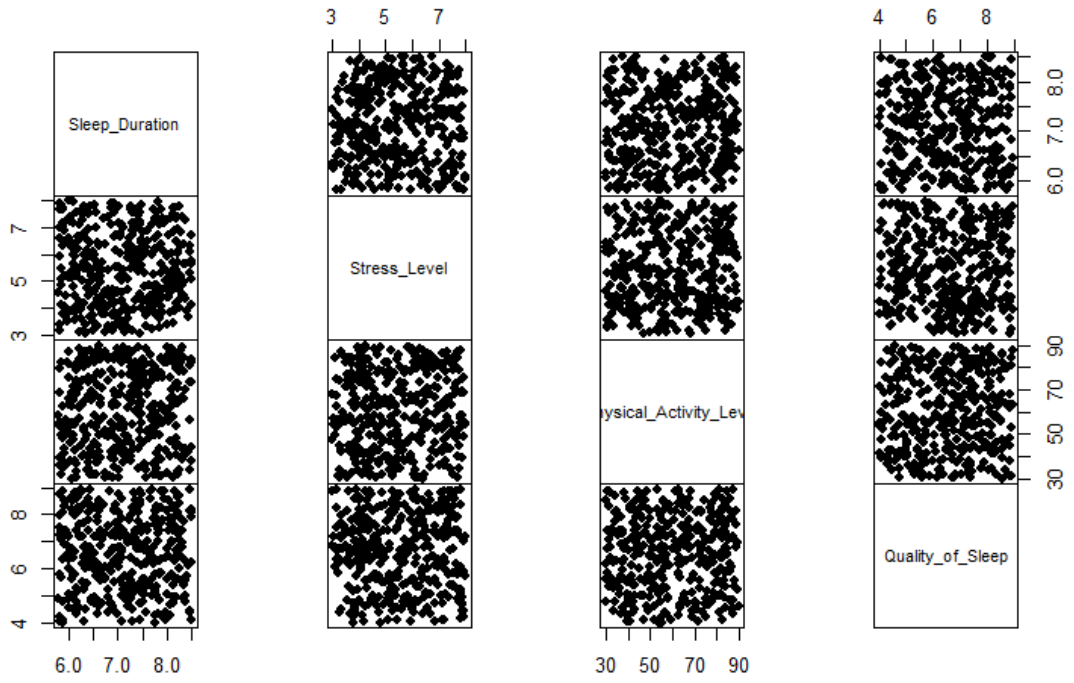
```
> head(Sleep_data,3)
         Occupation Sleep_Duration Stress_Level Physical_Activity_Level Quality
_of_Sleep
1 Software Engineer            6.1            6                       42
6
2            Doctor            6.2            8                       60
6
3            Doctor            6.2            8                       60
6
> df <- Sleep_data[, -1] #esclude the nominal variale Occuptio
>  # Random data generated from the iris data set
>  random_df <- apply(df, 2,function(x){runif(length(x), min(x), max(x))})
> random_df <- as.data.frame(random_df)
>
> #standardize the data
>
> df <- scale(df)
> random_df <- scale(random_df)
>
>
> #standardize data
>
> pairs(df , gap = 0, pch=16)
```




To visualizaing the data acces whether they contain meaningful cluster.

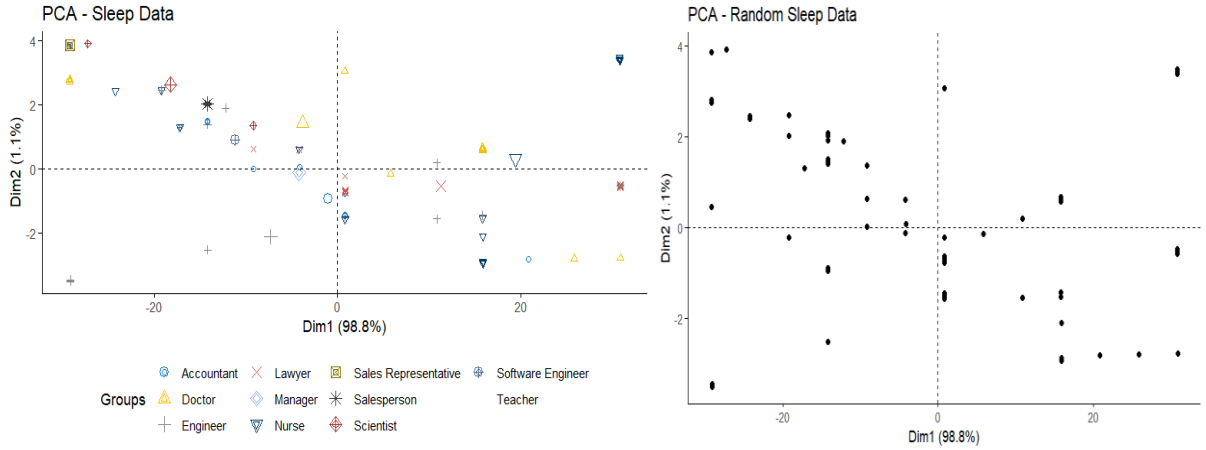
Instead, it can be seen that the standardized randomly generated uniform data do not contain meaninful clusters.

```
> #Standardize uniform random df data
>
> pairs(random_df, gap=0, pch=16)
```



As we can reduce the dimensionality, in order to make the visualization easier, usi ng the PCA (with the R function prcomp()). After performing PCA, we use the func tion fviz pca ind() (in the factoextra package) to visualize the output.

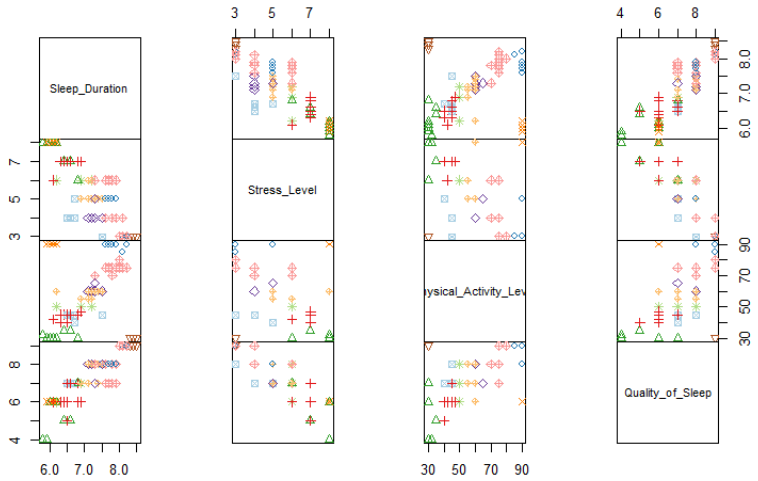
```
> library("factoextra")
>
> #plot the standardize df data
>
> fviz_pca_ind(prcomp(df), title ="PCA - Sleep Data",
+ habillage = Sleep_data$Occupation, palette ="jco",
+ geom = "point",ggtheme = theme_classic(),
+ legend = "bottom")

> #plot the random df data
>
> fviz_pca_ind(prcomp(df), title ="PCA - Random Sleep Data",
+ geom ="point", ggtheme =theme_classic())
```



