MARCH 2023

Data-comp Project

Part 1

PRESENTED TO

Dr. Amr Amin

PRESENTED BY

Rahma Ezzat	20201381362
Asmaa Hamdy	20201032790
Karim Radwan	20201498197
Omar Mekkawy	20201375851

Table of Contents

Overview

This project aims to perform clustering analysis using the k-means algorithm + (hierarchal) and optimize it using the elbow and silhouette methods. Clustering is an unsupervised machine learning technique that is used to group similar data points based on their similarity in characteristics. K-means is a commonly used clustering algorithm that partitions the data points into k clusters by minimizing the within-cluster sum of squares.

The first step of the project is to preprocess the data and prepare it for clustering. This may involve data cleaning, feature selection, and scaling.
The k-means algorithm will then be applied to the preprocessed data to create k clusters. The optimal number of clusters will be determined using the elbow method, which involves plotting the within-cluster sum of squares as a function of the number of clusters and selecting the value of k at the elbow point.

The silhouette method is used in clustering analysis to evaluate the quality of a clustering solution by measuring how well each data point fits into its assigned cluster. It provides a quantitative measure of cluster quality and helps identify outliers or misclassified points within a cluster, making it a valuable tool for refining and improving clustering results.

Overall, this project aims to demonstrate the effectiveness of the k-means algorithm for clustering and the importance of optimization techniques such as the elbow and silhouette methods for improving clustering results.

Understanding the Data

CUSTOMER PERSONALITY ANALYSIS HELPS A BUSINESS TO BETTER UNDERSTAND ITS CUSTOMERS AND MAKES IT EASIER FOR THEM TO MODIFY PRODUCTS ACCORDING TO THE SPECIFIC NEEDS, BEHAVIOR AND CONCERN OF DIFFERENT TYPES OF CUSTOMERS.

content:

CUSTOMER	PRODUCTS	PROMOTION	PLACE
• **ID**: Customer's unique identifier • **Year_Birth**: Customer's birth year • **Education**: Customer's education level • **Marital_Status**: Customer's marital status • Income: Customer's yearly household income • **Kidhome**: Number of children in customer's household • **Teenhome**: Number of teenagers in customer's household • **Dt_Customer**: Date of customer's enrollment with the company • **Recency**: Number of days since customer's last purchase • **Complain**: 1 if customer complained in the last 2 years, 0 otherwise	• **MntWines**: Amount spent on wine in last 2 years • **MntFruits**: Amount spent on fruits in last 2 years • **MntMeatProducts**: Amount spent on meat in last 2 years • **MntFishProducts**: Amount spent on fish in last 2 years • **MntSweetProducts**: Amount spent on sweets in last 2 years • **MntGoldProds**: Amount spent on gold in last 2 years	• **MntWines**: Amount spent on wine in last 2 years • **MntFruits**: Amount spent on fruits in last 2 years • **MntMeatProducts**: Amount spent on meat in last 2 years • **MntFishProducts**: Amount spent on fish in last 2 years • **MntSweetProducts**: Amount spent on sweets in last 2 years • **MntGoldProds**: Amount spent on gold in last 2 years	• **NumWebPurchases**: Number of purchases made through the company's web site • **NumCatalogPurchases**: Number of purchases made using a catalogue • **NumStorePurchases**: Number of purchases made directly in stores • **NumWebVisitsMonth**: Number of visits to company's web site in the last month

Goal

- Need to preform clustering to summarize customer segments.

Getting Data Ready!

LET'S GET STARTED

1st we load the libraries that we need :

- library(corrgram)
- library(cluster)
- library(dplyr)
- library(ggplot2)
- library(factoextra)
- library(flexclust)

2nd we load the Data csv file:

```
#LOAD DATA
DATA <- READ.CSV("D:/MARKETING_CAMPAIGN.CSV" , SEP = "\T")

HEAD(DATA)
```

```
> head(data)
    ID Year_Birth  Education Marital_Status Income Kidhome Teenhome Dt_Customer Recency MntWines
1 5524       1957 Graduation         Single  58138       0        0  04-09-2012      58      635
2 2174       1954 Graduation         Single  46344       1        1  08-03-2014      38       11
3 4141       1965 Graduation       Together  71613       0        0  21-08-2013      26      426
4 6182       1984 Graduation       Together  26646       1        0  10-02-2014      26       11
5 5324       1981        PhD        Married  58293       1        0  19-01-2014      94      173
6 7446       1967     Master       Together  62513       0        1  09-09-2013      16      520
  MntFruits MntMeatProducts MntFishProducts MntSweetProducts MntGoldProds NumDealsPurchases
1        88              546             172                88          88                 3
2         1                6               2                 1           6                 2
3        49              127             111                21          42                 1
4         4               20              10                 3           5                 2
5        43              118              46                27          15                 5
6        42               98               0                42          14                 2
  NumWebPurchases NumCatalogPurchases NumStorePurchases NumWebVisitsMonth AcceptedCmp3 AcceptedCmp4
1               8                  10                 4                 7            0            0
2               1                   1                 2                 5            0            0
3               8                   2                10                 4            0            0
4               2                   0                 4                 6            0            0
5               5                   3                 6                 5            0            0
6               6                   4                10                 6            0            0
  AcceptedCmp5 AcceptedCmp1 AcceptedCmp2 Complain Z_CostContact Z_Revenue Response
1            0            0            0        0            3        11        1
2            0            0            0        0            3        11        0
3            0            0            0        0            3        11        0
4            0            0            0        0            3        11        0
5            0            0            0        0            3        11        0
6            0            0            0        0            3        11        0
```

3rd we Display structure of data

4th we Display Columns names

Data contains 29 variables and 2240 observations about different customers

STR(DATA)

```
data.frame :   2240 obs. of
$ ID                 : int
$ Year_Birth         : int
$ Education          : chr
$ Marital_Status     : chr
$ Income             : int
$ Kidhome            : int
$ Teenhome           : int
$ Dt_Customer        : chr
$ Recency            : int
$ MntWines           : int
$ MntFruits          : int
$ MntMeatProducts    : int
$ MntFishProducts    : int
$ MntSweetProducts   : int
$ MntGoldProds       : int
$ NumDealsPurchases  : int
```

NAMES(DATA)

```
 [1] "ID"
 [3] "Education"
 [5] "Income"
 [7] "Teenhome"
 [9] "Recency"
[11] "MntFruits"
[13] "MntFishProducts"
[15] "MntGoldProds"
[17] "NumWebPurchases"
[19] "NumStorePurchases"
[21] "AcceptedCmp3"
[23] "AcceptedCmp5"
[25] "AcceptedCmp2"
[27] "Z_CostContact"
```

Cleaning!!

```r
#Data Cleaning
sum(is.na(data))
#handle missing values
data <- na.omit(data)
sum(is.na(data))

# Remove duplicate rows (all columns)
#data <- data %>% distinct()
# Remove duplicate rows
data <- data[!duplicated(data), ]

#Derive Age from Year_Birth
data$Age <- 2023 - data$Year_Birth
#extract the year as string from Dt_Customer then convert it to
print(min(data$Dt_Customer))
print(max(data$Dt_Customer))
data$Customer_Year <- as.character(data$Dt_Customer)
data$Customer_Year <- substr(data$Customer_Year , start = 7 , st
data$Customer_Year <- as.numeric(data$Customer_Year)
```

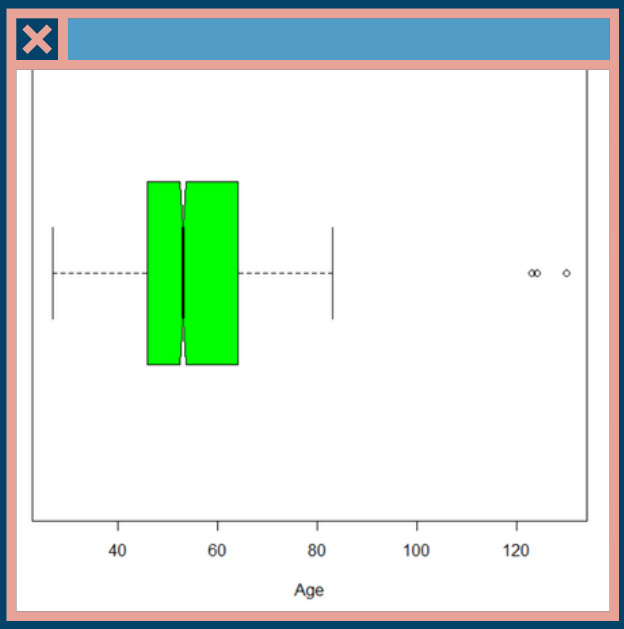
-ToDoz-

- Handle missing values
- Handle duplicates
- Derive age from Year_Birth
- change Dt_Customer to Year_Customer

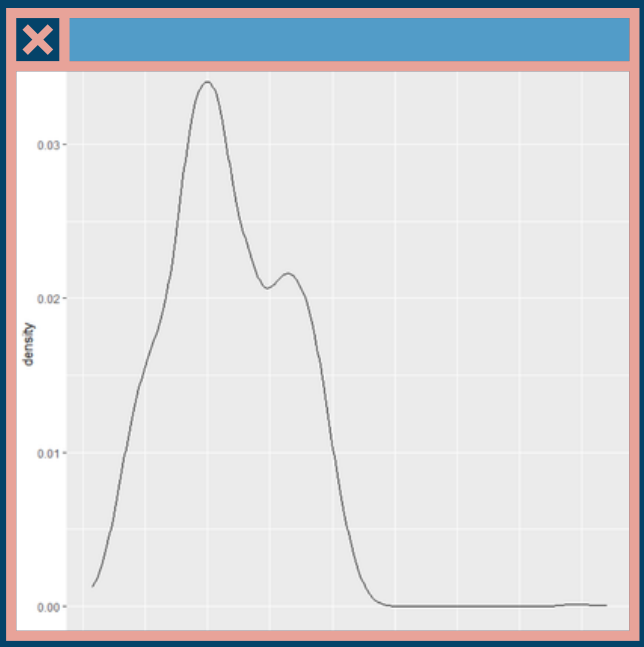
which means how many years has the customer been a customer, then calculate the seniority of the customer from Year_Customer

some Insights!!

```
BOXPLOT(DATA$AGE, MAIN = "CUSTOMER AGE", XLAB = "AGE",
COL = "GREEN", HORIZONTAL = T, NOTCH = T )
```




```
GGPLOT(DATA , AES(AGE)) + GEOM_DENSITY()
```



Most of customers age is between 48 : 58

DATA. PREB

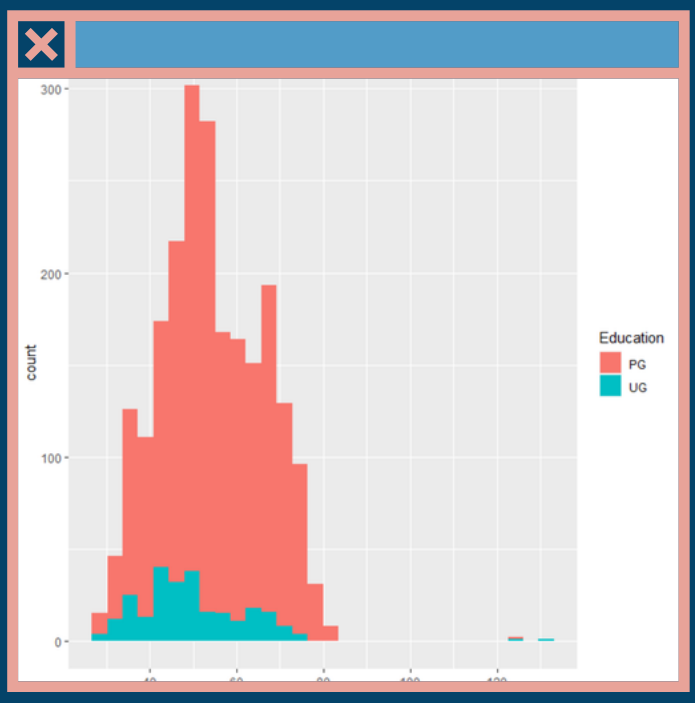
- convert Material_Status to Single & Couple categories
- convert Education to undergraduate & postgraduate categories

```
#CONVERT EDUCATION TO UG & PG CATEGORIES
UNIQUE(DATA$EDUCATION)
DATA$EDUCATION[DATA$EDUCATION  == "2N CYCLE"] = "UG"
DATA$EDUCATION[DATA$EDUCATION  == "BASIC"] = "UG"
DATA$EDUCATION[DATA$EDUCATION  == "GRADUATION"] = "PG"
DATA$EDUCATION[DATA$EDUCATION  == "MASTER"] = "PG"
DATA$EDUCATION[DATA$EDUCATION  == "PHD"] = "PG"
```

```
#CONVERT MATERIAL_STATUS TO SINGLE & COUPLE CATEGORIES
UNIQUE(DATA$MARITAL_STATUS)
DATA$MARITAL_STATUS[DATA$MARITAL_STATUS == "DIVORCED"] =
"SINGLE"
DATA$MARITAL_STATUS[DATA$MARITAL_STATUS == "ABSURD"] =
"SINGLE"
DATA$MARITAL_STATUS[DATA$MARITAL_STATUS == "YOLO"] =
"SINGLE"
DATA$MARITAL_STATUS[DATA$MARITAL_STATUS == "WIDOW"] =
"SINGLE"
DATA$MARITAL_STATUS[DATA$MARITAL_STATUS == "TOGETHER"] =
"COUPLE"
DATA$MARITAL_STATUS[DATA$MARITAL_STATUS == "MARRIED"] =
"COUPLE"
DATA$MARITAL_STATUS[DATA$MARITAL_STATUS == "ALONE"] =
"SINGLE"
```

```
GGPLOT(DATA = DATA, AES(AGE, FILL = EDUCATION)) + GEOM_HISTOGRAM()
```

Most of customers are postgraduate



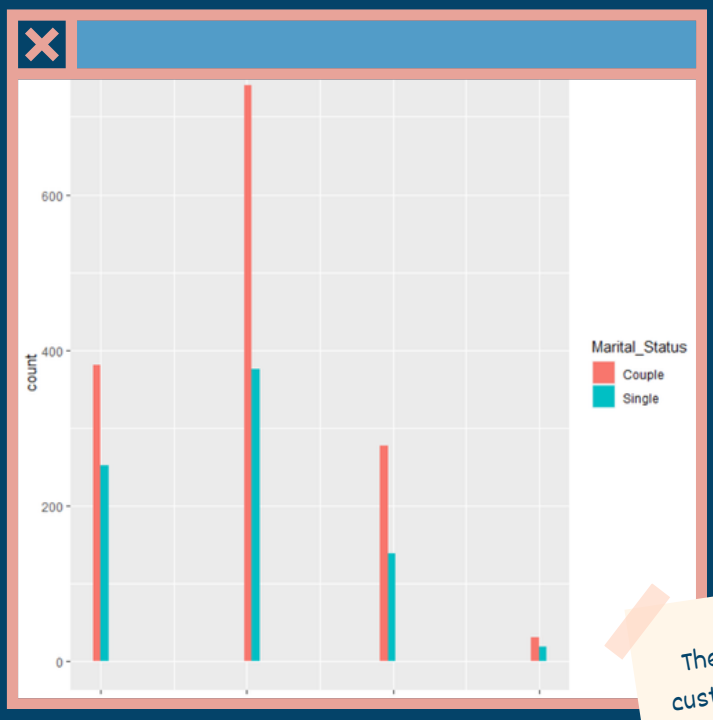
DATA. PREB

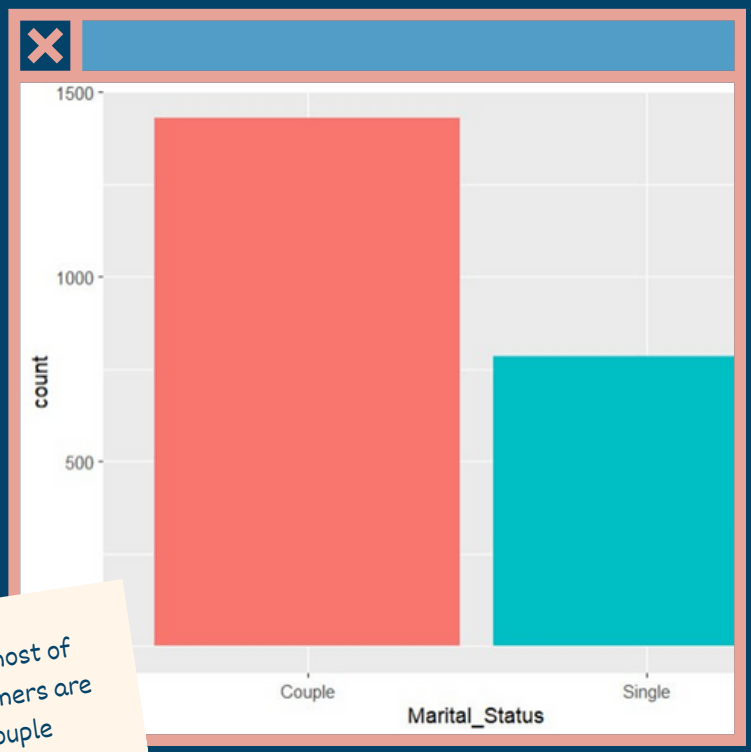
- Calculate the total number of customer's children (kids , teens)

```
#CALCULATE THE TOTAL NUMBER OF CHILDREN
DATA$CHILD <- DATA$KIDHOME + DATA$TEENHOME
```

```
GGPLOT(DATA = DATA,  AES(CHILD, FILL = MARITAL_STATUS)) +
GEOM_HISTOGRAM(POSITION = "DODGE" )
```

```
GGPLOT(DATA = DATA,  AES(CHILD, FILL = MARITAL_STATUS)) +
GEOM_HISTOGRAM(POSITION = "DODGE" )
```





