Project-high value client identification

R Project

methods to access The most common types methods are:

Maximum or complete linkage clustering: It computes all pairwise dissimilarities between the elements in cluster 1 and the elements in cluster 2, and considers the largest value (i.e., maximum value) of these dissimilarities as the distance between the two clusters. It tends to produce more compact clusters.

Minimum or single linkage clustering: It computes all pairwise dissimilarities between the elements in cluster 1 and the elements in cluster 2, and considers the smallest of these dissimilarities as a linkage criterion. It tends to produce long, “loose” clusters.

Mean or average linkage clustering: It computes all pairwise dissimilarities between the elements in cluster 1 and the elements in cluster 2, and considers the average of these dissimilarities as the distance between the two clusters.

Centroid linkage clustering: It computes the dissimilarity between the centroid for cluster 1 (a mean vector of length p variables) and the centroid for cluster 2.

Ward’s minimum variance method: It minimizes the total within-cluster variance. At each step the pair of clusters with minimum between-cluster distance are merged.

#========== Hierarchical clustering can be divided into two main types: Agglomerative (bottom-up) and divisive (top-down) Agglomerative clustering is good at identifying small clusters. Divisive hierarchical clustering is good at identifying large clusters here are multiple agglomeration methods to define clusters when performing a hierarchical cluster analysis; however, complete linkage and Ward’s method are often preferred for AGNES clustering.

