Assignment 4 - Fundamentals of Machine Learning

2022-10-26

```
library(ISLR)
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(class)
library(psych)
library(caret)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

```
## Loading required package: lattice
```

```
library(FNN)
```

```
##
## Attaching package: 'FNN'
```

```
## The following objects are masked from 'package:class':
##
##     knn, knn.cv
```

```
library(melt)
library(MASS)
library(reshape2)
library(reshape)
```

```
##
## Attaching package: 'reshape'
```

```
## The following objects are masked from 'package:reshape2':
##
##     colsplit, melt, recast

## The following object is masked from 'package:class':
##
##     condense
```

```
library(e1071)
library(stats)
library (dplyr)
```

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

## The following object is masked from 'package:reshape':
##
##     rename

## The following object is masked from 'package:MASS':
##
##     select

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

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

```
library(ggplot2)
library(ggfortify)
```

```
## Registered S3 methods overwritten by 'ggfortify':
##   method                 from
##   autoplot.Arima         forecast
##   autoplot.acf           forecast
##   autoplot.ar            forecast
##   autoplot.bats          forecast
##   autoplot.decomposed.ts forecast
##   autoplot.ets           forecast
##   autoplot.forecast      forecast
##   autoplot.stl           forecast
##   autoplot.ts            forecast
##   fitted.ar              forecast
##   fortify.ts             forecast
##   residuals.ar           forecast
```

```r
library(factoextra)
```

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

```r
library(NbClust)
library(flexclust)
```

```
## Loading required package: grid
```

```
## Loading required package: modeltools
```

```
## Loading required package: stats4
```

```
##
## Attaching package: 'flexclust'
```

```
## The following object is masked from 'package:e1071':
##
##     bclust
```

```r
library(dendextend)
```

```
##
## ---------------------
## Welcome to dendextend version 1.16.0
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
```

```
##
## Attaching package: 'dendextend'
```

```
## The following object is masked from 'package:stats':
##
##     cutree
```

A.

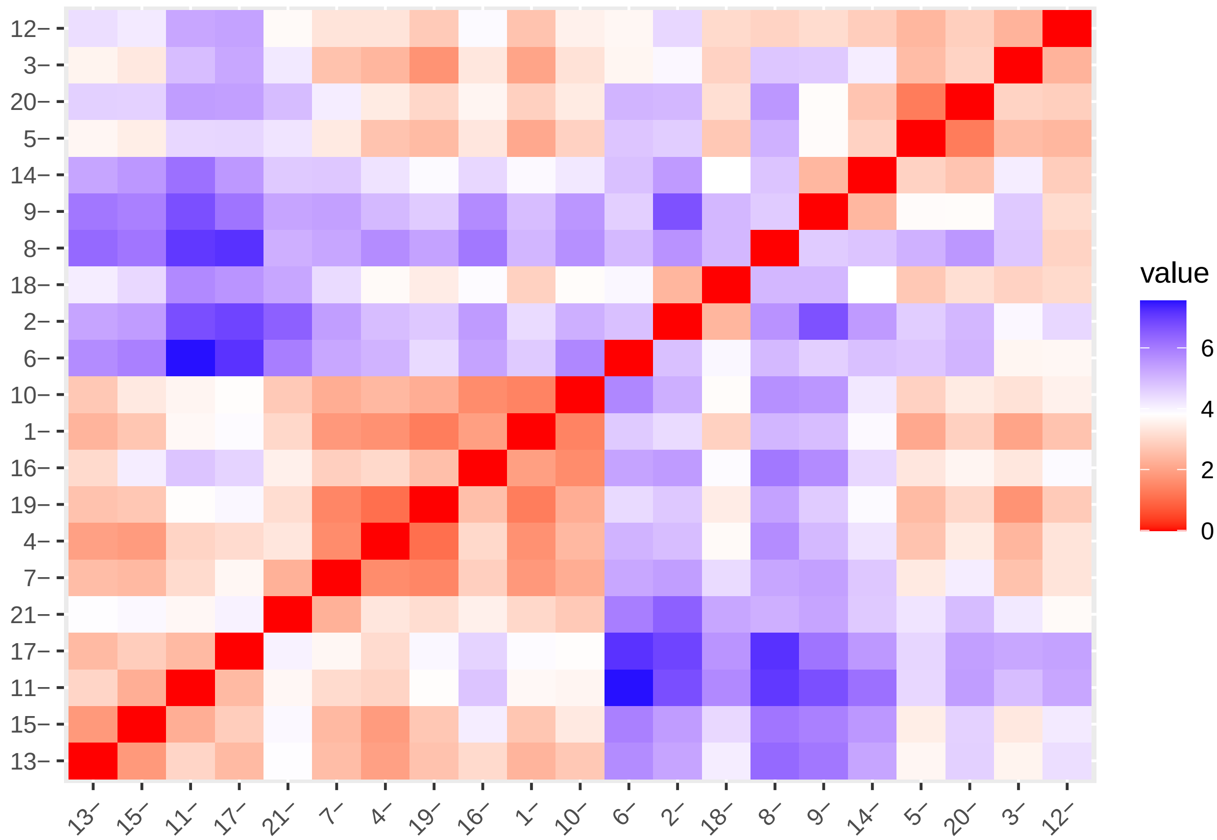
```r
rm(list = ls())
set.seed(1)
pharmaceutical_data <- read.csv("Pharmaceuticals.csv", header = TRUE)
row.names(pharmaceutical_data) <- pharmaceutical_data[,2]
pharmaceutical_data <- pharmaceutical_data[, -c(1,2,12,13,14)]
pharmaceutical_data.norm <- sapply(pharmaceutical_data, scale)