The most of customers are couple

DATA. PREB

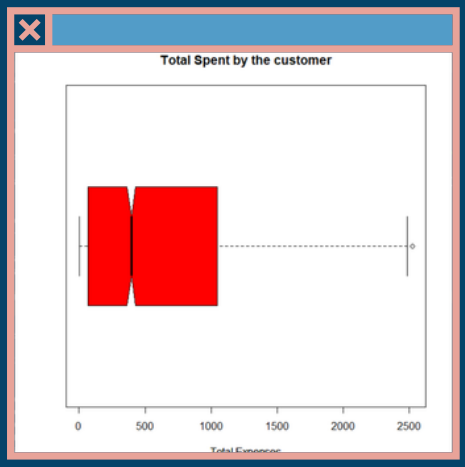
- sum up the total expenses and total accepted campaign for each customers

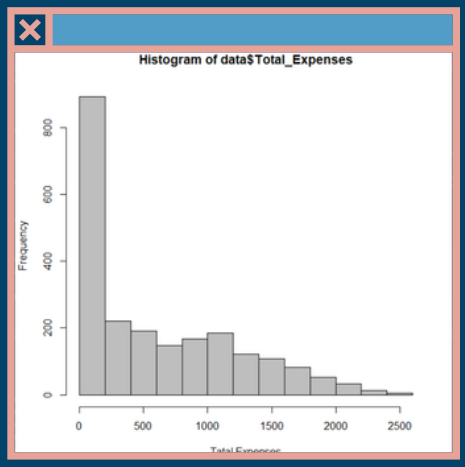
```
#SUM UP THE TOTAL EXPENSES AND TOTAL ACCEPTED CAMPAIGN FOR EACH CUSTOMERS
DATA$TOTAL_EXPENSES <- DATA$MNTWINES + DATA$MNTFRUITS + DATA$MNTMEATPRODUCTS + DATA$MNTFISHPRODUCTS + DATA$MNTSWEETPRODUCTS + DATA$MNTGOLDPRODS

DATA$TOTAL_ACC_CMP <- DATA$ACCEPTEDCMP1 + DATA$ACCEPTEDCMP2 + DATA$ACCEPTEDCMP3 + DATA$ACCEPTEDCMP4 + DATA$ACCEPTEDCMP5 + DATA$RESPONSE
```

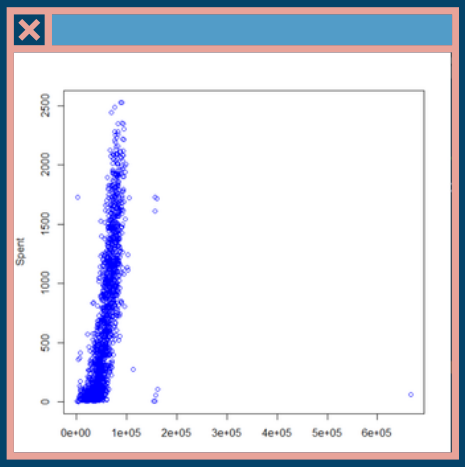
```
BOXPLOT(DATA$TOTAL_EXPENSE,
MAIN = "TOTAL SPENT BY THE CUSTOMER", XLAB = "TOTAL EXPENSES",
COL = "RED", HORIZONTAL = T, NOTCH = T )
```

```
HIST(DATA$TOTAL_EXPENSES , XLAB = "TATAL EXPENSES" , COL = "GREY")
```

```
PLOT(DATA$INCOME , DATA$TOTAL_EXPENSES ,
XLAB = "INCOME" , YLAB = "SPENT" , COL = "BLUE")
```







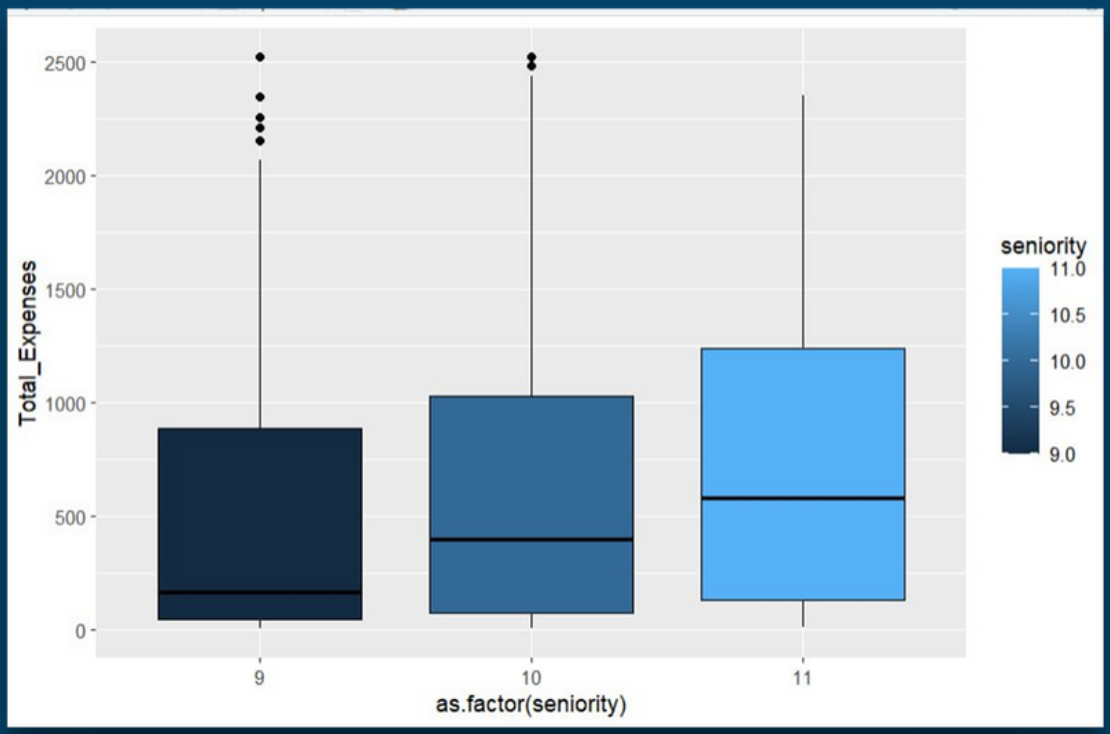
Data Transformation!

the process of converting, cleansing, and structuring data into a usable format that can be analyzed to support decision-making processes, and to propel the growth of an organization

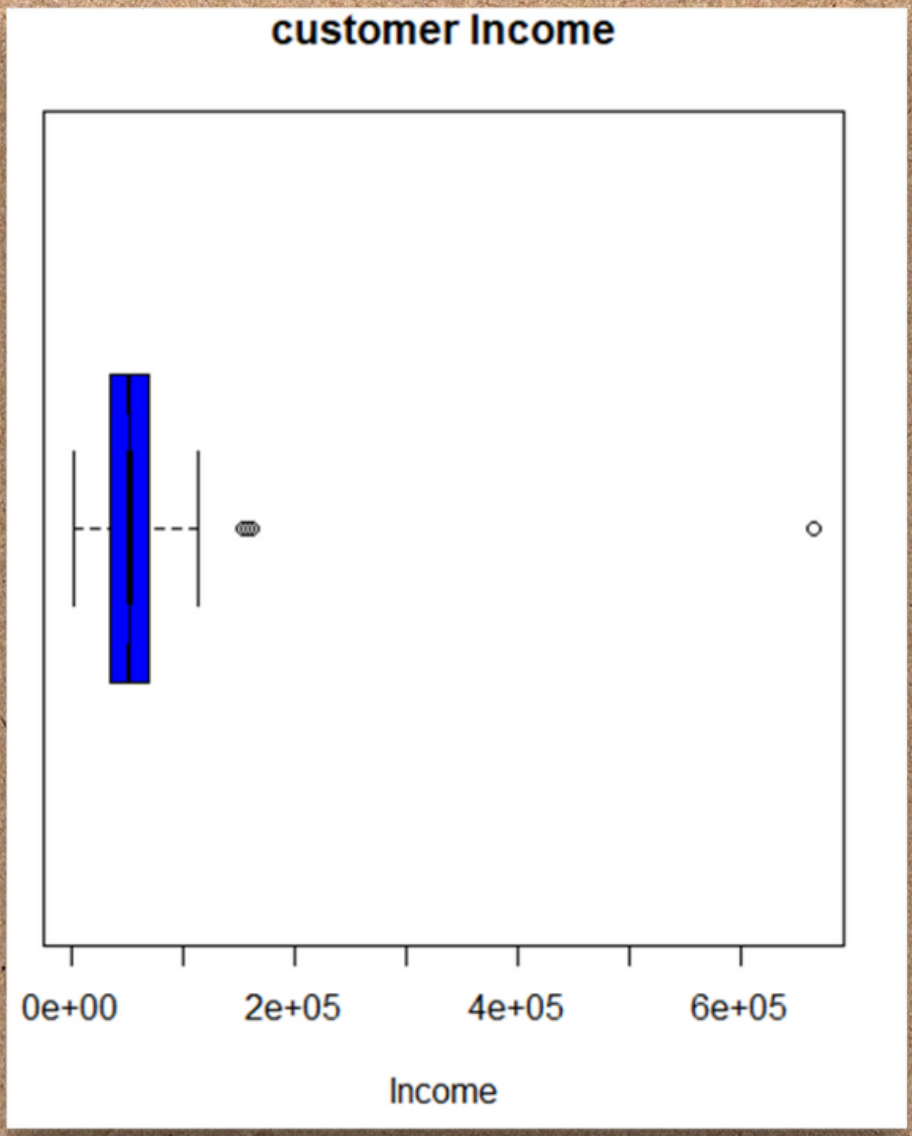
```
#DATA TRANSFORMATION
DATA$EDUCATION[DATA$EDUCATION  == "UG"] = 0
DATA$EDUCATION[DATA$EDUCATION  == "PG"] = 1
IS.NUMERIC(DATA$EDUCATION)
DATA$EDUCATION <- AS.NUMERIC(DATA$EDUCATION)
DATA$MARITAL_STATUS[DATA$MARITAL_STATUS == "SINGLE"] =
0
DATA$MARITAL_STATUS[DATA$MARITAL_STATUS == "COUPLE"] =
1
DATA$MARITAL_STATUS <-
AS.NUMERIC(DATA$MARITAL_STATUS)
```

-ToDoz-

- convert Education & Maritial_Status (nominal) to numeric data (0,1)



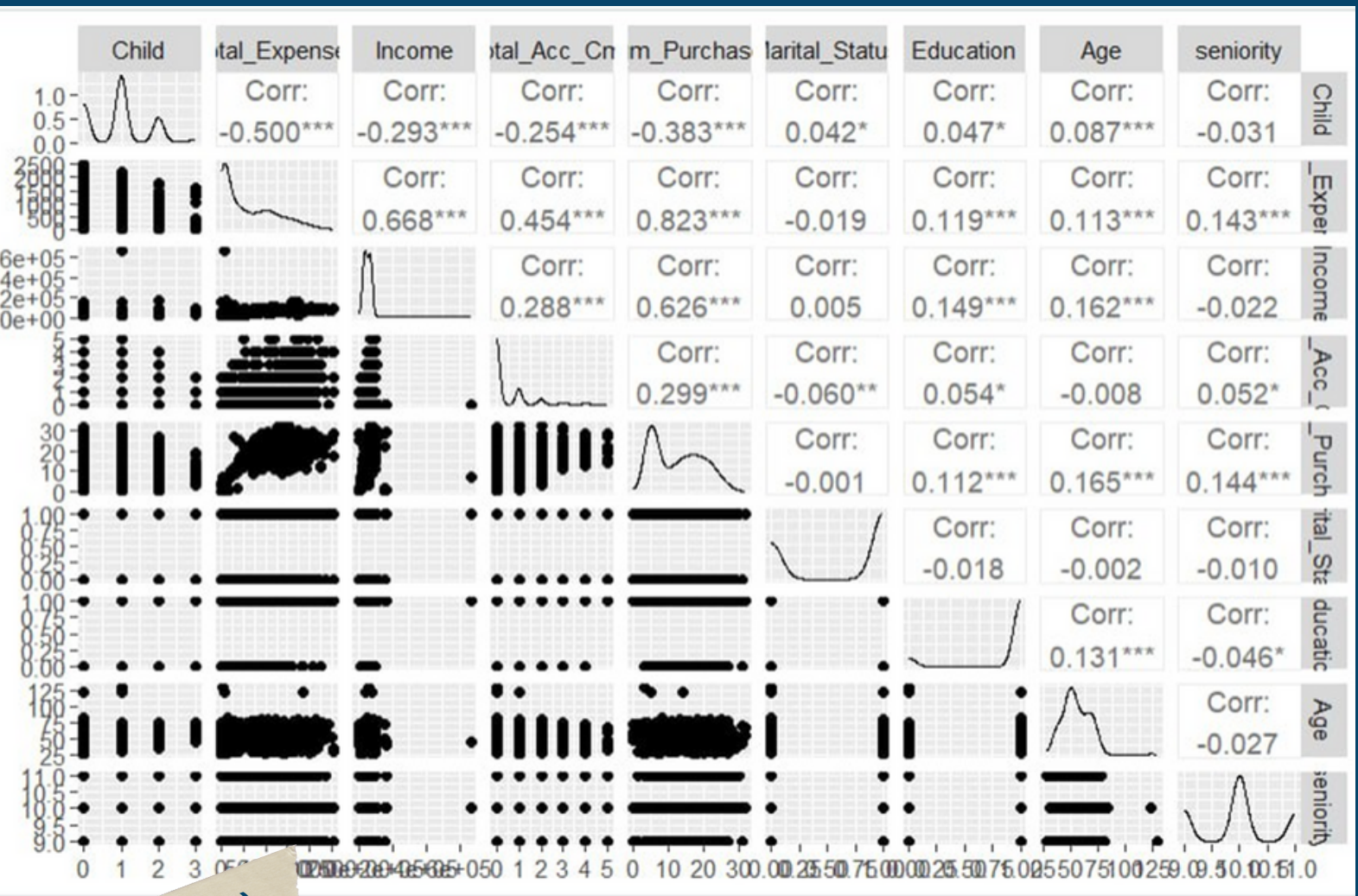

```
GGPLOT(DATA , AES(AS.FACTOR(SENIORITY) , TOTAL_EXPENSES , FILL = SENIORITY)) +
GEOM_BOXPLOT(COL = "BLACK")
```




```
BOXPLOT(DATA$INCOME, MAIN = "CUSTOMER INCOME", XLAB = "INCOME",
COL = "BLUE", HORIZONTAL = T, NOTCH = T )
```

Data Visualization!

the process of converting, cleansing, and structuring data into a usable format that can be analyzed to support decision-making processes, and to propel the growth of an organization



GGPAIRS(DF)

EACH BOX IN THE LOWER HALF DIAGONAL GIVES THE SCATTER PLOT WITH A LINE OF THE BEST FIT BETWEEN TWO VARIABLES WHICH SHOWS THE DISPERSION OF DATA AROUND THE LINE.