For DIANA, clusters are divided based on the maximum average dissimilarity which is very similar to the mean or average linkage clustering method outlined above.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(DataExplorer)
library(ggplot2)
library(cluster)
library(factoextra)
```

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

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(purrr)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v tibble  3.1.4     v stringr 1.4.0
## v tidyr   1.1.3     v forcats 0.5.1
## v readr   2.0.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x gridExtra::combine() masks dplyr::combine()
## x dplyr::filter()      masks stats::filter()
## x dplyr::lag()         masks stats::lag()
```

```
data=read.csv('C:\\Users\\HP\\Desktop\\R Projects\\6\\Ecommerce.csv')
#head(data) #541909 obs. of 9 variables

str(data) #541909 obs. of 9 variables
```

```
## 'data.frame':    541909 obs. of  9 variables:
##  $ InvoiceNo  : chr  "536365" "536365" "536365" "536365" ...
##  $ StockCode  : chr  "85123A" "71053" "84406B" "84029G" ...
##  $ Description: chr  "WHITE HANGING HEART T-LIGHT HOLDER" "WHITE METAL LANTERN" "CREAM CUP
ID HEARTS COAT HANGER" "KNITTED UNION FLAG HOT WATER BOTTLE" ...
##  $ Quantity   : int  6 6 8 6 6 2 6 6 6 32 ...
##  $ InvoiceDate: chr  "29-Nov-16" "29-Nov-16" "29-Nov-16" "29-Nov-16" ...
##  $ UnitPrice  : num  2.55 3.39 2.75 3.39 3.39 7.65 4.25 1.85 1.85 1.69 ...
##  $ CustomerID : int  17850 17850 17850 17850 17850 17850 17850 17850 17850 13047 ...
##  $ Country    : chr  "United Kingdom" "United Kingdom" "United Kingdom" "United Kingdom"
...
##  $ X          : logi  NA NA NA NA NA NA ...
```

```
colSums(is.na(data))
```

```
##   InvoiceNo   StockCode Description    Quantity InvoiceDate   UnitPrice
##           0           0           0           0           0           0
##  CustomerID     Country           X
##      135080           0      541909
```

```
# Remove column 'X' which has all the rows with NA
data<-select(data,-c(X)) # 541909 obs. of 8 variables

# Remove rows of CustomerID which has NA data (removed 135080 rows)
data<-na.omit(data) #406829 obs. of 8 variables
head(data)
```

```
##   InvoiceNo StockCode                         Description Quantity InvoiceDate
## 1    536365    85123A  WHITE HANGING HEART T-LIGHT HOLDER        6   29-Nov-16
## 2    536365     71053                 WHITE METAL LANTERN        6   29-Nov-16
## 3    536365    84406B      CREAM CUPID HEARTS COAT HANGER        8   29-Nov-16
## 4    536365    84029G KNITTED UNION FLAG HOT WATER BOTTLE        6   29-Nov-16
## 5    536365    84029E      RED WOOLLY HOTTIE WHITE HEART.        6   29-Nov-16
## 6    536365     22752        SET 7 BABUSHKA NESTING BOXES        2   29-Nov-16
##   UnitPrice CustomerID        Country
## 1      2.55      17850 United Kingdom
## 2      3.39      17850 United Kingdom
## 3      2.75      17850 United Kingdom
## 4      3.39      17850 United Kingdom
## 5      3.39      17850 United Kingdom
## 6      7.65      17850 United Kingdom
```

```
data$InvoiceDate <- as.Date(data$InvoiceDate,"%d-%B-%y")
str(data)
```

```
## 'data.frame':    406829 obs. of  8 variables:
##  $ InvoiceNo  : chr  "536365" "536365" "536365" "536365" ...
##  $ StockCode  : chr  "85123A" "71053" "84406B" "84029G" ...
##  $ Description: chr  "WHITE HANGING HEART T-LIGHT HOLDER" "WHITE METAL LANTERN" "CREAM CUP
ID HEARTS COAT HANGER" "KNITTED UNION FLAG HOT WATER BOTTLE" ...
##  $ Quantity   : int  6 6 8 6 6 2 6 6 6 32 ...
##  $ InvoiceDate: Date, format: "2016-11-29" "2016-11-29" ...
##  $ UnitPrice  : num  2.55 3.39 2.75 3.39 3.39 7.65 4.25 1.85 1.85 1.69 ...
##  $ CustomerID : int  17850 17850 17850 17850 17850 17850 17850 17850 17850 13047 ...
##  $ Country    : chr  "United Kingdom" "United Kingdom" "United Kingdom" "United Kingdom"
...
##  - attr(*, "na.action")= 'omit' Named int [1:135080] 623 1444 1445 1446 1447 1448 1449 145
0 1451 1452 ...
##   ..- attr(*, "names")= chr [1:135080] "623" "1444" "1445" "1446" ...
```

```
# Computing the line total
data <- data %>% mutate(LineTotal = Quantity * UnitPrice) #406829 obs. of 9 variables
head(data)
```

```
##   InvoiceNo StockCode                          Description Quantity InvoiceDate
## 1    536365    85123A  WHITE HANGING HEART T-LIGHT HOLDER        6  2016-11-29
## 2    536365     71053                 WHITE METAL LANTERN        6  2016-11-29
## 3    536365    84406B      CREAM CUPID HEARTS COAT HANGER        8  2016-11-29
## 4    536365    84029G KNITTED UNION FLAG HOT WATER BOTTLE        6  2016-11-29
## 5    536365    84029E      RED WOOLLY HOTTIE WHITE HEART.        6  2016-11-29
## 6    536365     22752        SET 7 BABUSHKA NESTING BOXES        2  2016-11-29
##   UnitPrice CustomerID        Country LineTotal
## 1      2.55      17850 United Kingdom     15.30
## 2      3.39      17850 United Kingdom     20.34
## 3      2.75      17850 United Kingdom     22.00
## 4      3.39      17850 United Kingdom     20.34
## 5      3.39      17850 United Kingdom     20.34
## 6      7.65      17850 United Kingdom     15.30
```

```
# Country Summary
countrySummary <- data %>%
  group_by(Country) %>%
  summarise(revenue = sum(LineTotal), transactions = n_distinct(InvoiceNo)) %>%
  mutate(aveOrdVal = (round((revenue / transactions),2))) %>%
  ungroup() %>%
  arrange(desc(revenue))

head(countrySummary)
```

```
## # A tibble: 6 x 4
##   Country         revenue transactions aveOrdVal
##   <chr>             <dbl>        <int>     <dbl>
## 1 United Kingdom 6767873.        19857      341.
## 2 Netherlands     284662.          101     2818.
## 3 EIRE            250285.          319      785.
## 4 Germany         221698.          603      368.
## 5 France          196713.          458      430.
## 6 Australia       137077.           69     1987.
```

```
# Total revenue generated and Total Items purchased by each customer
customerData <- data %>%
  group_by(CustomerID, Country) %>%
  summarise(TotalRevenue = sum(LineTotal), TotalItemsSold = sum(Quantity))
```

```
## `summarise()` has grouped output by 'CustomerID'. You can override using the `.groups` arg
ument.
```