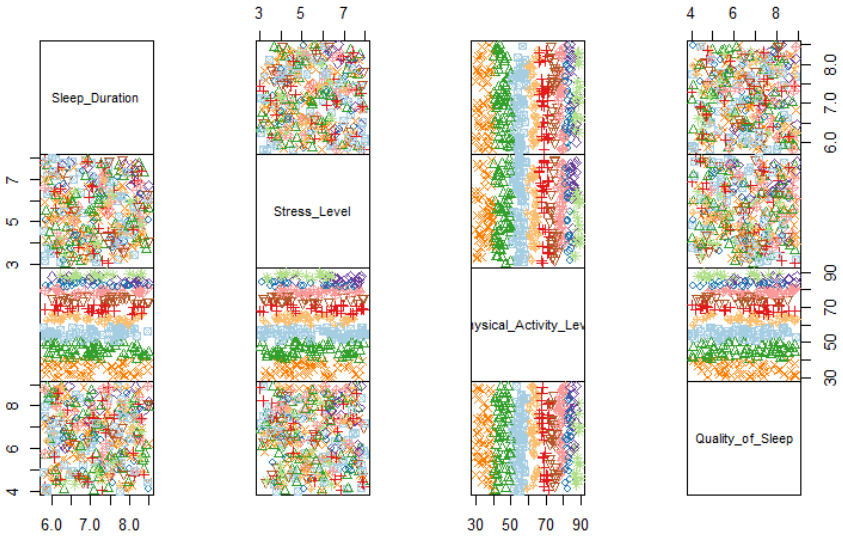
As we can see that the standardize Sleep data contain no meaninful clusters.

To illustrate why it's important to assess cluster tendency, we start by computing K-means and hierarchical clustering on the two data sets (df and random df).

```
> library(factoextra)
> set.seed(123)
>
> #Kmeans on DF data
>
> km.res1 <- kmeans(df, 10)
>
> #plot the original space
>
> cl1 <- km.res1$cluster
> pairs(df, gap = 0, pch=cl1, col=c("#1f78b4", "#33a02c", "#e31a1c", "#ff7f00",
"#6a3d9a", "#b15928", "#a6cee3", "#b2df8a", "#fb9a99", "#fdbf6f")[cl1])
```

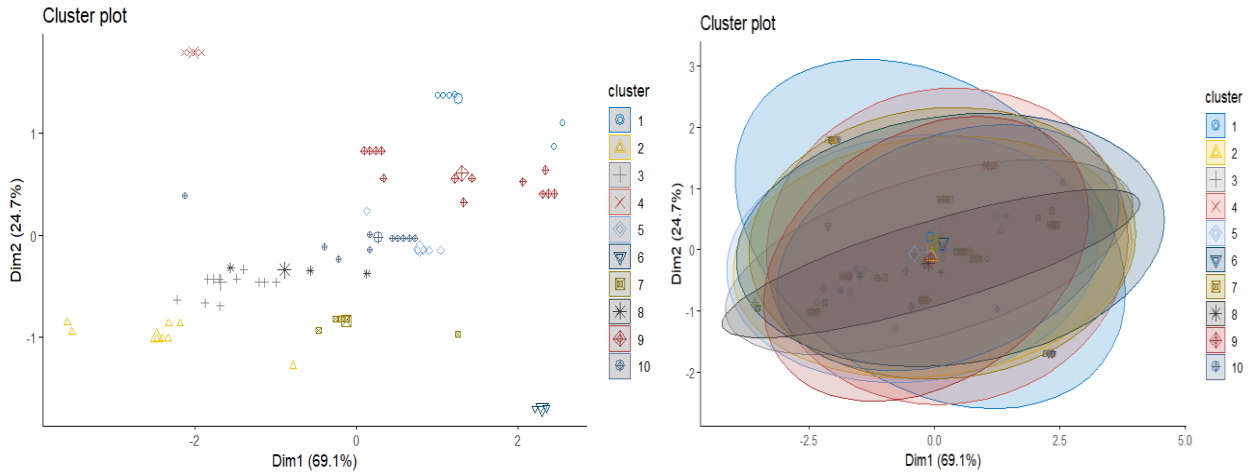


```
> #K-means on random df data
>
> km.res2 <- kmeans(random_df, 10)
>
> #Plot in the original space
> cl2<- km.res2$cluster
> pairs(random_df, gap=0, pch=cl2, col=c("#1f78b4", "#33a02c", "#e31a1c", "#ff7f00", "#6a
5928", "#a6cee3", "#b2df8a", "#fb9a99", "#fdbf6f")[cl2])


>
```




As we can see that K-means imposes a clustering structure on the random_df data even if the are no meaningful clusters.So according to this is why clustering tende ncy assessment methods should be used to evaluate the validity of clustering anal ysis, that is whether a given data set contains meaningful clusters.

```
> # Plot of K-means result on the df data in the PC space
> fviz_cluster(list(data = df, cluster = km.res1$cluster),
+              ellipse.type = "t",  # or "euclid"
+              geom = "point",
+              stand = FALSE,
+              palette = "jco",
+              ggtheme = theme_classic()
+ )
> #Plot of K-Means reults on the random df data in the PC space
>
> # Plot of K-means result on the df data in the PC space
> fviz_cluster(list(data = df, cluster = km.res2$cluster),
+              ellipse.type = "t",  # or "euclid"
+              geom = "point",
+              stand = FALSE,
+              palette = "jco",
+              ggtheme = theme_classic()
+ )
>
```



- The argument ellipse.type =”norm” specifies the type of the ellipse for each cluste r to be the one from the use of a multivariate normal distribution.

- Becauese of the clear overlap between groups, this plot confirms the need to asses s clustering tendency.

## Methods for assessing clustering tendency

We describe two methods for evaluating the clustering tendency:

- a statistical method is the Hopkins statistic;
- a visual method: Visual Assessment of cluster Tendency (VAT) algorithm.