THERE IS SLIGHT TREND BETWEEN EDUCATION AND INCOME

PEOPLE WITH HIGHER STUDIES EARN MORE

DIMENSIONALITY REDUCTION

STRONG CORRELATION BETWEEN INCOME , TOTAL EXPENSES

FEATURE SUBSET SELECTION : WILL SELECT INCOME AND TOTAL EXPENSE TO SEE THE CORRELATION BETWEEN THEM (0.7)

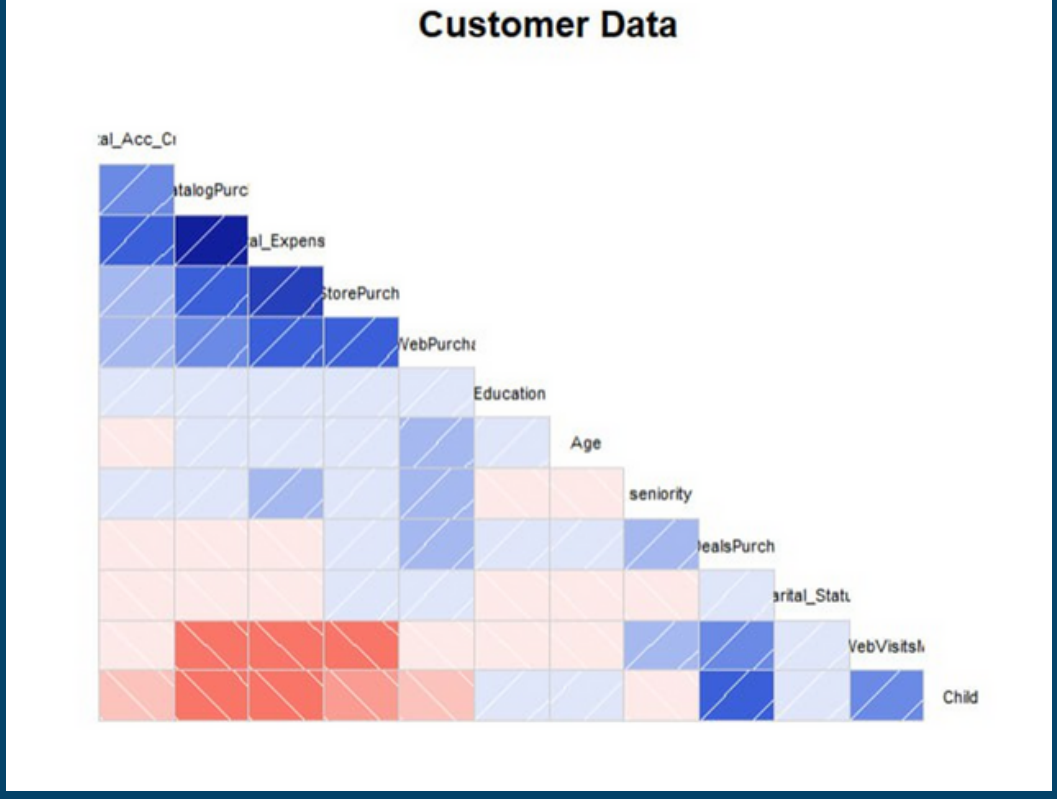
WITH FILTER METHOD CORRELATION WE WILL ELIMINATE INCOME COLUMN

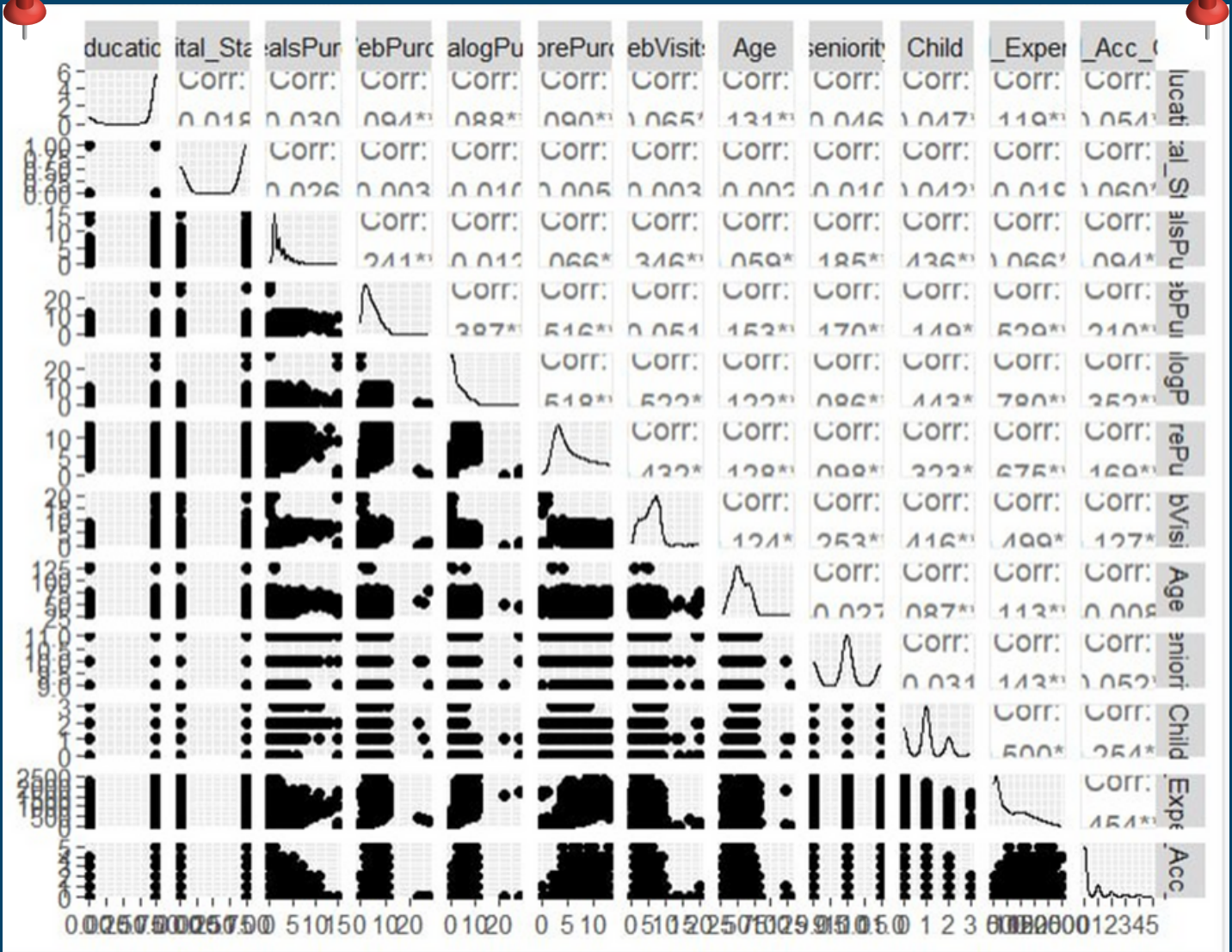
```
DF <- DATA[,!NAMES(DATA) %IN% C ("ID" , "YEAR_BIRTH" ,
                                 "KIDHOME" , "TEENHOME" ,
                                 "DT_CUSTOMER" , "RECENCY" ,
                                 "ACCEPTEDCMP3" , "ACCEPTEDCMP4" ,
                                 "ACCEPTEDCMP5" ,
                                 "ACCEPTEDCMP1" , "ACCEPTEDCMP2" ,
                                 "COMPLAIN" ,
                                 "Z_COSTCONTACT" , "Z_REVENUE" , "RESPONSE"
                                 ,
                                 "CUSTOMER_YEAR" , "MNTWINES" , "MNTFRUITS"
                                 , "MNTMEATPRODUCTS" ,
                                 "MNTFISHPRODUCTS" , "MNTSWEETPRODUCTS"
                                 ,"MNTGOLDPRODS" , "INCOME")]
```

```
> head(df)
  Education Marital_Status NumDealsPurchases NumWebPurchases NumCatalogPurchases NumStorePurchases
1         1              0                 3                8                  10                 4
2         1              0                 2                1                   1                 2
3         1              1                 1                8                   2                10
4         1              1                 2                2                   0                 4
5         1              1                 5                5                   3                 6
6         1              1                 2                6                   4                10
  NumWebVisitsMonth Age seniority Child Total_Expenses Total_Acc_Cmp
1                 7  66        11     0            1617             1
2                 5  69         9     2              27             0
3                 4  58        10     0             776             0
4                 6  39         9     1              53             0
5                 5  42         9     1             422             0
6                 6  56        10     1             716             0
```

```
CORRGRAM(DF, ORDER=T, LOWER.PANEL=PANEL.SHADE,
UPPER.PANEL=NULL,
TEXT.PANEL=PANEL.TXT, MAIN="CUSTOMER DATA")
```

CORRGRAM (CORRELATION MATRIX): USEFUL TO HIGHLIGHT THE MOST CORRELATED VARIABLES IN DATA.





NOW OUR DATA IS READY TO BUILD MODEL

Clustering

K-means

Clustering is one of the most data analysis technique used to find subgroups(cluster) of observations within a data set such that observations in the same cluster are similar while observations in different clusters are very different , clustering is considered an unsupervised learning method since we don't have the ground truth to compare the output of the clustering algorithm to the true labels to evaluate its performance.

K-means clustering is the most commonly used unsupervised machine learning algorithm for partitioning a given data set into a set of k groups. The basic idea behind k-means clustering consists of defining clusters so that the total intra-cluster variation is minimized. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's (Euclidean distances) is at the minimum.

The k-means algorithm can be summarized as follows:

1-Specify the number of clusters (K)

2-elect randomly k objects from the data set as the initial cluster centers

3-Assigns each observation to their closest centroid, based on the Euclidean distance between the object and the centroid

4-For each of the k clusters update the cluster centroid by calculating the new mean values of all the data points in the cluster.

5-Iterate steps 3 and 4 until the cluster assignments stop changing or the maximum number of iterations is reached

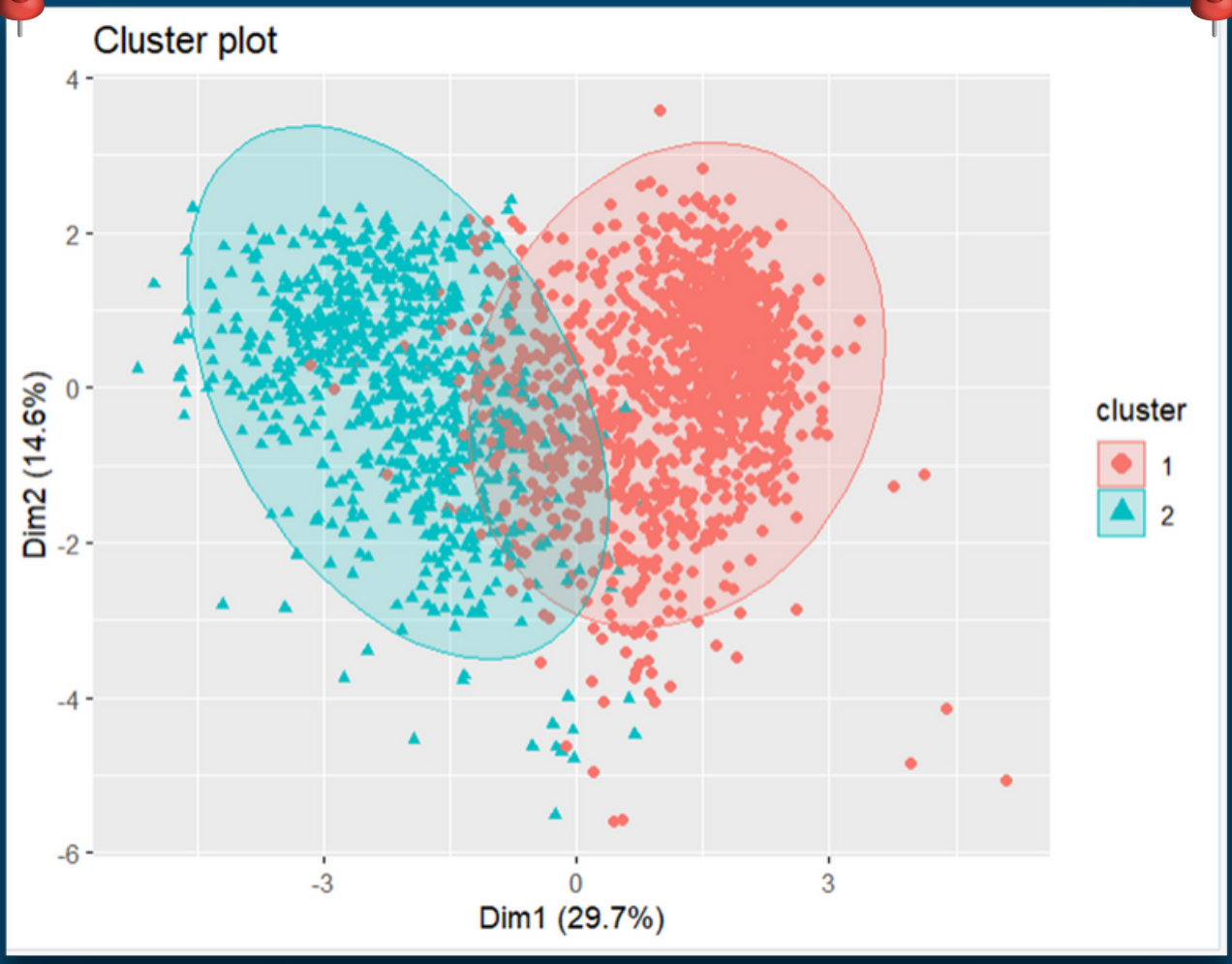
Kmeans in R:

- We can compute k-means in R using kmeans function. We will group data into two groups chosen by random , the kmeans function also has an nstart option that attempts multiple initial configurations and reports on the best one.

```
SET.SEED(12)
KMEAN <- KMEANS(DF, 2, NSTART = 10)
```

NOW WE CAN VISUALIZE OUR CLUSTER USING FVIZ_CLUSTER FUNCTION

```
FVIZ_CLUSTER(KMEAN, DF, GEOM = "POINT",ELLIPSE.TYPE =
"NORM",REPEL = TRUE)
```



Now we will use between the sum of a square and total sum of the square to Ratio of distance between clusters.
we can see in the scatter plot, there is different density into each cluster so we're not sure that 2 clusters are the best Clustering to our data.
So to get the right number of clusters we will use elbow and silhouette method

Elbow Method

code documentation :

This code performs the elbow method for finding the optimal number of clusters in a k-means clustering algorithm, using the flexclust and ggplot2 packages in R.

Here is a breakdown of the code:

The flexclust package is loaded using the library() function.

A function called kElbow() is defined to find the elbow point in the within-cluster sum of squares (WSS) curve. The function takes a vector of WSS values and returns the index of the elbow point.
*** further explanation :

The purpose of this function is to find the "elbow point" in the within-cluster sum of squares (WSS) plot. The elbow point is the value of k where the rate of decrease in WSS starts to level off, and it is typically used as an indicator of the optimal number of clusters.

Here's what each line of the function does:

n <- length(wss) calculates the length of the WSS vector.

ss_diff <- (n - 2):(1 + n %% 2) generates a sequence of values representing the differences in WSS between adjacent values of k, starting from k = 2. This is done by subtracting the WSS at k-1 from the WSS at k, for each k from 2 to n. The (n - 2) and (1 + n %% 2) values ensure that only the "odd" differences are included in the sequence.

elbow <- ss_diff[which.max(-diff(wss[ss_diff]))] + 1 finds the elbow point by identifying the index of the maximum negative difference in the ss_diff vector, and adding 1 to it to get the corresponding value of k. The negative sign is used to find the point of maximum decrease in WSS.