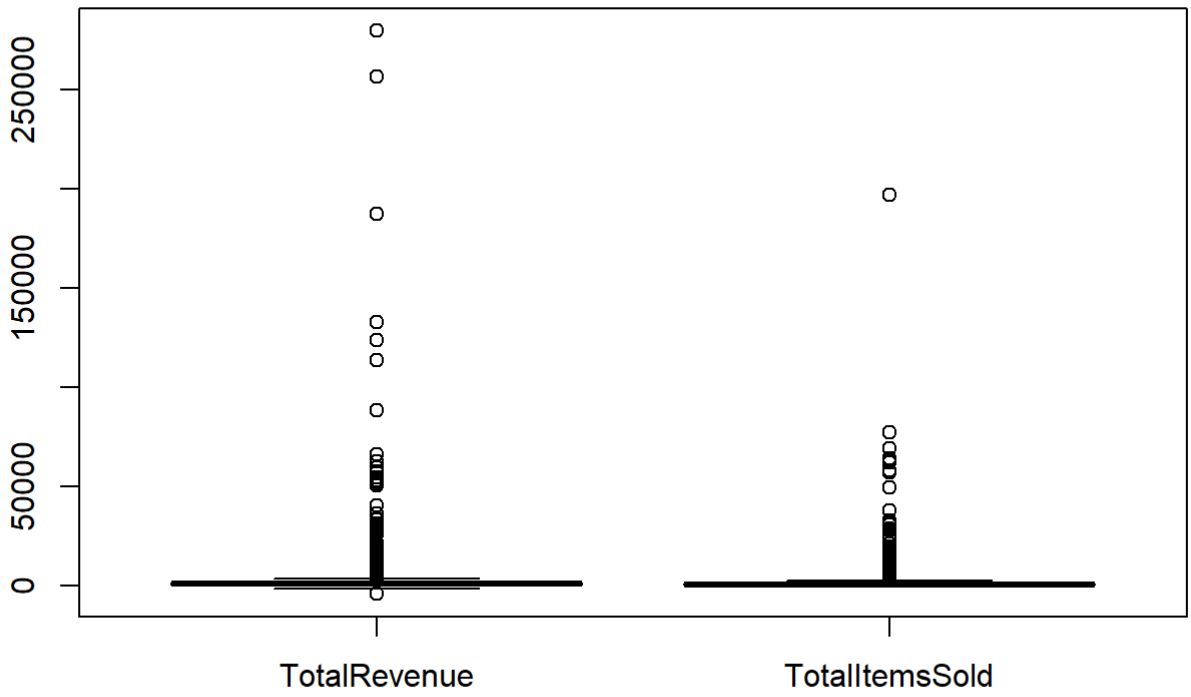
```
head(customerData)
```

```
## # A tibble: 6 x 4
## # Groups:   CustomerID [6]
##   CustomerID Country        TotalRevenue TotalItemsSold
##        <int> <chr>                 <dbl>          <int>
## 1      12346 United Kingdom           0              0
## 2      12347 Iceland               4310           2458
## 3      12348 Finland               1797.          2341
## 4      12349 Italy                 1758.           631
## 5      12350 Norway                 334.           197
## 6      12352 Norway                1545.           470
```

```
# Few customers are residents of two countries in the same year.
# Such data contributes to very less percentage of data and can be removed to ensure correct
 functionality for working with data.
# removed duplicate data #4364 obs, 4 variables
n_occur <- data.frame(table(customerData$CustomerID))
single_ResidentsIds = (n_occur[n_occur$Freq == 1,])$Var
customerData <- subset(customerData, (customerData$CustomerID %in% single_ResidentsIds))
remove(n_occur)
remove(single_ResidentsIds)


set.seed(123) # To ensure the same result every time

# Categorical variable - Country
length(unique(data$Country))
```

```
## [1] 37
```