**Hopkins Statistic:**

H close to 1 indicates clustered data; H around 0 indicates no cluster and so we c an say that there aren't no cluster , because the value of hopkins is 0.1170.
The R function hopkins() (in the clustertend package):

```
> ## Hopkins statistic
>
> library(clustertend)
>
> # compute Hopkins statistic for the Sleep data set
>
> set.seed(123)
>
> hopkins(df, n =nrow(df)-1)
$H
[1] 0.1170034
>
>
> set.seed(123)
> hopkins(random_df, n = nrow(random_df)-1)
$H
[1] 0.5011987
```

It can be seen that the Sleep data set (df) is clusterable because the H value (0.117 0034) is close enough to 0. However, the random df data set is not clusterable (H = 0.5011).

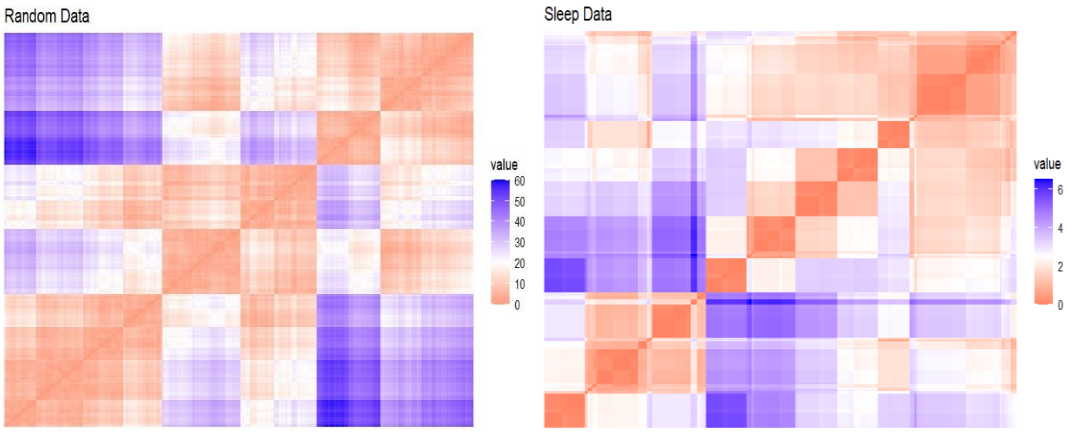
## VAT Algorithm

The VAT algorithm for visually evaluating cluster tendency consists of the following steps:

1. Calculate the dissimilarity matrix (DM) among the data set's units using the Euclidean distance.
2. Rearrange the DM to bring similar units into proximity, resulting in an ordered dissimilarity matrix (ODM).
3. Visualize the ODM as an ordered dissimilarity image (ODI), serving as the visual output of the VAT algorithm.

To visually assess clustering tendency, initiate the process by computing the dissimilarity matrix between observations, utilizing the **dist()** function. Subsequently, employ the **fviz_dist()** function from the factoextra package to present the dissimilarity matrix visually.

```
> ## VAT algorithm
>
> fviz_dist(dist(df), show_labels =FALSE)+
+ labs(title = "Sleep Data")
(dist(random_df), show_labels =FALSE)+
+      labs(title = "Random Data")

> >  fviz_dist(dist(random_df), show_labels =FALSE)+
+      labs(title = "Random Data")
```




- red denotes high similarity (i.e. low dissimilarity);
- blue denotes low similarity (i.e. high dissimilarity).

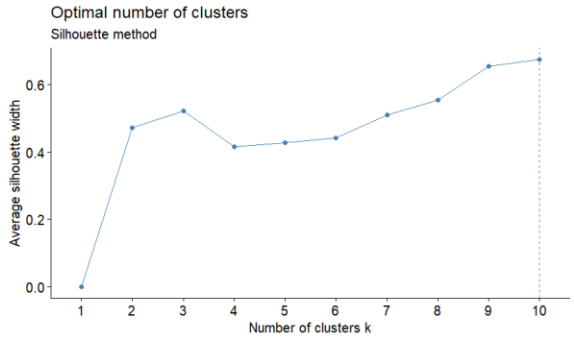
## Determining the Optimal Number of Clusters

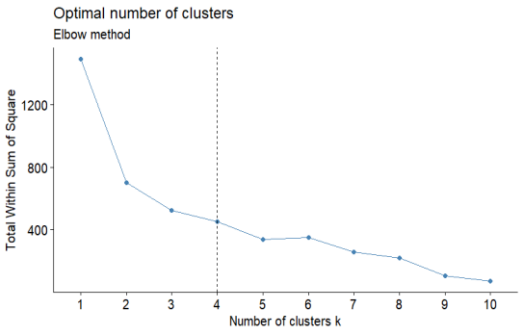
Determining the optimal number of clusters, denoted as K, is a subjective and method-dependent challenge with no definitive solution. Two main approaches are commonly employed:

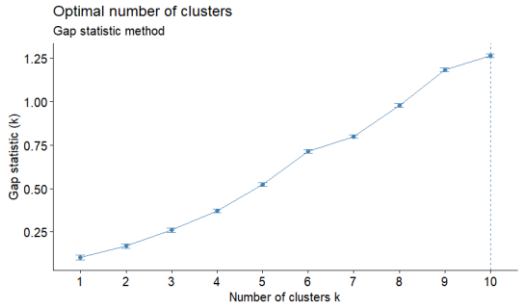
1. **Direct Methods:** These involve optimizing a specific criterion, such as the within-cluster sums of squares (WSS) or the average silhouette. The elbow method is used for WSS, while the silhouette method is applied for the average silhouette.
2. **Statistical Testing Methods:** These methods compare evidence against a null hypothesis of no inherent clustering structure. An example is the gap statistic.

It's noteworthy that, besides the elbow, silhouette, and gap statistic methods, there exist over thirty other indices and techniques published for determining the optimal number of clusters.

```
> #Elbow Method
>
> fviz_nbclust(df, kmeans, method ="wss")+
+ geom_vline(xintercept = 4, linetype = 2)+
+ labs(subtitle = "Elbow method")
>
> # Silhouette Method
>
> fviz_nbclust(df, kmeans, method ="silhouette")+
+ labs(subtitle = "Silhouette method")
>
>
> # Gap statistic
> # nboot = 50 to keep the functiob speedy.
> #recommented value: nboot = 500 for your analysis
> # use verbose = FALSE  to hide computing progression
>
> set.seed(123)
> fviz_nbclust(df, kmeans, nstart = 25, method = "gap_stat", nboot = 50)+
+ labs(subtitle = "Gap statistic method")
Clustering k = 1,2,..., K.max (= 10): .. done
Bootstrapping, b = 1,2,..., B (= 50)  [one "." per sample]:
.................................................. 50
```