Finally, the function returns the value of elbow.

So, when we call the kElbow function later on in the code, it will take the WSS vector wss as input and return the value of k at the elbow point.

The within-cluster sum of squares (WSS) is computed for values of k ranging from 1 to 15 using the sapply() function and the kmeans() function from the flexclust package.

The resulting WSS values are plotted against the number of clusters using ggplot2 in the variable elbow_plot. The x-axis shows the number of clusters, and the y-axis shows the WSS.

The resulting plot is displayed using the elbow_plot variable.

The elbow method helps to identify the optimal number of clusters by identifying the point on the curve where the rate of decrease in WSS starts to level off. The location of the elbow point on the curve indicates the optimal number of clusters to use in the k-means clustering algorithm.

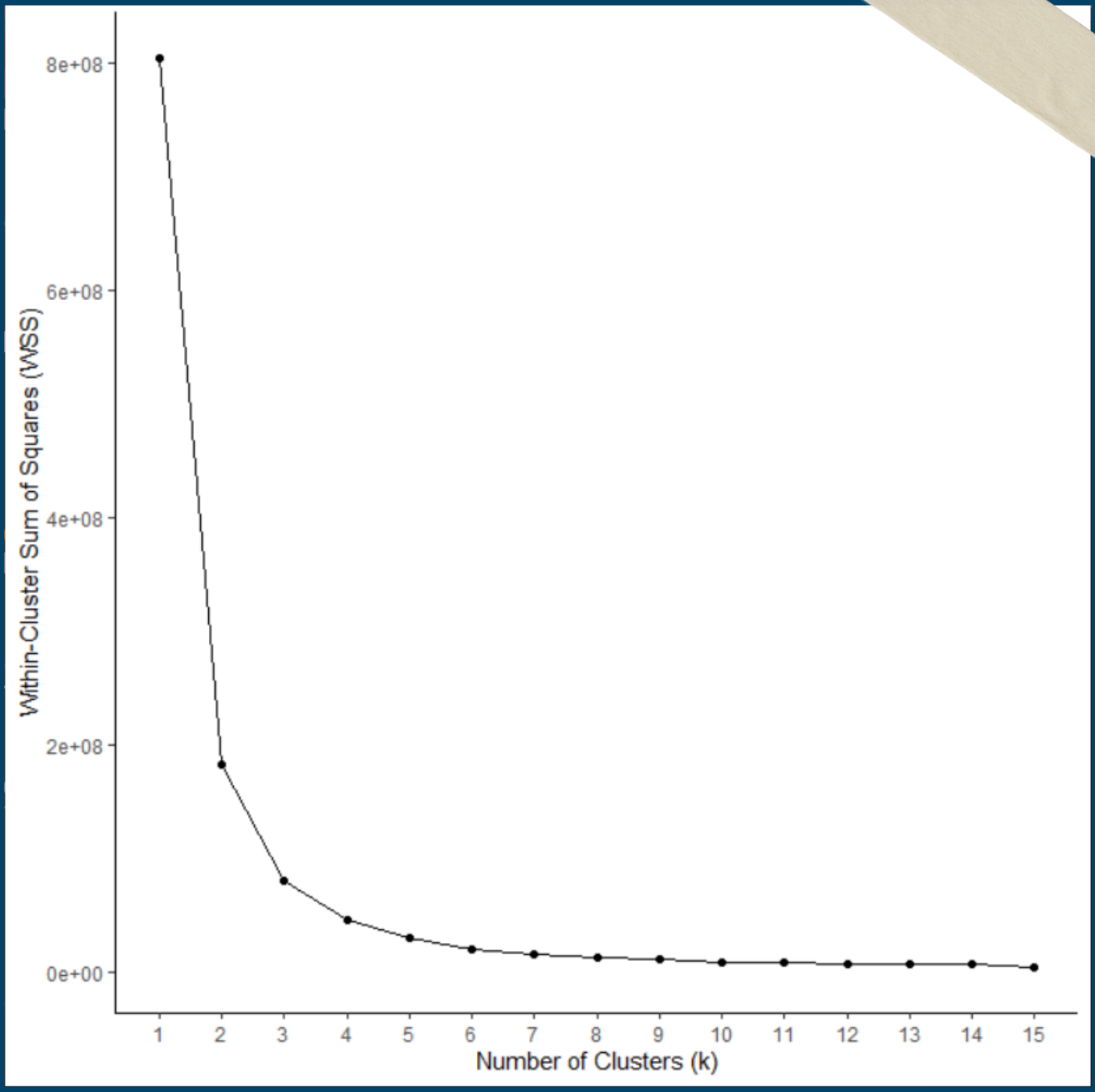
elbow_plot is a variable that stores a plot generated by the ggplot2 package in R.

The plot displays the within-cluster sum of squares (WSS) against the number of clusters used in the k-means clustering algorithm.

The x-axis of the plot shows the number of clusters, while the y-axis shows the WSS value. The WSS is a measure of the total distance between each observation and its assigned centroid within a cluster. When we use k-means clustering to group observations into k clusters, we can compute the WSS for each value of k. As the number of clusters increases, the WSS generally decreases because more centroids can be used to better fit the observations. However, at some point, adding more clusters will only result in a marginal improvement in the WSS.

The goal of the elbow method is to identify the optimal number of clusters by examining the shape of the WSS curve. The elbow point is the value of k at which the WSS starts to level off, indicating that adding more clusters is not resulting in a significant improvement in the clustering performance.

In the elbow_plot variable, the geom_point() function is used to add points to the plot, and the geom_line() function is used to connect the points with a line. The scale_x_continuous() function sets the breaks on the x-axis to be sequential integers from 1 to 15. The labs() function is used to set the x-axis and y-axis labels, and the theme_classic() function is used to set a classic theme for the plot.


8e+08
6e+08
4e+08
2e+08
0e+00
Within-Cluster Sum of Squares (WSS)
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Number of Clusters (k)

silhouette Method

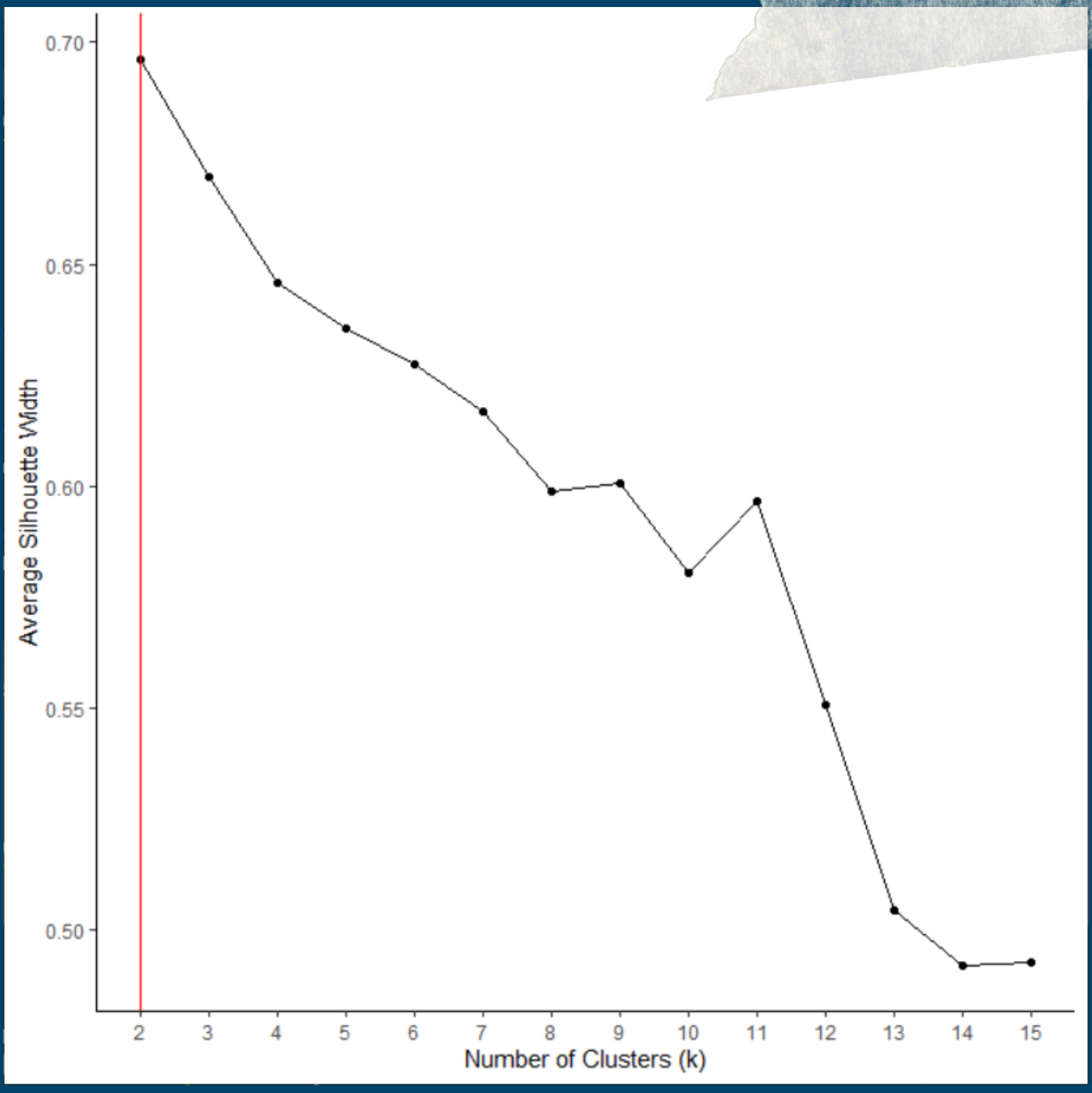
code documentation :

The silhouette method is a technique used to evaluate the quality of clustering.

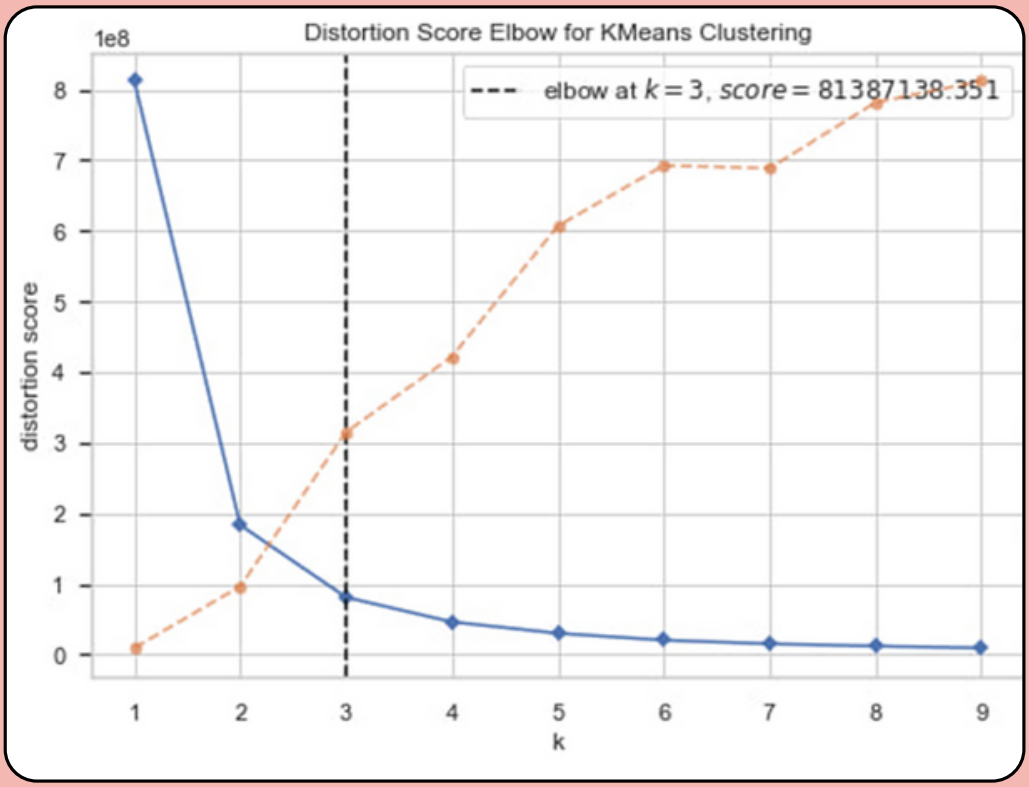
The code first initializes an empty vector called sil_width to store the silhouette width values for each value of k. The for loop iterates over the values of k from 2 to 15, and for each value of k, it performs a K-means clustering on the dataset df. The resulting clusters are used to calculate the silhouette width, which is then stored in the sil_width vector.

After iterating over all values of k, the code finds the optimal number of clusters (optimal_k) based on the highest average silhouette width. This is done by finding the index of the maximum value in the sil_width vector and adding 1 to it.

Finally, the code creates a plot of the silhouette width for each k value using ggplot. The red vertical line indicates the optimal number of clusters, which was determined in the previous step. This plot helps visualize the optimal number of clusters by looking for the peak value of the silhouette width.


0.70
0.65
0.60
0.55
0.50
Average Silhouette Width
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Number of Clusters (k)

BECAUSE OF THE DIFFERENCE IN THE NUMBER OF K BETWEEN THE ELBOW (K=3) AND THE SILHOUETTE (K=2), SO WE WILL PLOT THEM TOGETHER TO FIND THE POINT OF INTERSECTION (OPTIMAL K)

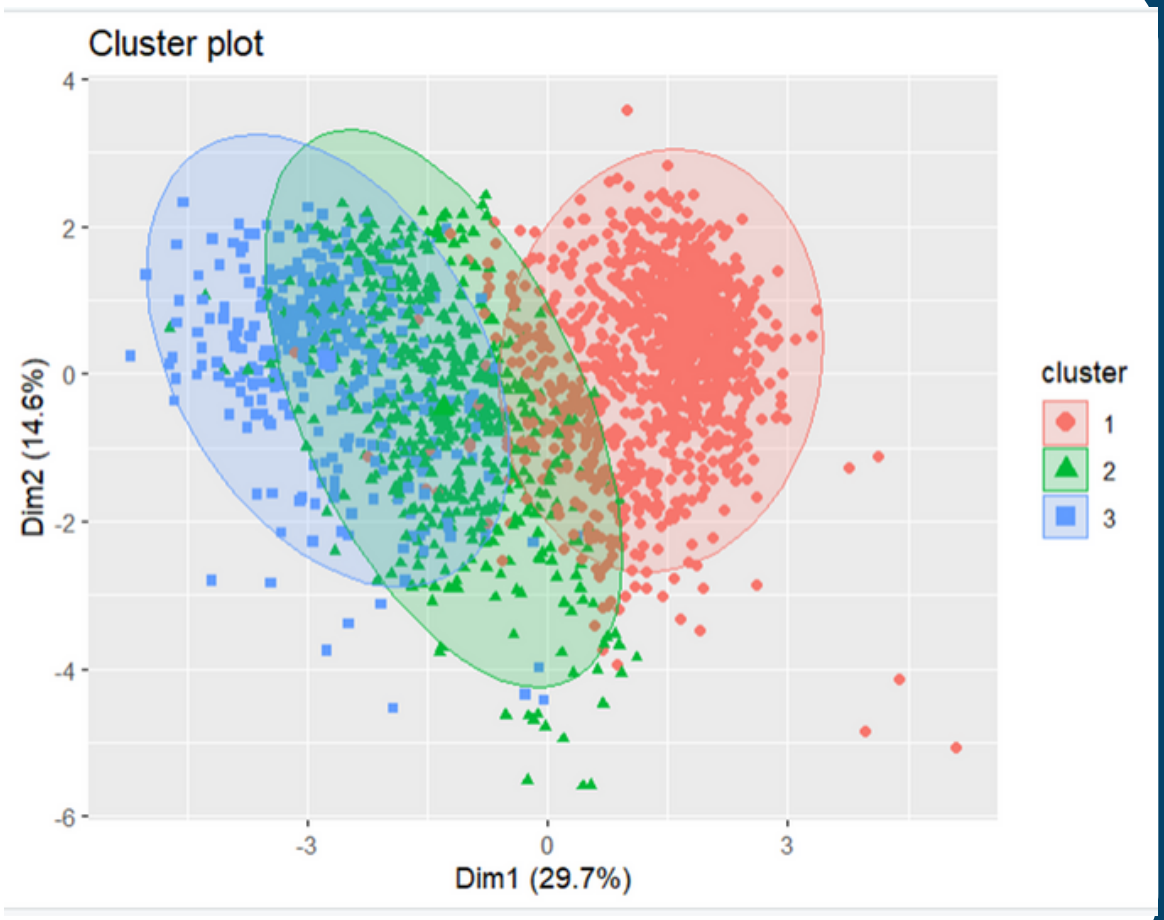


FROM THE ABOVE PLOT, WE CONCLUDE THAT THE NUMBER OF K = 3

```
> PRINT((KMEAN$BETWEENSS/KMEAN$TOTSS)*100)
[1] 77.41087
```

```
SET.SEED(12)
KMEAN <- KMEANS(DF, 3, NSTART = 10)
```

```
> PRINT((KMEAN$BETWEENSS/KMEAN$TOTSS)*100)
[1] 89.97024
```