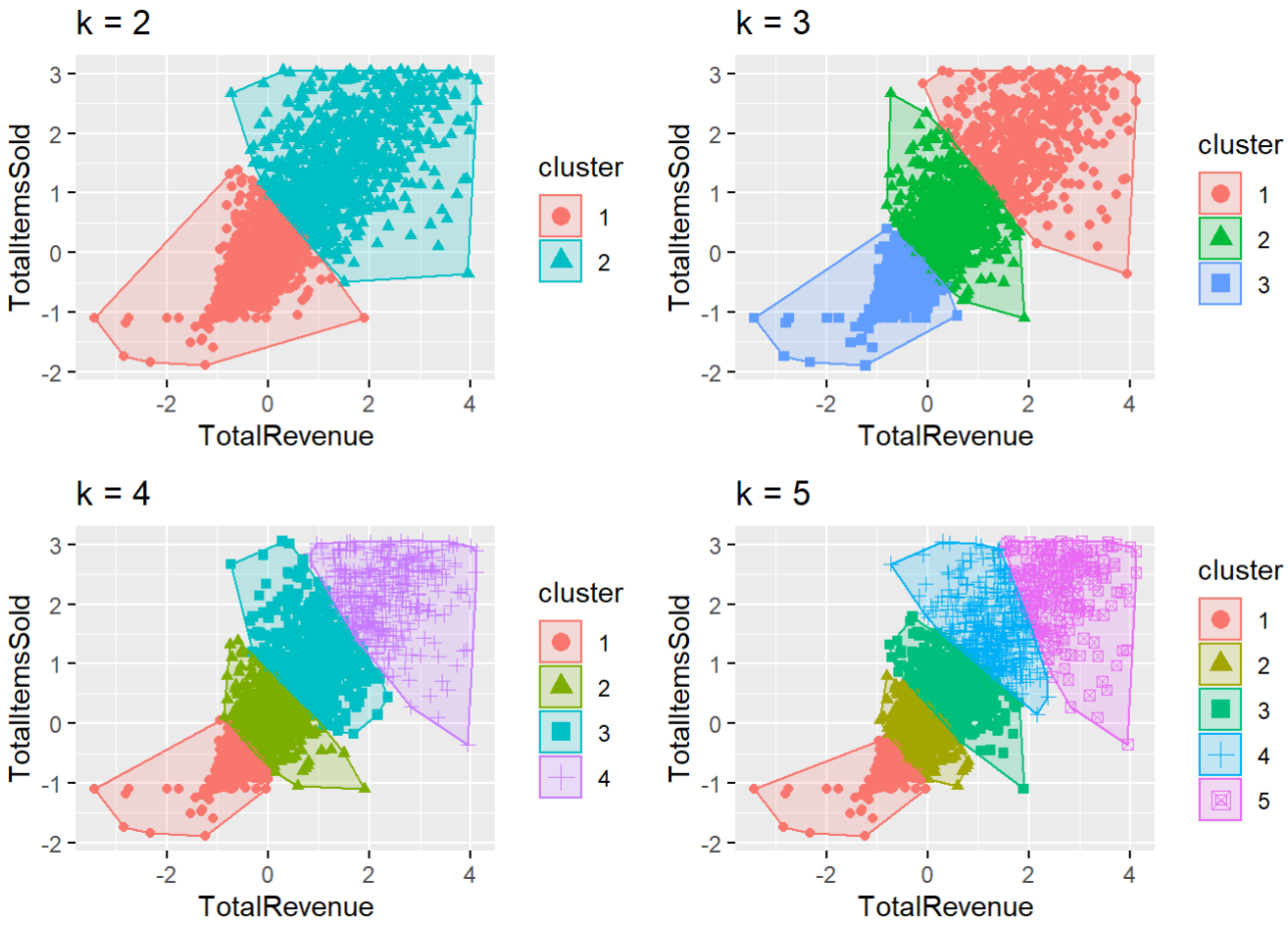
```
# Removing CustomerId and Country Columns
customerData <- customerData[-c(1:2)]

# Visualizing outliers
boxplot(customerData)
```




```
#boxplot(customerData,horizontal=TRUE)$out

# Eliminating outliers-1-TotalRevenue
iqr <- IQR(customerData$TotalRevenue)
Q <- quantile(customerData$TotalRevenue, probs=c(.25, .75), na.rm = FALSE)
eliminated <- subset(customerData, customerData$TotalRevenue > (Q[1] - 1.5*iqr) & customerDat
a$TotalRevenue < (Q[2]+1.5*iqr))

# Eliminating outliers-2-TotalRevenue
iqr <- IQR(eliminated$TotalItemsSold)
Q <- quantile(eliminated$TotalItemsSold, probs=c(.25, .75), na.rm = FALSE)
customerData <- subset(eliminated, eliminated$TotalItemsSold > (Q[1] - 1.5*iqr) & eliminated
$TotalItemsSold < (Q[2]+1.5*iqr))
remove(eliminated)

#=================================================================
# Scaling the data before applying the clustering algorithms
customerData <- scale(customerData)
head(customerData)
```

```
##      TotalRevenue TotalItemsSold
## [1,]   -1.0914952     -1.1012271
## [2,]    1.4921069      0.5515513
## [3,]   -0.5999265     -0.5852250
## [4,]    1.1802607      0.1298440
## [5,]   -0.9606651     -1.0488411
## [6,]    0.4952250      0.2870020
```

```
#=========================================================
# K-mean clustering algorithm
#=========================================================
set.seed(123)


# Grouping the data in to 3 clusters using k-means
k2 <- kmeans(customerData, centers = 2, nstart = 25)
k3 <- kmeans(customerData, centers = 3, nstart = 25)
k4 <- kmeans(customerData, centers = 4, nstart = 25)
k5 <- kmeans(customerData, centers = 5, nstart = 25)

#str(k3)
#k3$size

# plots to compare
p1 <- fviz_cluster(k2, geom = "point",  data = customerData) + ggtitle("k = 2")
p2 <- fviz_cluster(k3, geom = "point",  data = customerData) + ggtitle("k = 3")
p3 <- fviz_cluster(k4, geom = "point",  data = customerData) + ggtitle("k = 4")
p4 <- fviz_cluster(k5, geom = "point",  data = customerData) + ggtitle("k = 5")

grid.arrange(p1, p2, p3, p4, nrow = 2)
```