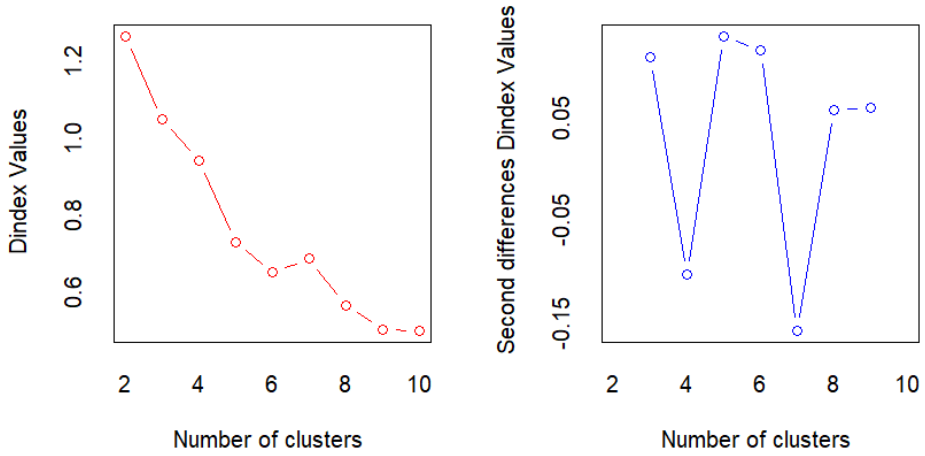

The suggested solutions are :
Elbow method: 4 clusters;
Silhouette method: 10 clusters;
Gap statistic Method: 10 cluters;

According to these results , it's possible to define K = 10 as the optimal number of cluster in the d ata because it has been chosen the most times

```
> library(NbClust)
>
> nb <- NbClust(df, distance = "euclidean", min.nc = 2, max.nc = 10, method = "
kmeans")
*** : The Hubert index is a graphical method of determining the number of clust
ers.
                In the plot of Hubert index, we seek a significant knee that co
rresponds to a
                significant increase of the value of the measure i.e the signif
icant 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 signifi
cant peak in Dindex
                second differences plot) that corresponds to a significant incr
ease of the value of
                the measure.

*******************************************************************
* Among all indices:
* 6 proposed 2 as the best number of clusters
* 3 proposed 3 as the best number of clusters
* 4 proposed 5 as the best number of clusters
* 2 proposed 6 as the best number of clusters
* 2 proposed 8 as the best number of clusters
* 5 proposed 9 as the best number of clusters
```

```
* 1 proposed 10 as the best number of clusters

                    ***** Conclusion *****

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

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

```
> library("factoextra")
> fviz_nbclust(nb)
```

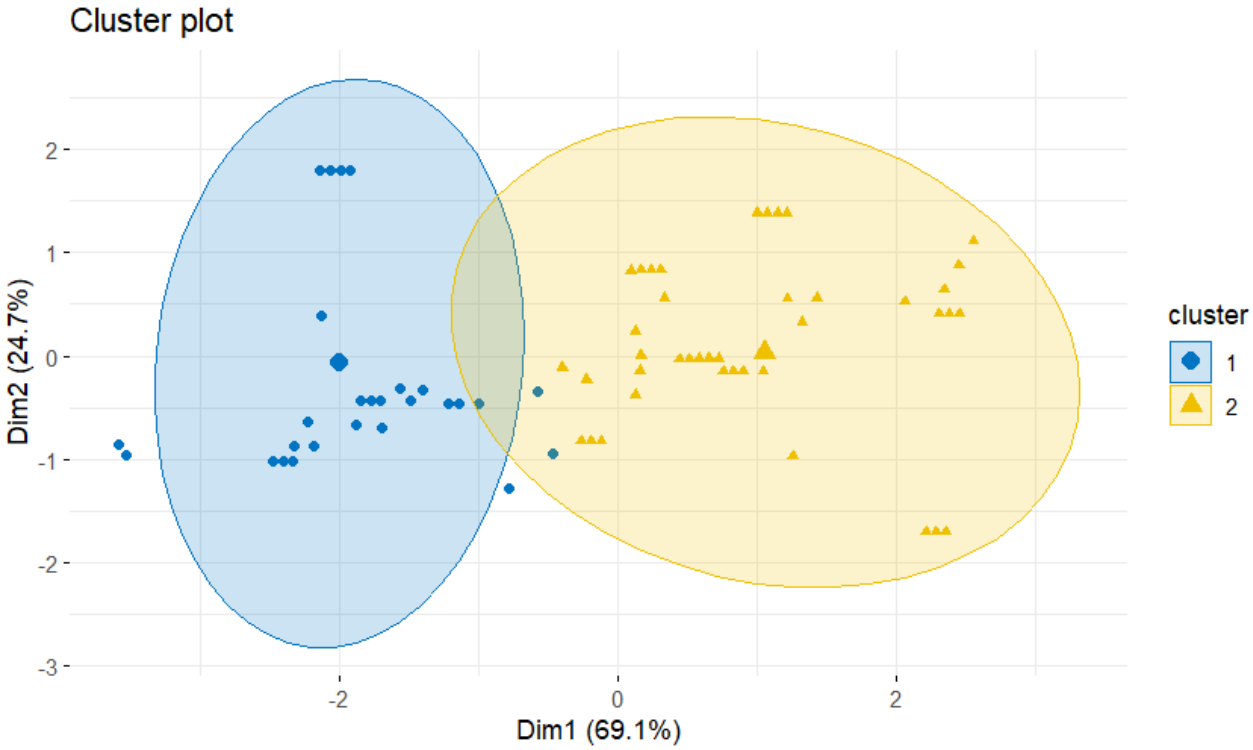


```
>
> library(flexclust)
> clus.res.kme <- kcca(df, 10,family = kccaFamily(which="kmeans", dist="euclide
an"))
> table(Sleep_data$Occupation,attr(clus.res.kme, "cluster"))

                      1  2  3  4  5  6  7  8  9 10
  Accountant          2  0  0  0 29  0  6  0  0  0
  Doctor              4 32  0  0  2  2  0  0  0 31
  Engineer           17  0  1 32 11  1  1  0  0  0
  Lawyer             17  0  0  0 27  0  3  0  0  0
  Manager             0  0  0  0  0  0  0  1  0  0
  Nurse               3  0  4 31  1  1  1  0 32  0
  Sales Representative  0  0  2  0  0  0  0  0  0  0
  Salesperson         0  0  0  0  0 32  0  0  0  0
  Scientist           0  0  2  0  0  2  0  0  0  0
  Software Engineer   0  0  1  0  2  1  0  0  0  0
  Teacher             0  0  2  0  6  3  2 27  0  0
```

Based on our earlier observations, it appears that there isn't a s
trong overall tendency for the data set to form distinct clusters.
However, upon closer analysis, we noticed a notable pattern: indiv
iduals categorized as 'Salesperson' consistently belong to cluster
6. To address this observation, we can make adjustments to the clu
stering assignments.