pharmaceutical_data.norm
```

```
##        Market_Cap        Beta    PE_Ratio         ROE        ROA Asset_Turnover
##  [1,]  0.1840960 -0.80125356 -0.04671323  0.04009035  0.2416121      0.0000000
##  [2,] -0.8544181 -0.45070513  3.49706911 -0.85483986 -0.9422871      0.9225312
##  [3,] -0.8762600 -0.25595600 -0.29195768 -0.72225761 -0.5100700      0.9225312
##  [4,]  0.1702742 -0.02225704 -0.24290879  0.10638147  0.9181259      0.9225312
##  [5,] -0.1790256 -0.80125356 -0.32874435 -0.26484883 -0.5664461     -0.4612656
##  [6,] -0.6953818  2.27578267  0.14948233 -1.45146000 -1.7127612     -0.4612656
##  [7,] -0.1078688 -0.10015669 -0.70887325  0.59693581  0.8617498      0.9225312
##  [8,] -0.9767669  1.26308721  0.03299122 -0.11237924 -1.1677918     -0.4612656
##  [9,] -0.9704532  2.15893320 -1.34037772 -0.70899938 -1.0174553     -1.8450624
## [10,]  0.2762415 -1.34655112  0.14948233  0.34502953  0.5610770     -0.4612656
## [11,]  1.0999201 -0.68440408 -0.45749769  2.45971647  1.8389364      1.3837968
## [12,] -0.9393967  0.48409069 -0.34100657 -0.29136529 -0.6979905     -0.4612656
## [13,]  1.9841758 -0.25595600  0.18013789  0.18593083  1.0872544      0.9225312
## [14,] -0.9632863  0.87358895  0.19240011 -0.96753478 -0.9610792     -1.8450624
## [15,]  1.2782387 -0.25595600 -0.40231769  0.98142435  0.8429577      1.8450624
## [16,]  0.6654710 -1.30760129 -0.23677768 -0.52338423  0.1288598     -0.9225312
## [17,]  2.4199899  0.48409069 -0.11415545  1.31287998  1.6322239      0.4612656
## [18,] -0.0240846 -0.48965495  1.90298017 -0.81506519 -0.9047030     -0.4612656
## [19,] -0.4018812 -0.06120687 -0.40231769 -0.21181593  0.5234929      0.4612656
## [20,] -0.9281345 -1.11285216 -0.43297324 -1.03382590 -0.6979905     -0.9225312
## [21,] -0.1614497  0.40619104 -0.75792214  1.92938746  0.5422849     -0.4612656
##          Leverage  Rev_Growth Net_Profit_Margin
##  [1,] -0.21209793 -0.52776752        0.06168225
##  [2,]  0.01828430 -0.38113909       -1.55366706
##  [3,] -0.40408312 -0.57211809       -0.68503583
##  [4,] -0.74965647  0.14744734        0.35122600
##  [5,] -0.31449003  1.21638667       -0.42597037
##  [6,] -0.74965647 -1.49714434       -1.99560225
##  [7,] -0.02011273 -0.96584257        0.74744375
##  [8,]  3.74279705 -0.63276071       -1.24888417
##  [9,]  0.61983791  1.88617085       -0.36501379
## [10,] -0.07130879 -0.64814764        1.17413980
## [11,] -0.31449003  0.76926048        0.82363947
## [12,]  1.10620040  0.05603085       -0.71551412
## [13,] -0.62166634 -0.36213170        0.33598685
## [14,]  0.44065173  1.53860717        0.85411776
## [15,] -0.39128411  0.36014907       -0.24310064
## [16,] -0.67286239 -1.45369888        1.02174835
## [17,] -0.54487226  1.10143723        1.44844440
## [18,] -0.30169102  0.14744734       -1.27936246
## [19,] -0.74965647 -0.43544591        0.29026942
## [20,] -0.49367621  1.43089863       -0.09070919
## [21,]  0.68383297 -1.17763919        1.49416183
```