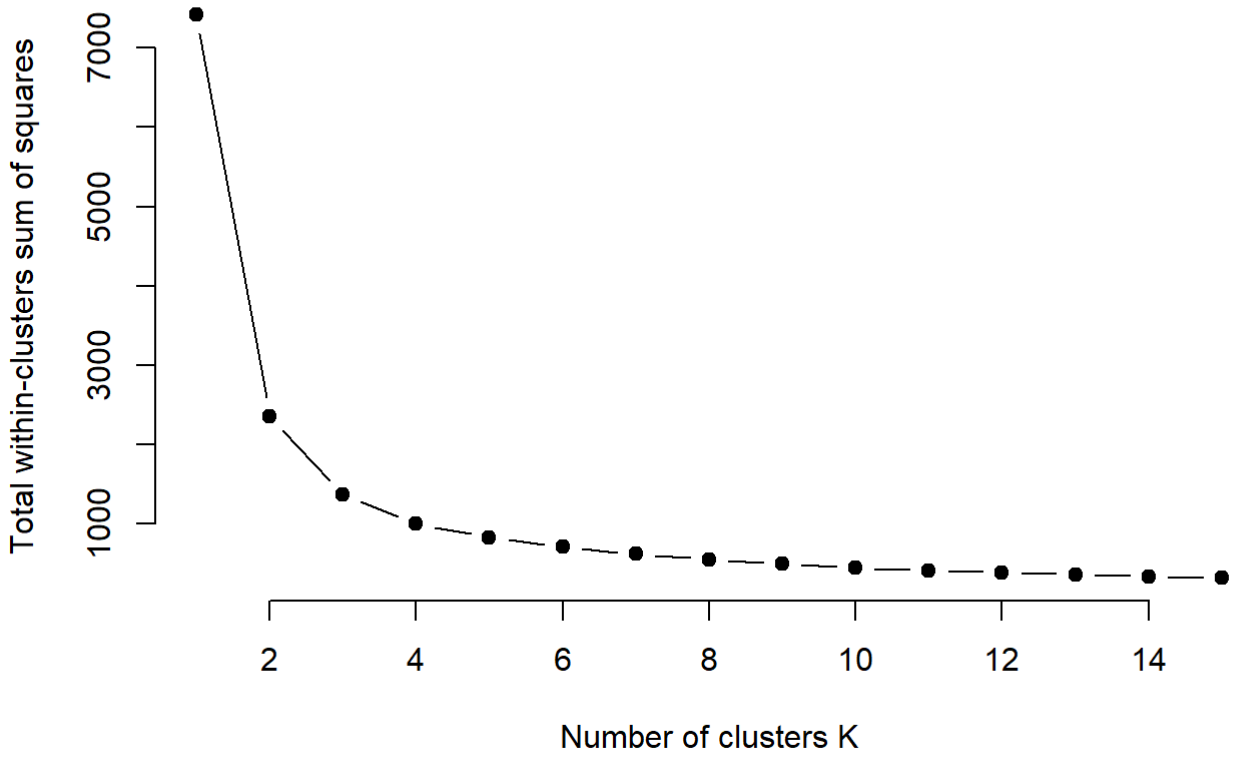
```
#====================================================================

#====================================================================

# Determining optimal number of clusters using Elbow Method
set.seed(123)

# function to compute total within-cluster sum of square
wss <- function(k) {
  kmeans(customerData, k, nstart = 10 )$tot.withinss
}

# Compute and plot wss for k = 1 to k = 15
k.values <- 1:15

# extract wss for 2-15 clusters
wss_values <- map_dbl(k.values, wss)
```

```
## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations
```

```
print(wss_values)
```

```
##  [1] 7414.0000 2350.9065 1368.3985 1000.5195  827.8971  713.5113  619.9442
##  [8]  551.8390  498.2309  452.0838  415.9363  389.2401  363.4926  339.8110
## [15]  320.4122
```