```
> km.res <- eclust(df, "kmeans", k = 10, nstart = 25, graph = FALSE)
>
> fviz_cluster(km.res, geom ="point", ellipse.type ="norm",
+ palette = "jco", ggtheme =theme_minimal())
```

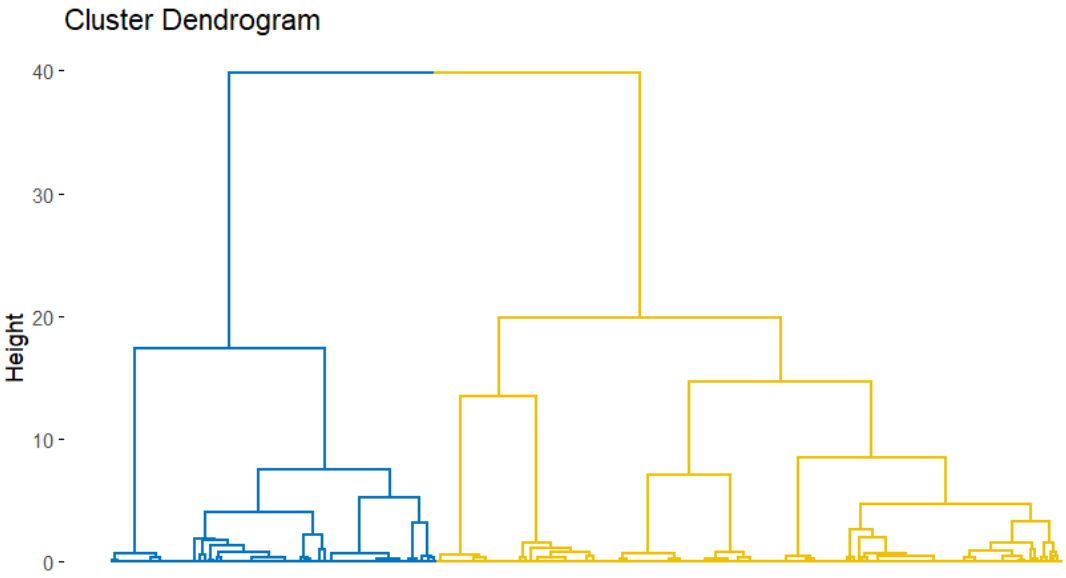
To compute a K-means clustering with k = 10




, we can write

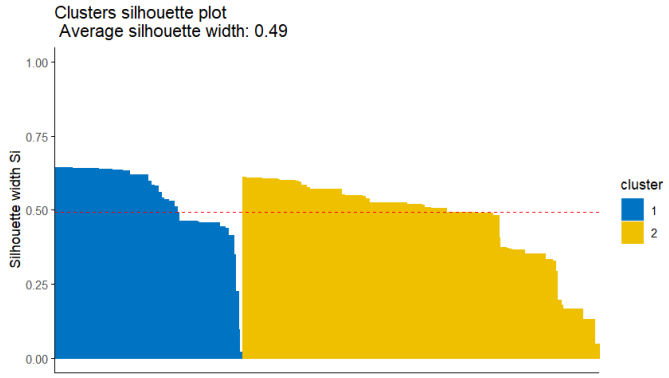
To compute a hierarchical clustering, and to cut it so to have K = 3 clusters, we ca n write

```
> hc.res <- eclust(df, "hclust", k = 2, hc_metric ="euclidian",
+ hc_method ="ward.D2", graph = FALSE)
>
> #Visualize the dendogram
>
> fviz_dend(hc.res, show_labels =FALSE,
+ palette = "jco", as.ggplot =TRUE)
```



# SILHOUETTE PLOT

```
> # sILHOUETTE plot
>
> fviz_silhouette(km.res, palette = "jco",
+ ggtheme = theme_classic())
  cluster size ave.sil.width
1       1  129          0.55
2       2  245          0.47
```




```
> #Silhouette information
>
> silinfo <- km.res$silinfo
> names(silinfo)
[1] "widths"          "clus.avg.widths" "avg.width"
>
> head(silinfo$width[, 1:2], 10)
    cluster neighbor
188       1        2
223       1        2
226       1        2
228       1        2
230       1        2
232       1        2
234       1        2
236       1        2
239       1        2
242       1        2
>
>
> #Average silhouette width of each cluster
>
> silinfo$clus.avg.widths
[1] 0.5455077 0.4669517
>
>
> #The total average(mean of all individual silhoette width)
```

```
> silinfo$avg.width
[1] 0.4940472
>
> #The size of each clusters
>
> km.res$size
[1] 129 245


>
```

## Computing Dunn index and other cluster validation statistics

The function cluster.stats() (in the fpc package) and the function NbClust() (in the NbClust package) can be used to compute the Dunn index and many other indices.
Using **cluster.stats**:

- **cluster.number**: Number of clusters.
- **cluster.size**: Vector containing the number of points in each cluster.
- **average.distance**, **median.distance**: Vector containing the cluster-wise within average/median distances.
- **average.between**: Average distance between clusters (want it to be as large as possible).
- **average.within**: Average distance within clusters (want it to be as small as possible).
- **clus.avg.silwidths**: Vector of cluster average silhouette widths.

```
> # Assuming df is your data and km.res$cluster contains cluster assignments
> km_stats <- cluster.stats(dist(df), km.res$cluster)
>
> # Dunn index
> km_stats$dunn
[1] 0.09893502
```

## External Cluster Validation

Does the K-means clustering matches with the true structure of the data?
We can use the function cluster.stats() to answer

```
> # Confusion matrix
>
> table(Sleep_data$Occupation, km.res$cluster)

                       1  2
  Accountant           6 31
  Doctor              33 38
  Engineer             3 60
  Lawyer               1 46
  Manager              0  1
  Nurse               37 36
  Sales Representative  2  0
  Salesperson          32  0
  Scientist            4  0
  Software Engineer    2  2
  Teacher              9 31
```

This confusion matrix provides a useful overview of the performance of the cluste ring algorithm in capturing the underlying patterns in the data, especially in comp arison to the known "Occupation" labels.