We can say that the points that have a low density in 2 clusters get another classification, and the density in the new 3 clusters is better than it in the previous Clustering. And the ratio between the points and each other, which is wss / tss, is low. So we have reached the correct clustering.

```
> kmean
K-means clustering with 3 clusters of sizes 1256, 601, 359

Cluster means:
  Education Marital_Status NumDealsPurchases NumWebPurchases NumCatalogPurchases NumStorePurchases
1 0.8606688      0.6528662          2.340764        2.719745           0.8073248          3.730892
2 0.9001664      0.6405990          2.728785        6.093178           4.3777038          8.660566
3 0.9470752      0.6267409          1.584958        5.501393           6.3342618          8.256267
  NumWebVisitsMonth      Age seniority     Child Total_Expenses Total_Acc_Cmp
1          6.314490 52.77150  9.909236 1.2300955       152.6863     0.2070064
2          4.379368 56.68220 10.004992 0.7504160       921.4676     0.4326123
3          3.409471 54.91643 10.133705 0.2869081      1670.4819     1.3203343
```

Cluster 1: spend very low count (Total Expense), average age between 50 – 54, very low number of web purchases & catalog purchases & store purchases, visit the website frequently

Cluster 2: spend average amount, average age between 54 – 58, the biggest number of web purchases $ store purchases, visit the website little

Cluster 3: spend very high, average age between 52 and 56, a high number of web purchases and store purchases, the biggest number of catalog purchases, visit the website rarely

Hierarchical clustering

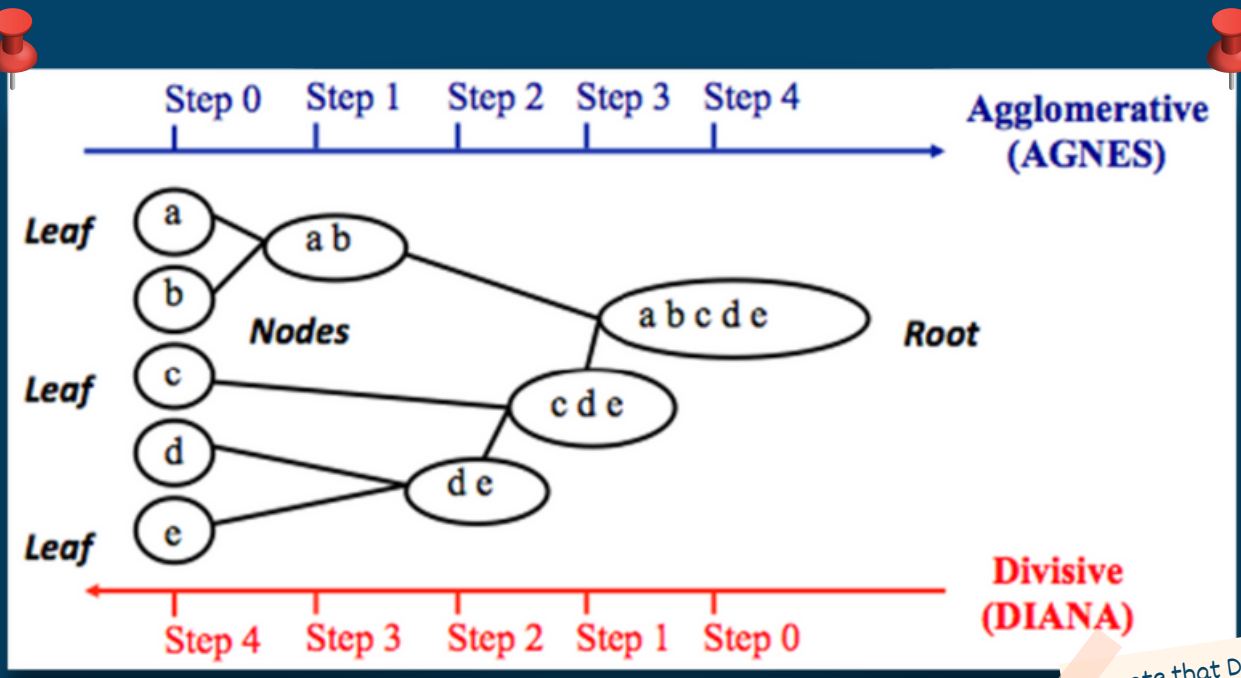
Agglomerative and Divisive clustering

IN DATA MINING AND STATISTICS, HIERARCHICAL CLUSTERING ANALYSIS IS A METHOD OF CLUSTER ANALYSIS THAT SEEKS TO BUILD A HIERARCHY OF CLUSTERS I.E. TREE-TYPE STRUCTURE BASED ON THE HIERARCHY.

BASICALLY, THERE ARE TWO TYPES OF HIERARCHICAL CLUSTER ANALYSIS STRATEGIES :

1. **AGGLOMERATIVE CLUSTERING:** ALSO KNOWN AS BOTTOM-UP APPROACH OR HIERARCHICAL AGGLOMERATIVE CLUSTERING (HAC). A STRUCTURE THAT IS MORE INFORMATIVE THAN THE UNSTRUCTURED SET OF CLUSTERS RETURNED BY FLAT CLUSTERING LIKE K-MEANS. THIS CLUSTERING ALGORITHM DOES NOT REQUIRE US TO PRESPECIFY THE NUMBER OF CLUSTERS. IN THIS METHOD, EACH OBSERVATION IS ASSIGNED TO ITS OWN CLUSTER. THEN, THE SIMILARITY (OR DISTANCE) BETWEEN EACH OF THE CLUSTERS IS COMPUTED AND THE TWO MOST SIMILAR CLUSTERS ARE MERGED INTO ONE. FINALLY, STEPS 2 AND 3 ARE REPEATED UNTIL THERE IS ONLY ONE CLUSTER LEFT.

2. **DIVISIVE METHOD:** IN DIVISIVE METHOD WE ASSUME THAT ALL OF THE OBSERVATIONS BELONG TO A SINGLE CLUSTER AND THEN DIVIDE THE CLUSTER INTO TWO LEAST SIMILAR CLUSTERS. THIS IS REPEATED RECURSIVELY ON EACH CLUSTER UNTIL THERE IS ONE CLUSTER FOR EACH OBSERVATION. THIS TECHNIQUE IS ALSO CALLED DIANA, WHICH IS AN ACRONYM FOR DIVISIVE ANALYSIS.



note that Divisive method is good for identifying large clusters while Agglomerative method is good for identifying small clusters.

Computing Distance Matrix (for update):

While merging two clusters we check the distance between two every pair of clusters and merge the pair with least distance/most similarity. But the question is how is that distance determined. There are different ways of defining Inter Cluster distance/similarity. Some of them are:

1. Min Distance: Find minimum distance between any two points of the cluster.
2. Max Distance: Find maximum distance between any two points of the cluster.
3. Group Average: Find average of distance between every two points of the clusters

Hierarchical clustering in R

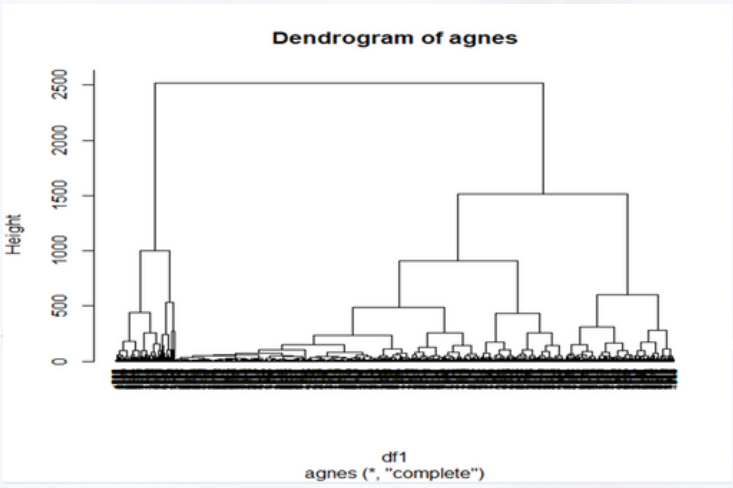
Agglomerative and Divisive clustering

THERE ARE SEVERAL FUNCTIONS AVAILABLE IN R FOR HIERARCHICAL CLUSTERING.

HERE ARE SOME COMMONLY USED ONES:
'HCLUST' (STATS PACKAGE) AND 'AGNES' (CLUSTER PACKAGE) FOR AGGLOMERATIVE HIERARCHICAL CLUSTERING
'DIANA' (CLUSTER PACKAGE) FOR DIVISIVE HIERARCHICAL CLUSTERING

```
# COMPUTE WITH AGNES (PACKAGE CLUSTER)
HC2 <- AGNES(DF, METHOD = "COMPLETE")
PLTREE(HC2, CEX = 0.6, HANG = -1, MAIN = "DENDROGRAM OF AGNES")
# AGGLOMERATIVE COEFFICIENT
HC2$AC
```

_Agglomerative _ we need to specify the linkage method we want to use ("complete", "average", "single")



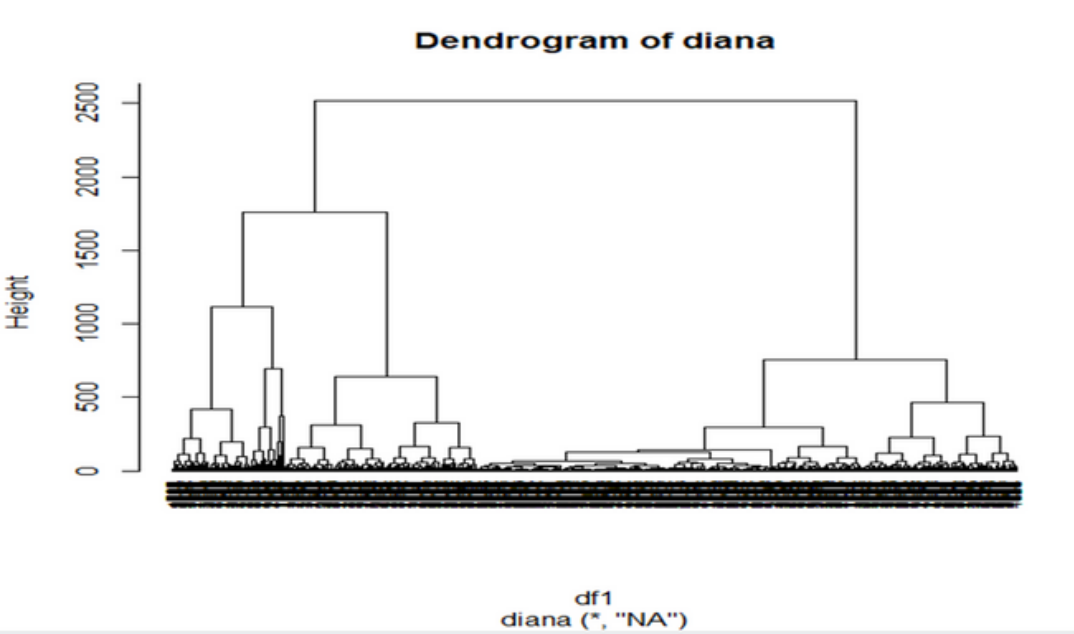
WITH THE AGNES FUNCTION YOU CAN ALSO GET THE AGGLOMERATIVE COEFFICIENT, WHICH MEASURES THE AMOUNT OF CLUSTERING STRUCTURE FOUND (VALUES CLOSER TO 1 SUGGEST STRONG CLUSTERING STRUCTURE).
OUR AGGLOMERATIVE COEFFICIENT HERE IS:

```
> HC2$AC 0.9978071
```

DIVISIVE HIERARCHICAL CLUSTERING

THE FUNCTION 'DIANA' IN THE CLUSTER PACKAGE HELPS US PERFORM DIVISIVE HIERARCHICAL CLUSTERING. INSTEAD OF AGGLOMERATIVE COEFFICIENT, WE HAVE DIVISIVE COEFFICIENT.

```
HC4 <- DIANA(DF)
# PLOT DENDROGRAM
PLTREE(HC4, CEX = 0.6, HANG = -1, MAIN = "DENDROGRAM OF DIANA")
# DIVISE COEFFICIENT
```



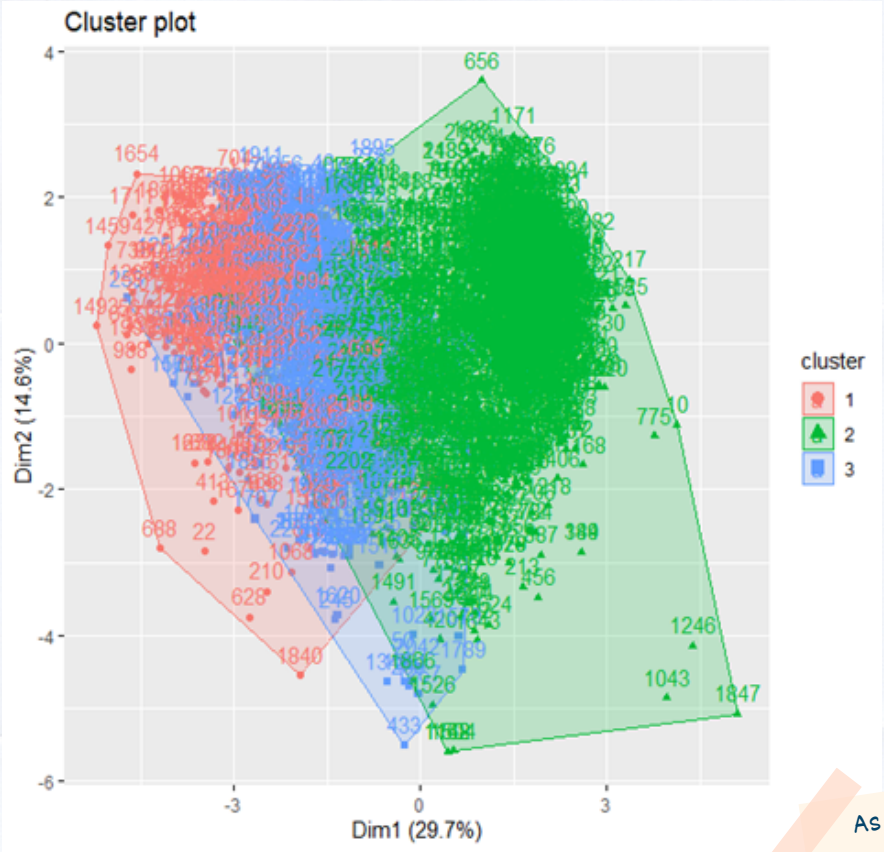
OUR DIVISIVE COEFFICIENT HERE IS:

```
> HC4$DC
[1] 0.9976146
```

the final step is assigning clusters to the data points to get a better vision, This can be done with the R function cutree. It cuts a tree (or dendrogram), as resulting from (diana/agnes), into several groups by specifying the desired number of groups (k).