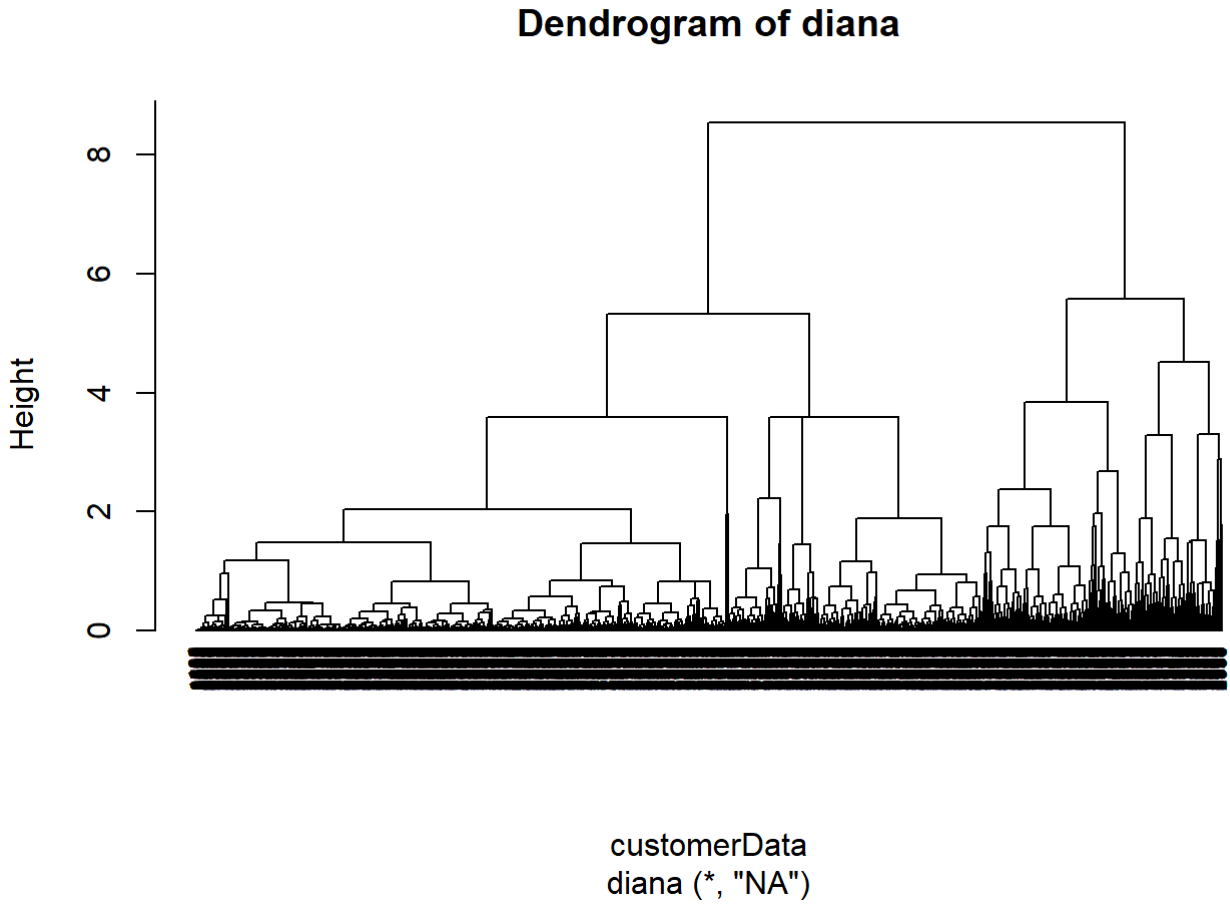
```
plot(k.values, wss_values,
     type="b", pch = 19, frame = FALSE,
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")
```




```
#============================================================

#============================================================
## DIVISIVE HIERARCHICAL CLUSTERING

# compute divisive hierarchical clustering
hc2 <- diana(customerData)

# Divise coefficient; amount of clustering structure found
hc2$dc
```

```
## [1] 0.9960423
```

```
# plot dendrogram
pltree(hc2, cex = 0.6, hang = -1, main = "Dendrogram of diana")
```




```
#============================================================
m <- c( "average", "single", "complete", "ward")
names(m) <- c( "average", "single", "complete", "ward")
d <- dist(customerData, method = "euclidean")
# function to compute coefficient
ac <- function(x) {
  agnes(customerData, method = x)$ac
}

map_dbl(m, ac)
```

```
##   average    single  complete      ward
## 0.9912533 0.9671068 0.9965356 0.9997057
```

```
# Ward's method
hc3 <- hclust(d, method = "ward.D2" )

# Cut tree into 3 groups
sub_grp <- cutree(hc3, k = 3)
str(sub_grp)
```

```
##  int [1:3708] 1 2 1 3 1 3 3 2 3 2 ...
```