```
> library(clValid)
>
> df <- scale(Sleep_data[, -1])
> rownames(df) <- rownames(Sleep_data) # add rownames
> clmethods <- c("hierarchical", "kmeans", "pam")
> intern <- clValid(df, nClust = 2:10, clMethods = clmethods, validation = "int
ernal")
>
> # Display summary
> summary(intern)

Clustering Methods:
 hierarchical kmeans pam

Cluster sizes:
 2 3 4 5 6 7 8 9 10

Validation Measures:
                                 2       3       4       5       6       7
8       9      10      11

hierarchical Connectivity   4.3841  4.3841  4.3841  4.8341  9.0623 13.1881 13.7
214 15.3881 15.3881 19.2460
             Dunn           0.1515  0.1515  0.1515  0.1866  0.1866  0.2772  0.2
923  0.2923  0.2923  0.3373
             Silhouette     0.4954  0.4877  0.5345  0.5509  0.5332  0.6265  0.6
627  0.6431  0.6936  0.6864
kmeans       Connectivity   6.5194  6.5194  8.8218  8.1163 12.3444 16.0369 22.3
270 28.3135 25.3556 29.2135
             Dunn           0.0950  0.0950  0.0989  0.0896  0.0914  0.2843  0.1
348  0.0880  0.0757  0.0919
             Silhouette     0.4961  0.4905  0.5381  0.5525  0.5344  0.6289  0.6
639  0.6572  0.7094  0.7035
pam          Connectivity   8.8218  8.8218 11.3940 11.7063 15.8496 19.9742 18.2
897 21.9000 18.2278 18.2278
             Dunn           0.0989  0.0989  0.1744  0.0416  0.0801  0.0833  0.0
896  0.1262  0.1262  0.1262
             Silhouette     0.4940  0.4898  0.5350  0.5456  0.6378  0.6747  0.7
182  0.7197  0.7250  0.7431

Optimal Scores:

             Score  Method       Clusters
Connectivity 4.3841 hierarchical 2
Dunn         0.3373 hierarchical 10
Silhouette   0.7431 pam          10
```

```
> library(clValid)
>
> # Ora prova a eseguire il tuo codice
> clmethods <- c("hierarchical", "kmeans", "pam")
> stab <- clValid(df, nClust = 2:10, clMethods = clmethods, validation = "stabi
lity")
> summary(stab)

Clustering Methods:
 hierarchical kmeans pam

Cluster sizes:
 2 3 4 5 6 7 8 9 10

Validation Measures:
                         2      3      4      5      6      7      8      9     1
0

hierarchical APN  0.0474 0.1516 0.0944 0.1444 0.2410 0.1357 0.1203 0.1648 0.116
0
             AD   1.7360 1.6265 1.2833 1.1592 1.0859 0.8587 0.7370 0.7259 0.602
3
             ADM  0.1652 0.5966 0.3349 0.5213 0.5906 0.4012 0.3500 0.3966 0.304
8
             FOM  0.7080 0.6927 0.6426 0.6020 0.5515 0.5256 0.4719 0.4623 0.433
4
kmeans       APN  0.0429 0.1487 0.0934 0.1512 0.2461 0.1303 0.1294 0.1581 0.115
0
             AD   1.7289 1.6194 1.2789 1.1509 1.0803 0.8451 0.7374 0.7048 0.587
1
             ADM  0.1473 0.5963 0.3352 0.5164 0.5932 0.3812 0.3628 0.3971 0.319
2
             FOM  0.7087 0.6942 0.6476 0.6036 0.5471 0.5212 0.4697 0.4603 0.436
4
pam          APN  0.0483 0.3020 0.3172 0.2208 0.1403 0.1595 0.1605 0.0758 0.082
1
             AD   1.7355 1.6997 1.4823 1.1627 0.8806 0.7888 0.6544 0.5142 0.478
4
             ADM  0.1589 0.7819 0.7801 0.5538 0.3631 0.3805 0.3580 0.2127 0.243
9
             FOM  0.7119 0.7110 0.6807 0.6178 0.5276 0.5174 0.4430 0.3982 0.393
7

Optimal Scores:

    Score  Method Clusters
APN 0.0429 kmeans 2
AD  0.4784 pam    10
ADM 0.1473 kmeans 2
FOM 0.3937 pam    10
```

The output from **clValid** provides information about the quality of clustering for different cluster numbers (2 to 10) using various clustering methods (hierarchical, kmeans, pam). Here's an explanation of some key terms:

**1. Connectivity:**

- This measures how well-connected the clusters are. Lower values are better, indicating tighter and more well-defined clusters.

**2. Dunn Index:**

- The Dunn index measures the compactness of clusters (within-cluster similarity) relative to the separation between clusters (between-cluster dissimilarity). Higher values are better.

**3. Silhouette Score:**

- The silhouette score measures how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The score ranges from -1 to 1, where a higher score indicates better-defined clusters.

**Optimal Scores:**

- This section provides the best score and the corresponding method and number of clusters for each validation measure.
- In this case, for Connectivity, the hierarchical method with 2 clusters is optimal.
- For Dunn Index, the hierarchical method with 10 clusters is optimal.
- For Silhouette Score, the pam method with 10 clusters is optimal.

**Explanation:**

- The optimal number of clusters can vary depending on the validation measure used. In this case, you might choose the number of clusters based on the specific criterion that is most important for your analysis.
- For example, if you prioritize tight and well-connected clusters, you might choose 2 clusters based on the Connectivity measure.
- If you prioritize a balance between within-cluster similarity and between-cluster dissimilarity, you might choose 10 clusters based on the Dunn Index.
- If you prioritize well-defined clusters with clear boundaries, you might choose 10 clusters based on the Silhouette Score.

# Model Based Clustering

**Model-Based Clustering - Solving Traditional Clustering Issues:** Traditional clustering methods have a problem – they're not very formal. Model-Based Clustering solves this by using statistical models, making our understanding more precise.

**Why Gaussian Mixtures?** Gaussian mixtures are like versatile recipes. They can describe various shapes of data, and the cool thing is, the model automatically figures out the recipe by looking at our data.

**Parsimonious Gaussian Mixtures - Keeping It Simple:** We want our recipe (model) to be simple but still explain our data well. So, we're talking about "parsimonious configurations." This means we're being clever about using fewer ingredients (parameters) but still making our recipe powerful.

**Eigen-decomposition to the Rescue:** Gaussian mixtures can be a bit complicated, so we simplify them using eigen-decomposition. It's like breaking down a complex task into smaller, manageable steps.

**Balancing Act: Fit vs. Complexity:** Imagine you're making a cake. You want it to taste great (fit the data well), but you don't want to use too many ingredients (keep it simple). We're looking for the sweet spot, where the cake tastes awesome without needing tons of ingredients. This balancing act is what we call the "right compromise."

**Penalizing Complexity - Like a Game Score:** Making your cake taste better might involve adding more ingredients, but there's a catch. We don't want to go overboard. It's like playing a game where your score is how well the cake tastes, but there's a penalty for using too many ingredients. We're trying to win the game with the highest score without breaking too many rules.

**Selecting the Best Configuration - Bayesian Information Criterion (BIC) to the Rescue:** Now, we need to pick the number of flavors (clusters) and the best, simplest recipe (parsimonious configuration). We use something like a judge's scorecard – the Bayesian Information Criterion (BIC). It helps us decide which combination of clusters and recipe simplicity is the winner.

So, in simple terms, we're using smart recipes (Gaussian mixtures) to explain our data in a simple way (parsimonious), playing a game where we balance making things taste great with not using too many ingredients. In the end, the judge (BIC) helps us choose the best combination of flavors (clusters) and recipe simplicity.