assigning clusters for the tree obtained by diana function (Divisive Hierarchical Clustering):

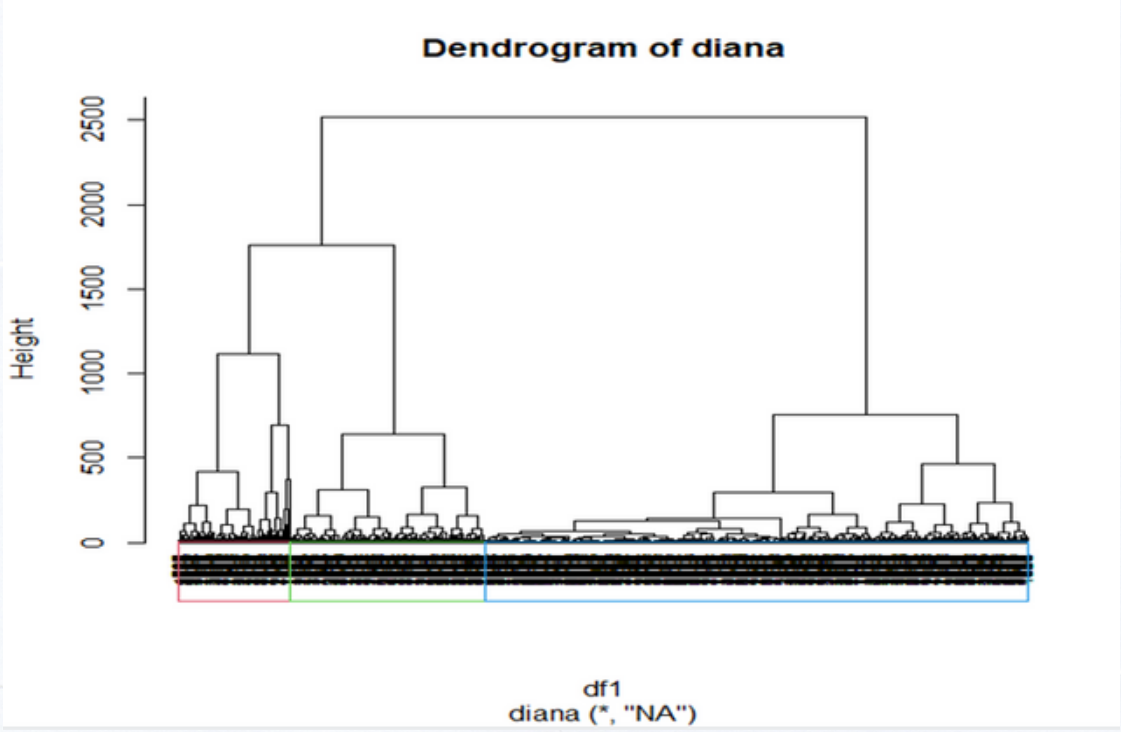
```
CLUST <- CUTREE(HC2, K = 3)
#FVIZ_CLUSTER FUNCTION FROM THE FACTOEXTRA PACKAGE
#TO VISUALIZE THE RESULT IN A SCATTER PLOT.
FVIZ_CLUSTER(LIST(DATA = DF, CLUSTER = CLUST))
```



As we can see our results from divisive clustering and agglomerative clustering are very similar to K-means result.

We can also visualize the clusters inside the dendrogram itself by putting borders as shown next:

```
#VISUALIZE THE CLUSTERS INSIDE THE DENDROGRAM
RECT.HCLUST(HC4, K = 3, BORDER = 2:10)
```



MAY 2023

Data-comp Project

Part 2

PRESENTED TO

Dr. Amr Amin

PRESENTED BY

Rahma Ezzat	20201381362
Asmaa Hamdy	20201032790
Karim Radwan	20201498197
Omar Mekkawy	20201375851

Part 2

MAY 2023

Table of Contents

DBSCAN

DENSITY-BASED SPATIAL CLUSTERING OF APPLICATIONS WITH NOISE - UNVEILING INSIGHTS AND PATTERNS IN DATA EFFICIENTLY.

AT FIRST, WE WILL FIND OPTIMAL MINIMUM POINTS & EPSILON (RADIUS)
THE DATA CONTAINS 13 FEATURES SO MIN POINT = 14 (D+1 => 2D)
THEN TO FIND THE OPTIMAL EPSILON, WE WILL CLASSIFY OUR DATA BY USING KNOWN (K = MIN POINT = 14)
WE HAVE TO ADD A SEGMENT (CLASSIFIER) TO THE DATA FOR THE CLASSIFICATION
SPLITTING THE DATASET INTO TRAINING (70%) & TESTING (30%) SETS

```
split <- sample.split(df, SplitRatio = 0.7)
train_cl <- subset(df, split == "TRUE")
test_cl <- subset(df, split == "FALSE")
length(train_cl)
train_scale <- scale(train_cl[, 1:12])
test_scale <- scale(test_cl[, 1:12])
length(train_scale)
length(test_scale)
```

```
> split <- sample.split(df, SplitRatio = 0.7)
> train_cl <- subset(df, split == "TRUE")
> test_cl <- subset(df, split == "FALSE")
> length(train_cl)
[1] 13
> train_scale <- scale(train_cl[, 1:12])
> test_scale <- scale(test_cl[, 1:12])
> length(train_scale)
[1] 18396
> length(test_scale)
[1] 8196
```

FITTING KNN MODEL TO THE TRAINING DATASET:

```
library(class)
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$Segmant,
                      k = 14)
classifier_knn
```

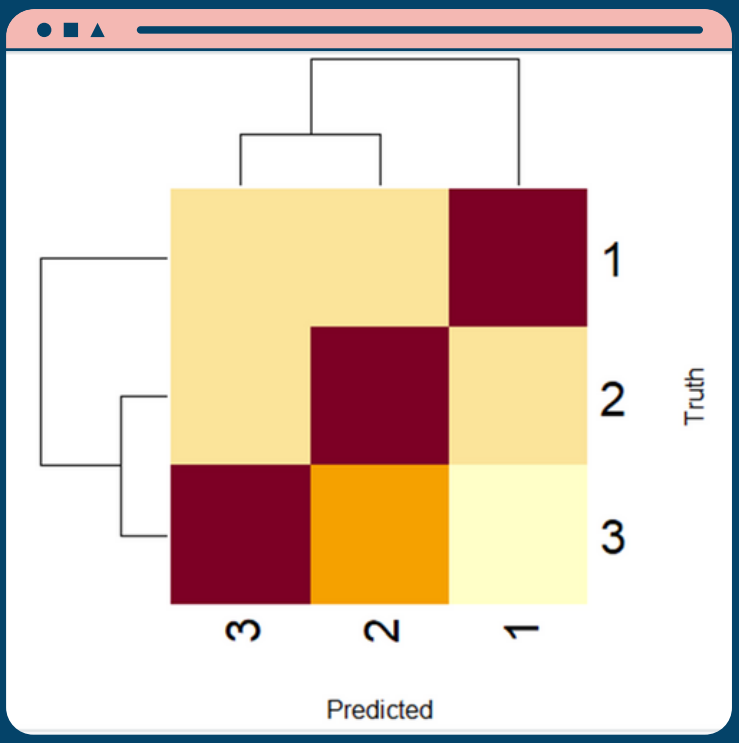
```
> classifier_knn <- knn(train = train_scale,
+                       test = test_scale,
+                       cl = train_cl$Segmant,
+                       k = 14)
> classifier_knn
  [1] 3 1 1 1 1 3 1 1 1 3 1 2 1 1 2 2 2 1 3 2 2 2 1 3 2 1 1 1 1 1 2 1 3 1 3 1 3 1
 [49] 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 2 1 2 1 1 1 1 1 1 3 3 1 1 1 2 1 1 1 1 2 1 1
 [97] 3 1 1 1 3 3 1 1 3 2 2 3 1 2 1 1 1 2 1 1 3 1 1 1 1 1 1 1 1 3 2 1 2 3 2 1 1 1
[145] 1 1 2 2 2 2 3 1 2 1 1 3 1 2 3 2 1 1 1 1 2 1 1 1 3 1 2 1 1 1 3 1 1 3 1 3 1 1
[193] 1 2 2 1 3 1 1 1 1 1 1 1 2 2 2 1 2 1 1 1 2 3 3 1 3 2 2 2 1 3 1 1 1 1 3 2 3 2
[241] 1 1 2 1 1 1 3 2 1 3 2 2 1 1 2 1 3 2 1 2 1 2 1 1 2 1 2 1 3 3 1 1 2 1 2 1 3 3
[289] 1 1 1 1 1 1 2 1 2 3 1 1 2 1 1 1 3 2 1 3 1 1 1 1 1 1 3 1 2 1 2 1 1 1 1 2 3 2
[337] 2 1 2 2 1 2 1 1 1 1 3 3 2 1 2 2 2 2 1 2 1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 1 2 1
[385] 3 3 1 1 1 2 1 3 1 3 1 1 1 3 1 1 2 1 1 1 1 1 3 2 1 1 2 1 1 1 2 3 2 1 1 1 1 1
[433] 2 2 1 1 1 1 1 1 2 2 1 1 2 2 1 1 2 2 2 2 3 1 1 3 3 1 1 2 3 2 1 1 2 2 3 1 2 2
[481] 3 1 1 1 1 2 1 1 1 1 2 1 1 1 2 2 1 2 1 2 2 1 1 2 1 1 2 1 1 2 2 2 2 1 1 1 2 1 2
[529] 3 1 2 1 1 2 3 1 2 2 2 2 2 2 1 1 1 1 2 1 1 2 3 1 1 2 2 3 2 1 3 1 1 1 1 2 3 1
[577] 1 3 1 1 1 1 2 1 1 1 1 2 2 2 1 2 1 1 1 2 1 2 1 2 2 3 2 1 1 1 1 1 2 1 3 1 2 1
[625] 1 1 3 1 2 1 1 1 2 3 1 1 1 1 3 2 2 1 1 3 1 2 1 1 1 1 1 1 1 1 2 1 1 1 2 1 3 1
[673] 1 1 2 2 1 3 1 2 1 2 2
Levels: 1 2 3
```

CONFUSION MATRIX TO DEFINE THE PERFORMANCE OF CLASSIFICATION ALGORITHM

```
> cm <- table(test_cl$Segmant, classifier_knn)
> cm
   classifier_knn
      1   2   3
  1 372   3   0
  2  13 164  11
  3   1  38  81
```

VISUALIZE THE CLASSIFICATION PROBLEMS AS A HEATMAP EACH ROW IS AN ATTRIBUTE WHERE COLUMNS CORRESPOND TO THE SEGMENT CLASS TO BE CLASSIFIED

```
heatmap(cm ,xlab = "Predicted" , ylab ="Truth")
```



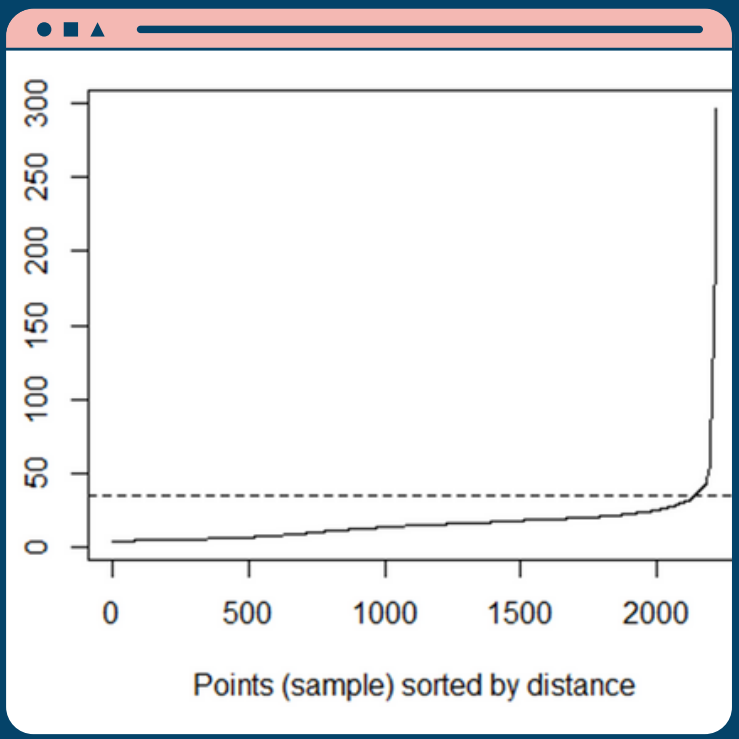
CALCULATE OUT-OF-SAMPLE ERROR TO CALCULATE ACCURACY

```
misClassError <- mean(classifier_knn != test_cl$Segmant)
print(paste('Accuracy =', 1-misClassError))
```

```
> misClassError <- mean(classifier_knn != test_cl$Segmant)
> print(paste('Accuracy =', 1-misClassError))
[1] "Accuracy = 0.903367496339678"
>
```

PLOT KNN TO GET OPTIMAL EPSILON

```
dbscan::kNNdistplot(df, k =  14)
abline(h = 35, lty = 2)
```

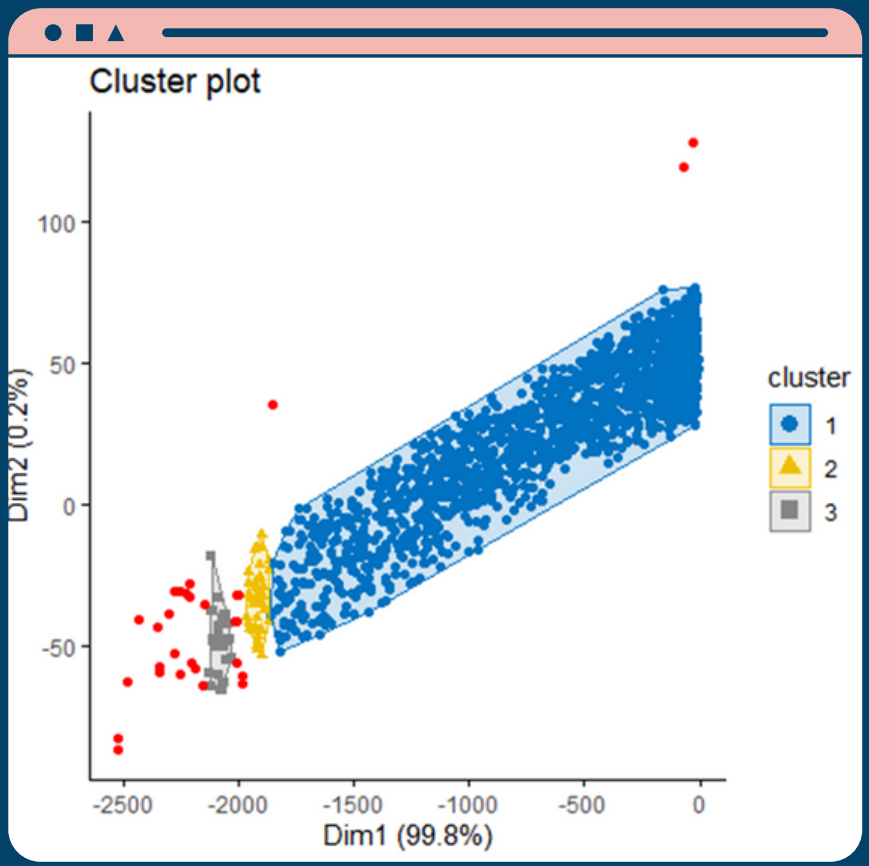


From the plot, we conclude that epsilon = 35

DBSCAN MODEL WITH MIN POINT = 14 & EPSILON = 35

```
set.seed(123)
db <- dbscan(df, eps = 35, minPts = 14)
print(db)
fviz_cluster(db, data = df, stand = F,outlier.color="Red",
             ellipse = T,geom = "point",palette = "jco" ,ggtheme = theme_classic())
```

```
> db <- dbscan(df, eps = 35, minPts = 14)
> print(db)
DBSCAN clustering for 2216 objects.
Parameters: eps = 35, minPts = 14
Using euclidean distances and borderpoints = TRUE
The clustering contains 3 cluster(s) and 32 noise points.

   0    1    2    3
  32 2129   32   23

Available fields: cluster, eps, minPts, dist, borderPoints
> fviz_cluster(db, data = df, stand = F,outlier.color="Red",
+              ellipse = T,geom = "point",palette = "jco" ,ggtheme = theme_classic())
```



PCA

Principal Component Analysis (PCA) is one of the most commonly used unsupervised machine learning algorithms across a variety of applications: exploratory data analysis, dimensionality reduction, information compression

How Does PCA Work?