```
pharmaceutical_data <- pharmaceutical_data.norm
```

B.

```
pharmaceutical_data <- scale(pharmaceutical_data)
distance <- get_dist(pharmaceutical_data)
fviz_dist(distance)
```




```
k4 <- kmeans (pharmaceutical_data, centers = 4, nstart = 25)
k4$centers
```

```
##    Market_Cap       Beta   PE_Ratio        ROE        ROA Asset_Turnover
## 1  1.69558112 -0.1780563 -0.1984582  1.2349879  1.3503431   1.153164e+00
## 2 -0.03142211 -0.4360989 -0.3172485  0.1950459  0.4083915   1.729746e-01
## 3 -0.82617719  0.4775991 -0.3696184 -0.5631589 -0.8514589  -9.994088e-01
## 4 -0.52462814  0.4451409  1.8498439 -1.0404550 -1.1865838  -7.401487e-17
##     Leverage Rev_Growth Net_Profit_Margin
## 1 -0.4680782  0.4671788         0.5912425
## 2 -0.2744931 -0.7041516         0.5569544
## 3  0.8502201  0.9158889        -0.3319956
## 4 -0.3443544 -0.5769454        -1.6095439
```

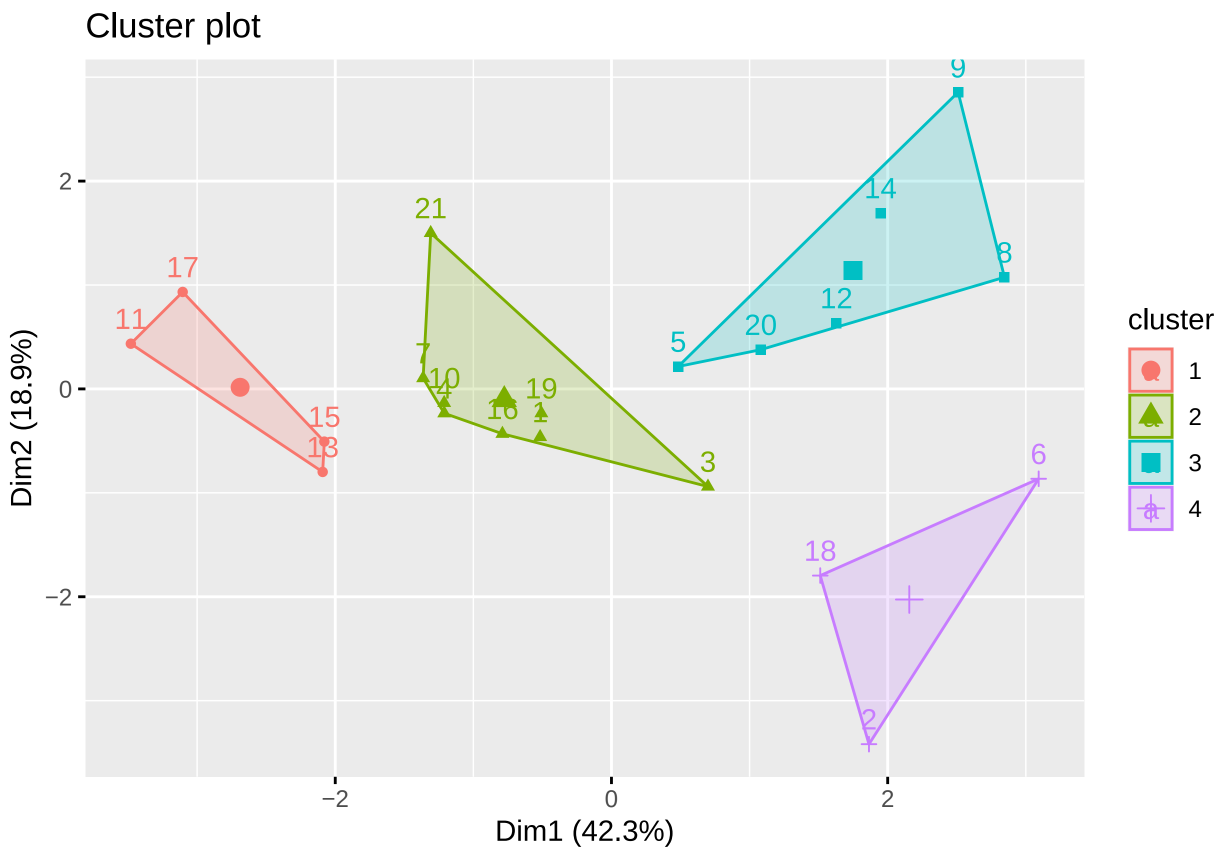
```
k4$size
```