```
>  mod <- Mclust(df, G = 1:10, modelNames = NULL)
fitting ...
  |=======================================================================
=======================================================================
=============| 100%
> summary(mod$BIC)
Best BIC values:
           EEV,10    VII,10     VII,9
BIC      952.4111 894.06700  508.9922
BIC diff   0.0000 -58.34409 -443.4189
```

- The **mod** object is the result of fitting Gaussian mixture models to your data using the **Mclust** function.
- The **summary(mod$BIC)** command displays the BIC values for the models with different numbers of components (G).
- The "Best BIC values" section shows the BIC values for the top three models. In this case, the model with 10 components (EEV,10) has the lowest BIC value, indicating that it is the best-fitting model according to the BIC criterion.
- The "BIC diff" section shows the differences in BIC values compared to the best-fitting model. Negative values indicate that the corresponding models are less favored.

The term "VII" refers to a specific model configuration. In Mclust, models are named according to a combination of a letter representing the covariance type and a number indicating the shape of the clusters.

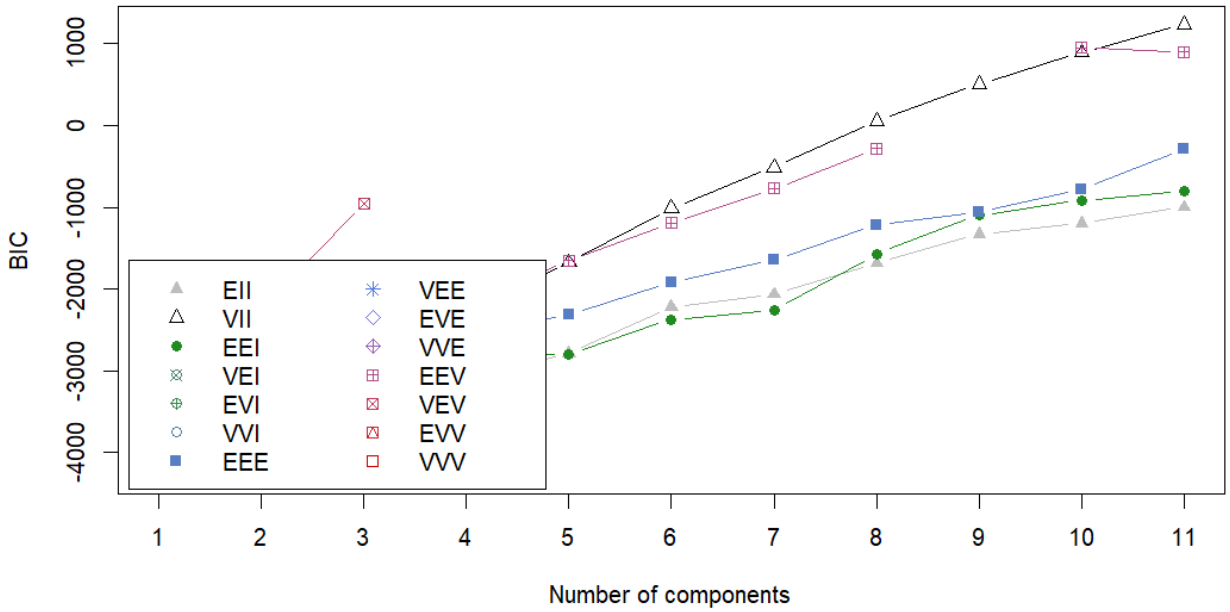
In the case of "VII":

- The letter "V" stands for "Variance Equal," indicating that the model assumes equal variance among the clusters.
- The "II" suggests a particular shape of the clusters; the exact interpretation of this may depend on the specific implementation details of the Mclust package.

So, the model denoted as "VII" assumes that the clusters have equal variance and have a specific shape as indicated by "II." Each model configuration (like "VII") represents a set of assumptions about the underlying distribution of the data, and the algorithm fits the data based on these assumptions.

In summary, "VII" is a shorthand notation representing a specific combination of covariance type and cluster shape assumptions in the Mclust model.

```
> plot(mod, what = "BIC", ylim = range(mod$BIC, na.rm = TRUE), legendArgs = lis
t(x = "b
+ ottomleft"))
```




In this plot, which is the graphical counterpart of the results of the previous code, we have the BIC curves for the penalized models; we have a different curve for eac h number of clusters and for each parsimonious configuration we have a different symbol. The maximum in this configuration is related to 10 clusters and VVI mod el.

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

Mclust VVI (ellipsoidal, equal volume and shape)
model with 10 components:

 log-likelihood   n  df      BIC      ICL
       810.926 374 113 952.4111 952.2416
```

```
Clustering table:
 1  2  3  4  5  6  7  8  9 10
18 42 42 31 72 39 37 29 32 32
```

- **Model Information:**
- **Mclust VII Model:** It's a specific type of model from the Mclust package, labeled as "VII," which stands for spherical, varying volume.
- **Number of Components:** The model has 10 components or clusters.
- **Fit Quality:**
- **Log-Likelihood:** The log-likelihood is a measure of how well the model explains the observed data. In this case, the log-likelihood is 810.926.
- **BIC (Bayesian Information Criterion):** This is a criterion for model selection. The lower the BIC, the better. Here, the BIC is 952.4111.
- **ICL (Integrated Completed Likelihood):** Another criterion for model selection. Like BIC, lower values are better. Here, the ICL is 952.2416.
- **Clustering Table:**
- This table shows the number of data points assigned to each of the 11 clusters. For example, cluster 1 has 60 data points, cluster 2 has 42, and so on.

In summary, the model is a Mclust VII model with 10 clusters, and it provides information on how well it fits the data (log-likelihood, BIC, and ICL), as well as the distribution of data points among the clusters.