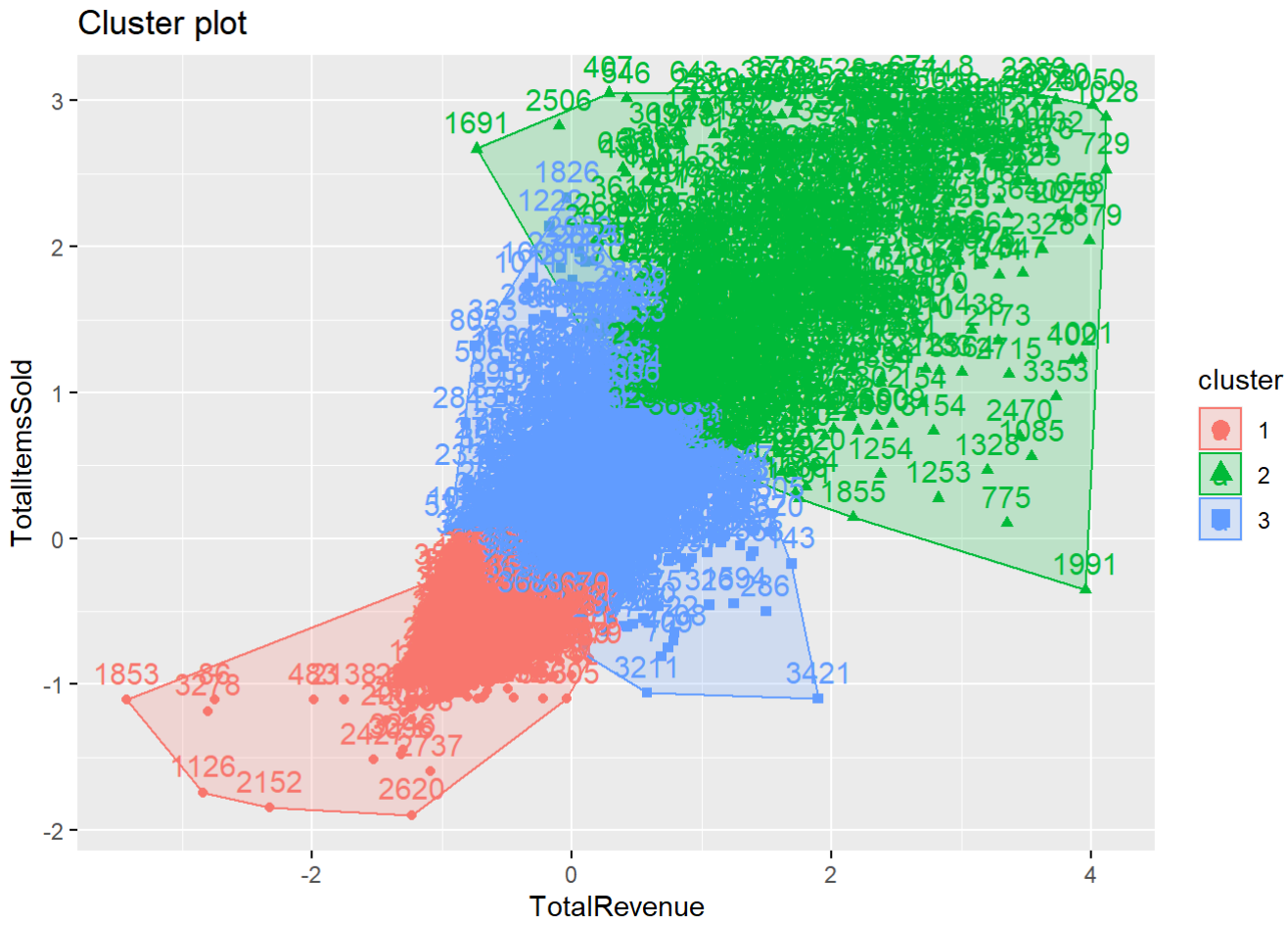
```
head(sub_grp)
```

```
## [1] 1 2 1 3 1 3
```