1-Normalize the dataset

information with different scales can lead to a biased result when we perform PCA on it, so first we will ensure that each attribute has the same level of contribution, preventing one variable from dominating others.

2-Calculate covariance matrix

The next step is to calculate the covariance matrix of the standardized data. This matrix shows how each variable is related to every other variable in the dataset.

3-Calculate the eigenvectors and eigenvalues

The eigenvectors represent the directions in which the data varies the most, while the eigenvalues represent the amount of variation along each eigenvector.

4-Selection of principal components

The principal components are the eigenvectors with the highest eigenvalues. These components represent the directions in which the data varies the most and are used to transform the original data into a lower-dimensional space.

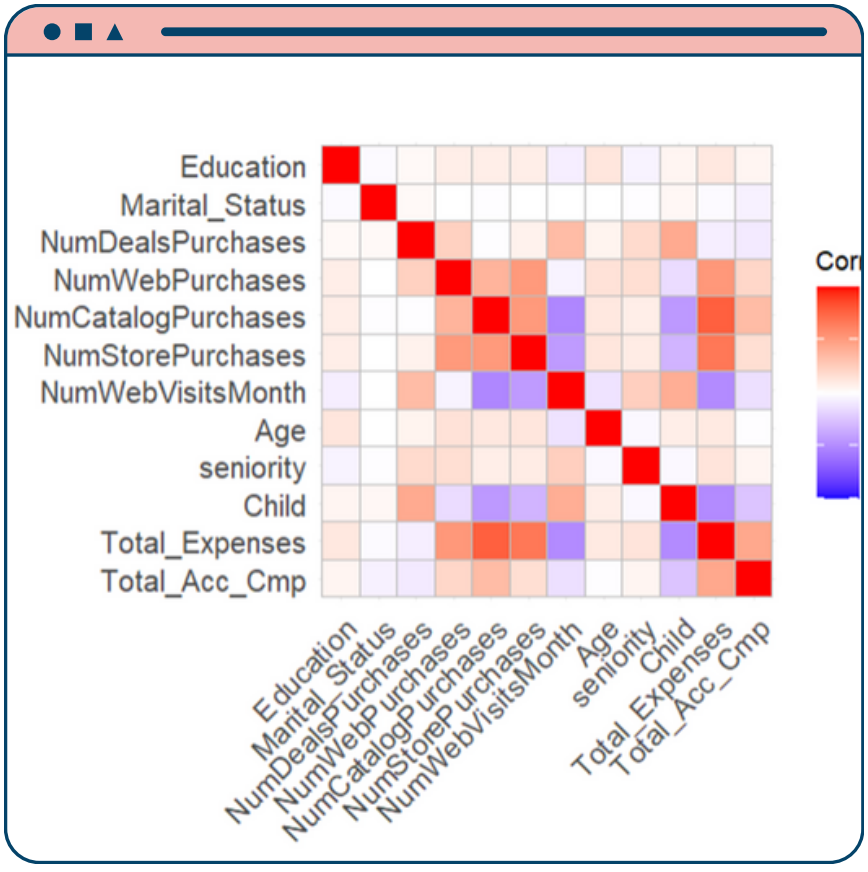
5-Transform the data

The final step is to transform the original data into the lower-dimensional space defined by the principal components.

WE WILL NORMALIZE OUR DATASET USING SCALE FUNCTION

```
> data_normalized <- scale(df)
> head(data_normalized)
  Education Marital_Status NumDealsPurchases NumWebPurchases NumCatalogPurchases NumStorePurchases NumWebVisitsMonth
1 0.3597239     -1.3485242         0.3516341       1.4282310           2.5041470       -0.55401784        0.6930755
2 0.3597239     -1.3485242        -0.1681932      -1.1256271          -0.5709535       -1.16925390       -0.1315448
3 0.3597239      0.7412168        -0.6880206       1.4282310          -0.2292757        1.29169031       -0.5438550
4 0.3597239      0.7412168        -0.1681932      -0.7607902          -0.9126314       -0.55401784        0.2807654
5 0.3597239      0.7412168         1.3912888       0.3337204           0.1124021        0.06121821       -0.1315448
6 0.3597239      0.7412168        -0.1681932       0.6985572           0.4540800        1.29169031        0.2807654
         Age   seniority       Child Total_Expenses Total_Acc_Cmp
1  0.9862203  1.50000426 -1.26451786      1.6751100     0.6179059
2  1.2365216 -1.41707291  1.40548841     -0.9621412    -0.5026174
3  0.3187501  0.04146567 -1.26451786      0.2801866    -0.5026174
4 -1.2664915 -1.41707291  0.07048527     -0.9190163    -0.5026174
5 -1.0161902 -1.41707291  0.07048527     -0.3069750    -0.5026174
6  0.1518826  0.04146567  0.07048527      0.1806677    -0.5026174
```

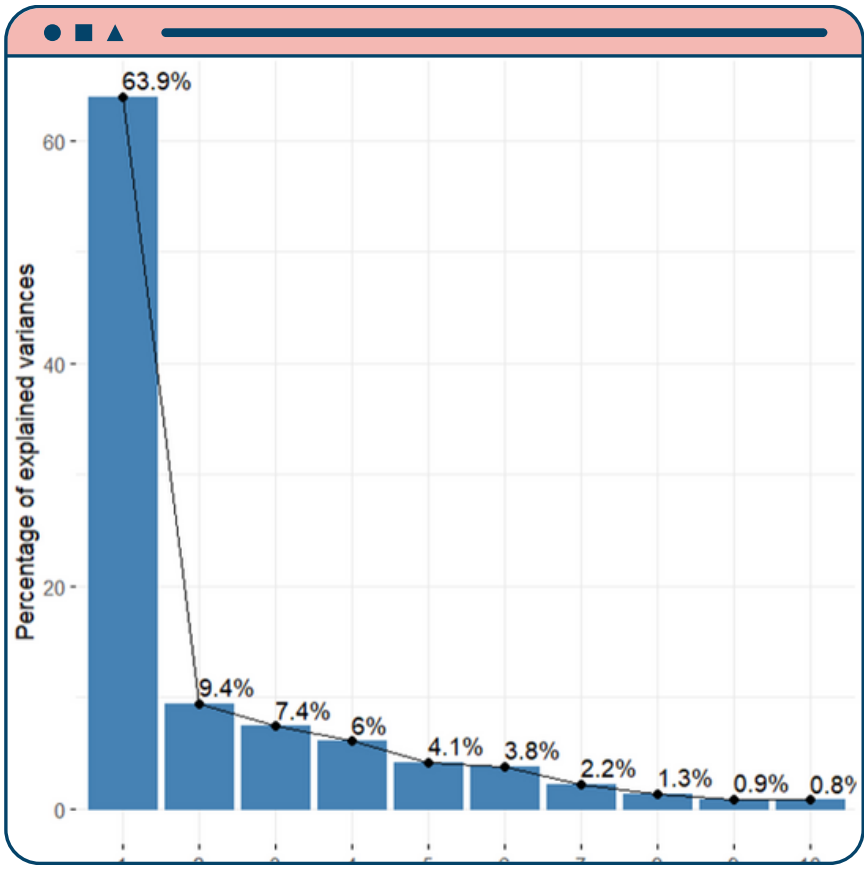
NOW ,WE WILL COMPUTE COVARIANCE MATRIX USING THE CORR() FUNCTION , FOR BETTER VISUALIZATION WE WILL USE GGCORRPLOT() FUNCTION



WE WILL COMPUTE PCA USING PRINCOMP() FUNCTION , WE USE SUMMARY() FUNCTION SHOWS THE RESULT.

```
> data.pca <- princomp(corr_matrix)
> summary(data.pca)
Importance of components:
                          Comp.1     Comp.2     Comp.3     Comp.4     Comp.5     Comp.6     Comp.7     Comp.8
Standard deviation     0.9662779 0.37125743 0.32846627 0.2966348 0.24517237 0.23464609 0.17961045 0.13946254
Proportion of Variance 0.6394813 0.09440045 0.07389335 0.0602654 0.04116867 0.03770947 0.02209464 0.01332106
Cumulative Proportion  0.6394813 0.73388172 0.80777506 0.8680405 0.90920913 0.94691860 0.96901324 0.98233430
                           Comp.9      Comp.10     Comp.11 Comp.12
Standard deviation     0.112360905 0.105453549 0.045253526       0
Proportion of Variance 0.008646776 0.007616337 0.001402583       0
Cumulative Proportion  0.990981080 0.998597417 1.000000000       1
```

THE FVIZ_EIG() SHOWS THE PROPORTION OF VARIANCE EXPLAINED BY EACH PRINCIPAL COMPONENT.



FVIZ_PCA_VAR() SHOWS THE RELATIONSHIP BETWEEN THE VARIABLES AND THE FIRST TWO PRINCIPAL COMPONENTS. THE COL.VAR = "BLACK" ARGUMENT SETS THE COLOR OF THE VARIABLE LABELS TO BLACK.

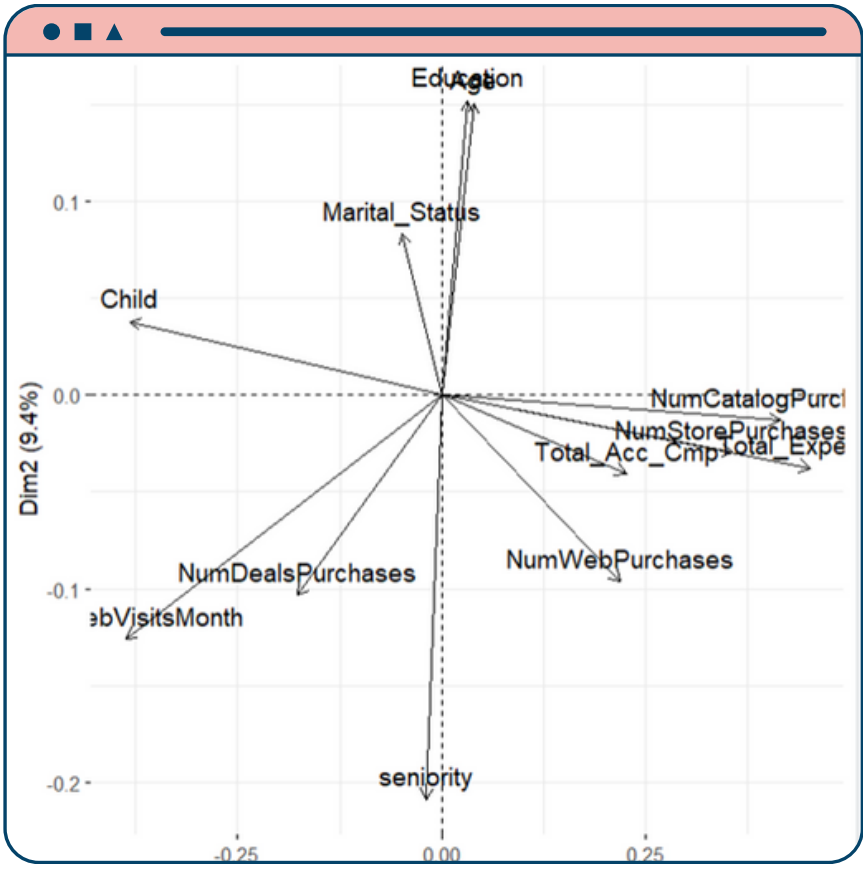
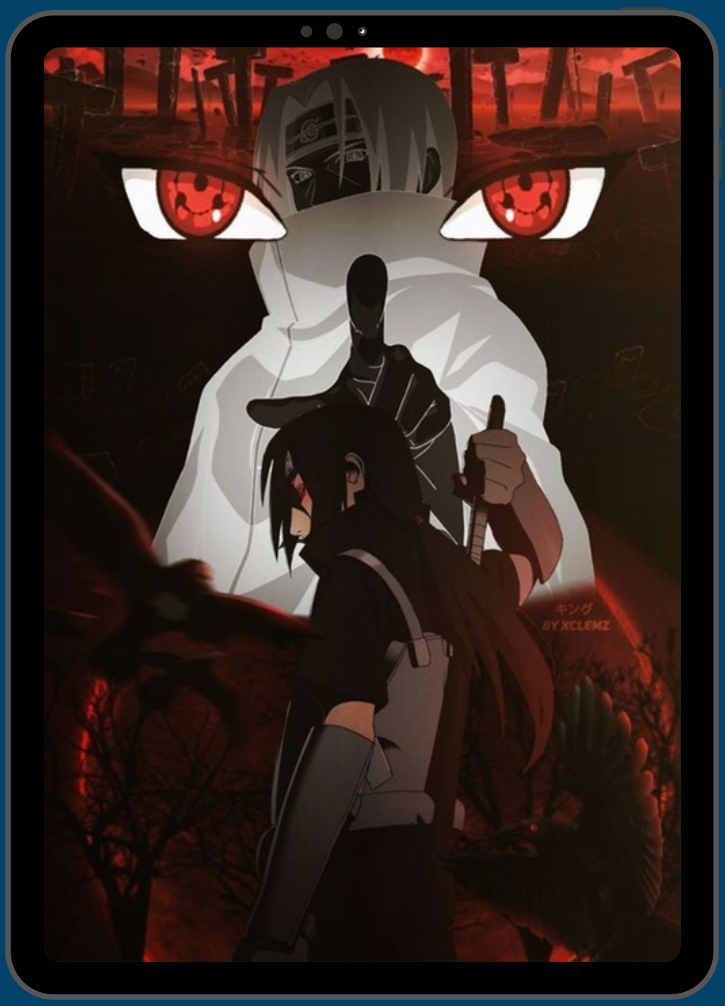


Image Compression

in

Using PCA

PCA can be used for image compression by reducing the dimensionality of the image while retaining as much of the original information as possible



the image that we will be working on

PERFORM PCA ON THE RED, GREEN, BLUE COLOR CHANNELS OF AN IMAGE REPRESENT THE IMAGE AS A THREE 1002 X 564 MATRICES ARRAY WITH EACH MATRIX CORRESPONDING TO THE RGB COLOR VALUE SCHEME AND THEN EXTRACT THE INDIVIDUAL COLOR VALUE MATRICES TO PERFORM PCA ON EACH