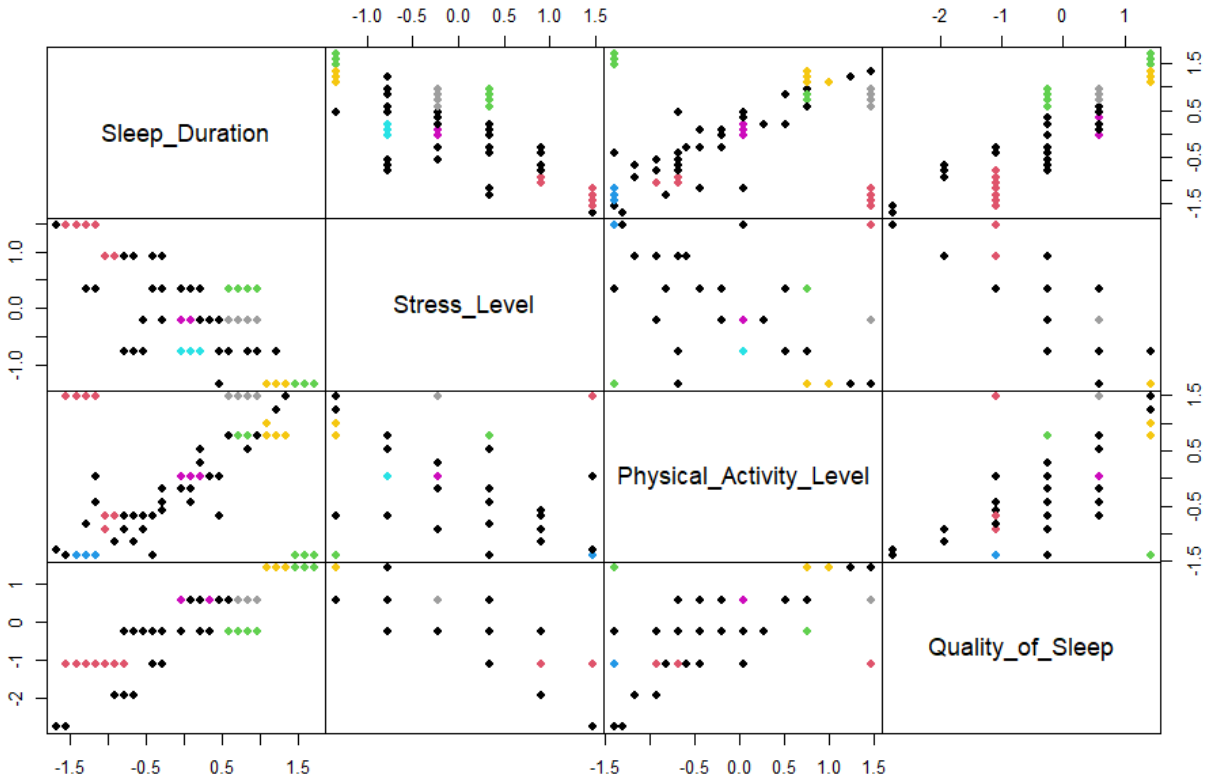
```
> head(round(mod$z, 6), 30)
> head(round(mod$z, 6), 30)
       [,1]     [,2]     [,3]     [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
1  1.000000 0.000000 0.000000 0.000000    0    0    0    0    0     0     0
2  1.000000 0.000000 0.000000 0.000000    0    0    0    0    0     0     0
3  1.000000 0.000000 0.000000 0.000000    0    0    0    0    0     0     0
4  1.000000 0.000000 0.000000 0.000000    0    0    0    0    0     0     0
5  1.000000 0.000000 0.000000 0.000000    0    0    0    0    0     0     0
6  1.000000 0.000000 0.000000 0.000000    0    0    0    0    0     0     0
7  0.242458 0.757542 0.000000 0.000000    0    0    0    0    0     0     0
8  0.000017 0.000000 0.999983 0.000000    0    0    0    0    0     0     0
9  0.000017 0.000000 0.999983 0.000000    0    0    0    0    0     0     0
10 0.000017 0.000000 0.999983 0.000000    0    0    0    0    0     0     0
11 0.000002 0.000000 0.000000 0.999998    0    0    0    0    0     0     0
12 0.000017 0.000000 0.999983 0.000000    0    0    0    0    0     0     0
13 0.000002 0.000000 0.000000 0.999998    0    0    0    0    0     0     0
14 0.000003 0.000000 0.000000 0.999997    0    0    0    0    0     0     0
15 0.000003 0.000000 0.000000 0.999997    0    0    0    0    0     0     0
16 0.000003 0.000000 0.000000 0.999997    0    0    0    0    0     0     0
17 1.000000 0.000000 0.000000 0.000000    0    0    0    0    0     0     0
18 0.000003 0.000000 0.000000 0.999997    0    0    0    0    0     0     0
19 1.000000 0.000000 0.000000 0.000000    0    0    0    0    0     0     0
20 0.000959 0.000000 0.999041 0.000000    0    0    0    0    0     0     0
21 0.000010 0.000000 0.999990 0.000000    0    0    0    0    0     0     0
22 0.000010 0.000000 0.999990 0.000000    0    0    0    0    0     0     0
23 0.000010 0.000000 0.999990 0.000000    0    0    0    0    0     0     0
```

```
24 0.000010 0.000000 0.999990 0.000000     0     0     0     0     0     0     0
25 0.000017 0.000000 0.999983 0.000000     0     0     0     0     0     0     0
26 0.003987 0.000000 0.996013 0.000000     0     0     0     0     0     0     0
27 0.000017 0.000000 0.999983 0.000000     0     0     0     0     0     0     0
28 0.003987 0.000000 0.996013 0.000000     0     0     0     0     0     0     0
29 0.003987 0.000000 0.996013 0.000000     0     0     0     0     0     0     0
30 0.003987 0.000000 0.996013 0.000000     0     0     0     0     0     0     0
```

This is z, the matrix of posterior probabilities, the one of soft assignment. We are looking at its first 30 rows. In particular, unit 1, has a probability equal to just ab out 1 to belonging to cluster 1, while, for the remaining part, it has a zero probab ility to belong to cluster 2 and a probability equal to zero to belong to cluster 3 an d this is the same to the cluster 10 ; hence unit 1 will be assigned to cluster 1 with the hard assignment. Same for unit 2, and so on.

Finally, we can graphically visualize the clustering results:

```
> pairs(df, gap=0, pch = 16, col = mod$classification)
```



Colours arise by the best model-based clustering model, VII with 10 clusters. So, w e can see that there is a clustering structure also by the result of Model-Based Clu stering.