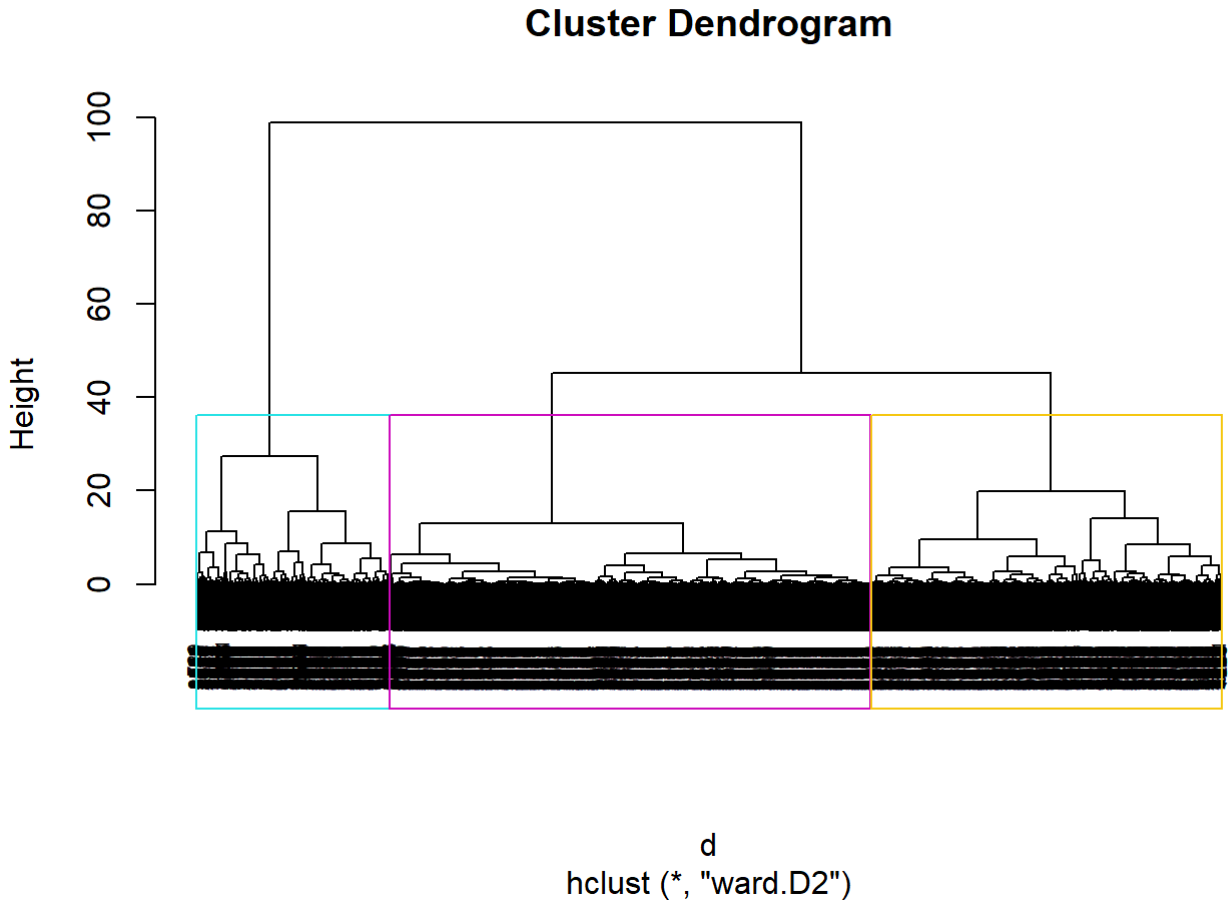
```
table(sub_grp)
```

```
## sub_grp
##    1    2    3
## 1739  700 1269
```

```
fviz_cluster(list(data = customerData, cluster = sub_grp))
```




```
# Dendogram with border around 3 clusters
plot(hc3, cex = 0.6)
rect.hclust(hc3, k = 3, border = 5:10)
```


Cluster Dendrogram
Height
0
20
40
60
80
100
d
hclust (*, "ward.D2")

```r
#============================================================

# Determining Optimal clusters in Hierarchical Clustering using Elbow method
# fviz_nbclust(customerData, FUN = hcut, method = "wss")

# Compute WSS for given k value -> Hierarchical Clustering

wssForHierarchical <- function(k){
  sub_grp <- cutree(hc3, k)
  df=as.data.frame(customerData)

  df2=df %>%
    mutate(cluster = sub_grp)%>%
    group_by(cluster)%>%
    summarize(TR=mean(TotalRevenue), TS=mean(TotalItemsSold))

  df1=df %>%
    mutate(cluster = sub_grp)

  df3=left_join(df1,df2,by="cluster")

  D=(df3$TotalRevenue-df3$TR)^2+(df3$TotalItemsSold-df3$TS)^2

  df4=cbind(df3,D)
  WSS=sum(df4$D)
  WSS
}

# Calculating WSS at hcut=3
wssForHierarchical(3)
```

```
## [1] 1522.435
```