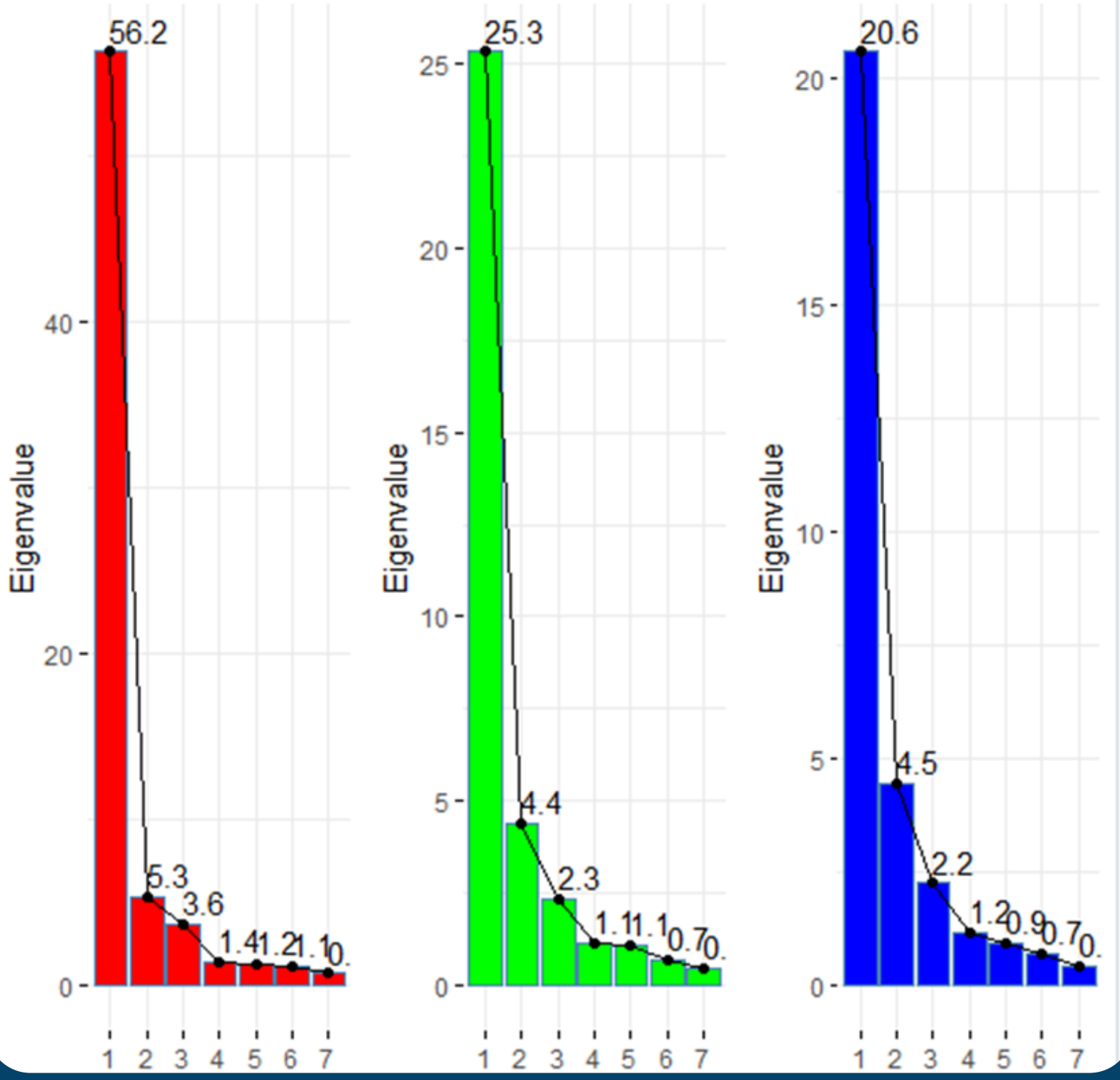
```
dim(img)
red <- img[,,1]
green <- img[,,2]
blue <- img[,,3]

red.pca <- prcomp(red, center=FALSE, scale.=FALSE)
green.pca <- prcomp(green, center=FALSE, scale.=FALSE)
blue.pca <- prcomp(blue, center=FALSE, scale.=FALSE)

list.img.pca <- list(red.pca, green.pca, blue.pca)
```

PCA TARGETS FEATURES WITH HIGHER VARIANCE
THEREFORE SCALING AND CENTERING OF OUR DATA WOULD NOT BE NECESSARY FOR IMAGE COMPRESSION
WE NOW PUT THEM TOGETHER AS A LIST TO INTEGRATE IT INTO THE THREE DIMENSIONAL TABLE OF THE RGB

```
> dim(img)
[1] 1002  564    3
> red <- img[,,1]
> green <- img[,,2]
> blue <- img[,,3]
```

PLOT THE EIGEN VALUES FOR THE PRINCIPAL COMPONENTS FOR ALL THE RGB COMPONENTS
DISPLAYING JUST (7) PRINCIPAL COMPONENTS

```
f1 <- fviz_eig(red.pca, choice = 'eigenvalue', main = "Red", barfill = "red", ncp = 7, addlabels = TRUE)
f2 <- fviz_eig(green.pca, choice = 'eigenvalue', main = "Green", barfill = "green", ncp = 7, addlabels = TR
f3 <- fviz_eig(blue.pca, choice = 'eigenvalue', main = "Blue", barfill = "blue", ncp = 7, addlabels = TRUE)

grid.arrange(f1, f2, f3, ncol=3)
```

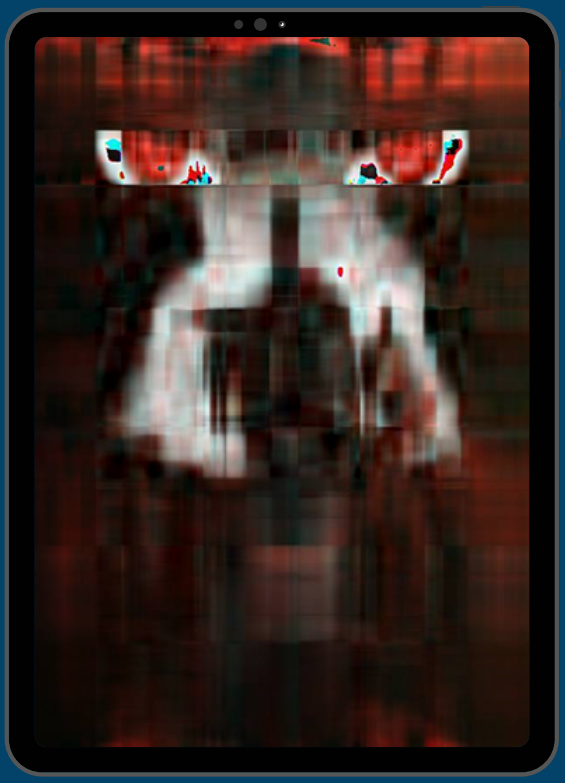

Eigenvalue
56.2
5.3
3.6
1.4
1.2
1.1
0.
40
20
0
1 2 3 4 5 6 7
Eigenvalue
25.3
4.4
2.3
1.1
1.1
0.7
0.
25
20
15
10
5
0
1 2 3 4 5 6 7
Eigenvalue
20.6
4.5
2.2
1.2
0.9
0.7
0.
20
15
10
5
0
1 2 3 4 5 6 7

NOW WE ARE GOING TO TAKE A LOOK AT HOW THE IMAGE IS GOING TO LOOK LIKE AFTER WE COMPRESS IT WITH DIFFERENT NUMBER OF PRINCIPAL COMPONENTS

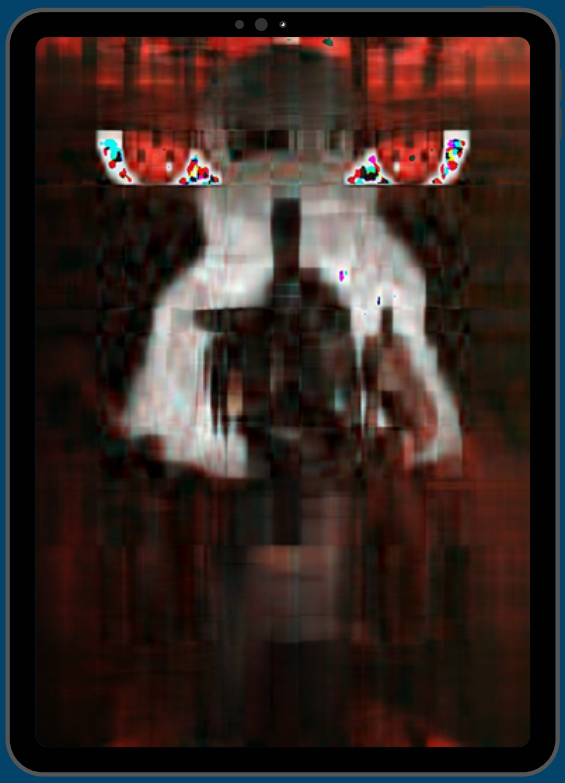
```
library(abind)

for (i in c(10,15,30,60,80,120,180,210)) {
  new_image <- abind(red.pca$x[,1:i] %*% t(red.pca$rotation[,1:i]),
                     green.pca$x[,1:i] %*% t(green.pca$rotation[,1:i]),
                     blue.pca$x[,1:i] %*% t(blue.pca$rotation[,1:i]),
                     along = 3)
  writeJPEG(new_image, paste0('Compressed_image_with_',i, '_components.jpg'))
}

image_plot <- function(path, plot_name) {
  require('jpeg')
  img <- readJPEG("C:/Users/Kimo Store/Downloads/itatci_uchiha.jpg")
  d <- dim(img)
  plot(0,0,xlim=c(0,d[2]),ylim=c(0,d[2]),xaxt='n',yaxt='n',xlab='',ylab='',bty='n')
  title(plot_name, line = -0.5)
  rasterImage(img,0,0,d[2],d[2])
}

par(mfrow = c(2,2), mar = c(0,0,1,1))
for (i in c(10,15,30,60,80,120,180,210)) {
  image_plot(paste0('Compressed_image_with_',i, '_components.jpg'),
             paste0(round(i,0), ' Components'))
}
```

RESULTS?

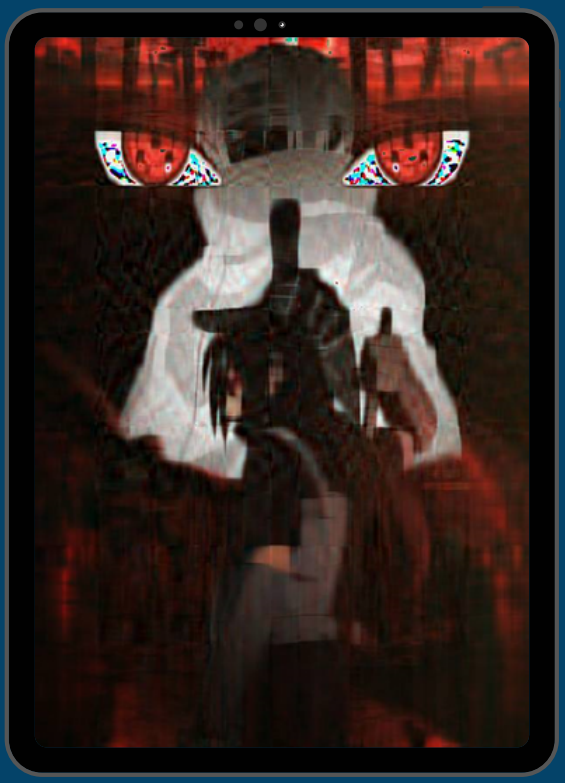
10 components



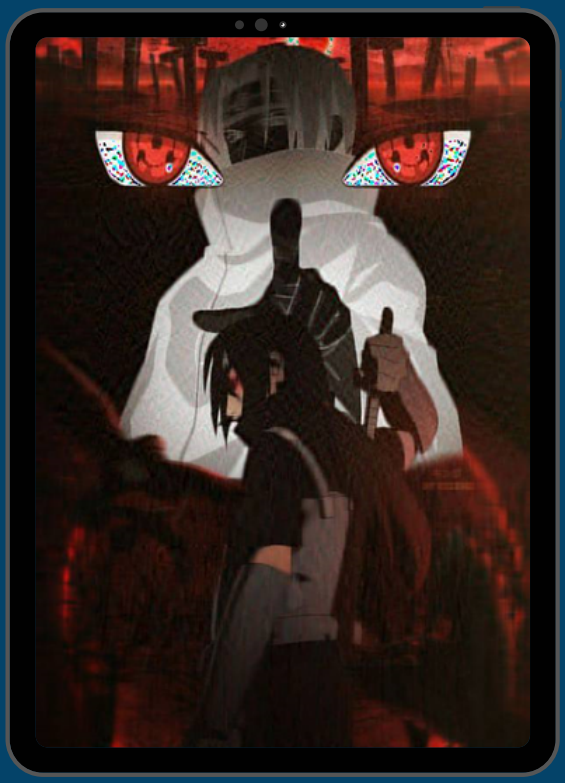
15 components



30 components

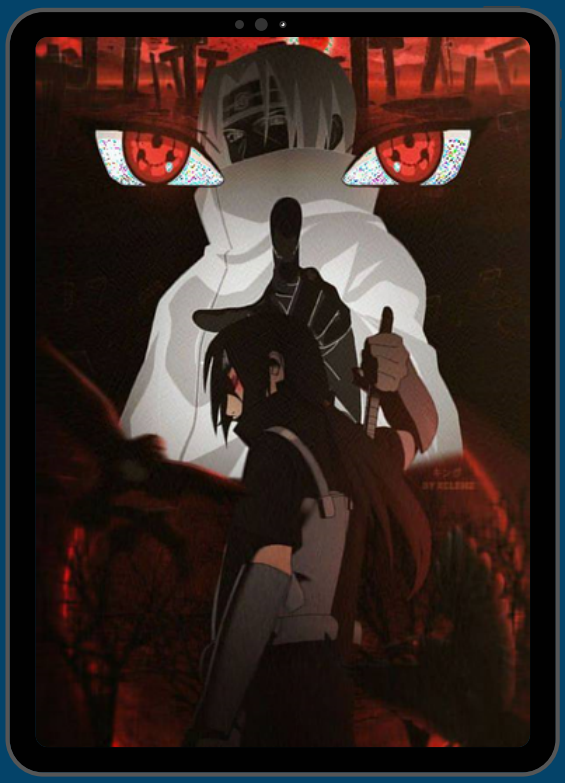


60 components



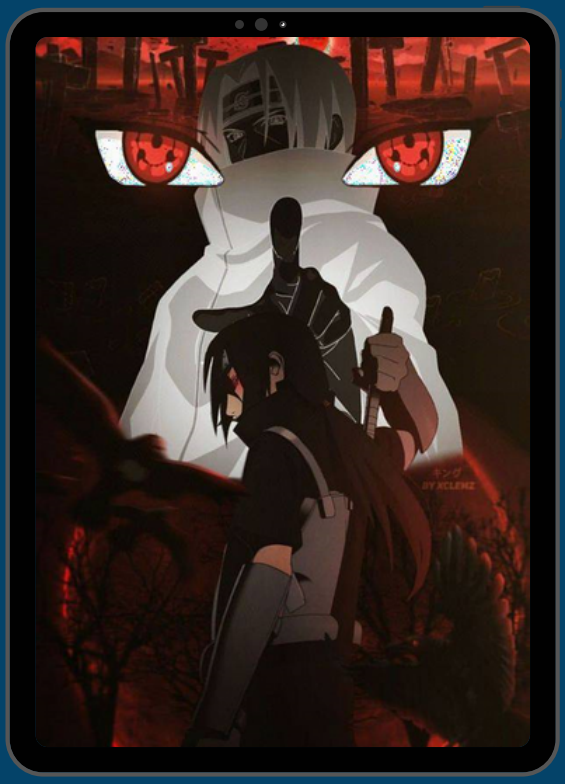
80 components

120 components



180 components

220 components



We will notice that the more principal components there are, the more similar the newly created image will be to the original image

This progressive improvement in quality is due to the fact that as more principal components are included in the compression,

the more the variance (information) is described in the output

```
library(knitr)

table <- matrix(0,9,3)
colnames(table) <- c("Number of components", "Image size (kilobytes)", "Saved Disk Space (kilobytes)")
table[,1] <- c(10,15,30,60,80,120,180,210,"Original Itatchi-Uchiha image")
table[9,2:3] <- round(c(file.info('C:/Users/Kimo Store/Downloads/Itatchi-Uchiha.jpg')$size/1024, 0),2)
for (i in c(1:8)) {
  path <- paste0('Compressed_image_with_',table[i,1], '_components.jpg')
  table[i,2] <- round(file.info(path)$size/1024,2)
  table[i,3] <- round(as.numeric(table[9,2]) - as.numeric(table[i,2]),2)
}

kable(table)
```

Now, let's compare the sizes in kilobytes to know if there is a significant change in the sizes of these images to ascertain our point that, PCA for image compression helps to save disk space without sacrificing quality since it only removes less significant components.

```
|Number of components          |Image size (kilobytes) |Saved Disk Space
|:-----------------------------|:----------------------|:---------------
|10                            |40.03                  |22.32
|15                            |44.21                  |18.14
|30                            |50.64                  |11.71
|60                            |55.61                  |6.74
|80                            |56.88                  |5.47
|120                           |58.13                  |4.22
|180                           |56.82                  |5.53
|210                           |56.24                  |6.11
|Original Itatchi-Uchiha image |62.35                  |0
> |
```

THANKS!!

شكراً