```
## [1] 4 8 6 3
```

```
k4$cluster[120]
```

```
## [1] NA
```

```
fviz_cluster(k4, data = pharmaceutical_data)
```



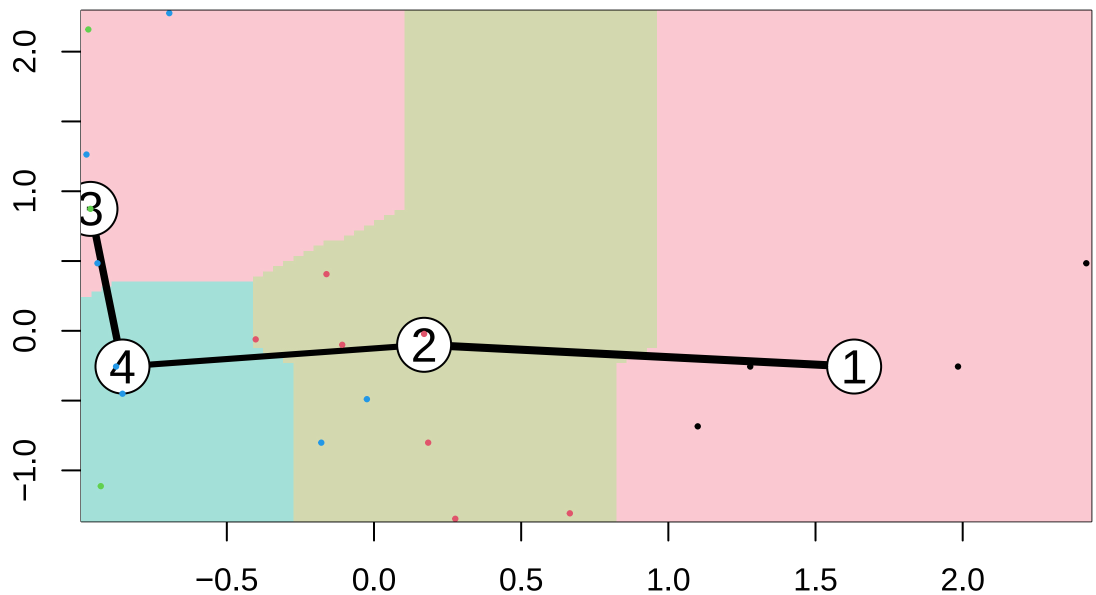

```
set.seed(123)
k4 = kcca(pharmaceutical_data, k=4, kccaFamily("kmedians"))
k4
```

```
## kcca object of family 'kmedians'
##
## call:
## kcca(x = pharmaceutical_data, k = 4, family = kccaFamily("kmedians"))
##
## cluster sizes:
##
## 1 2 3 4
## 4 7 3 7
```

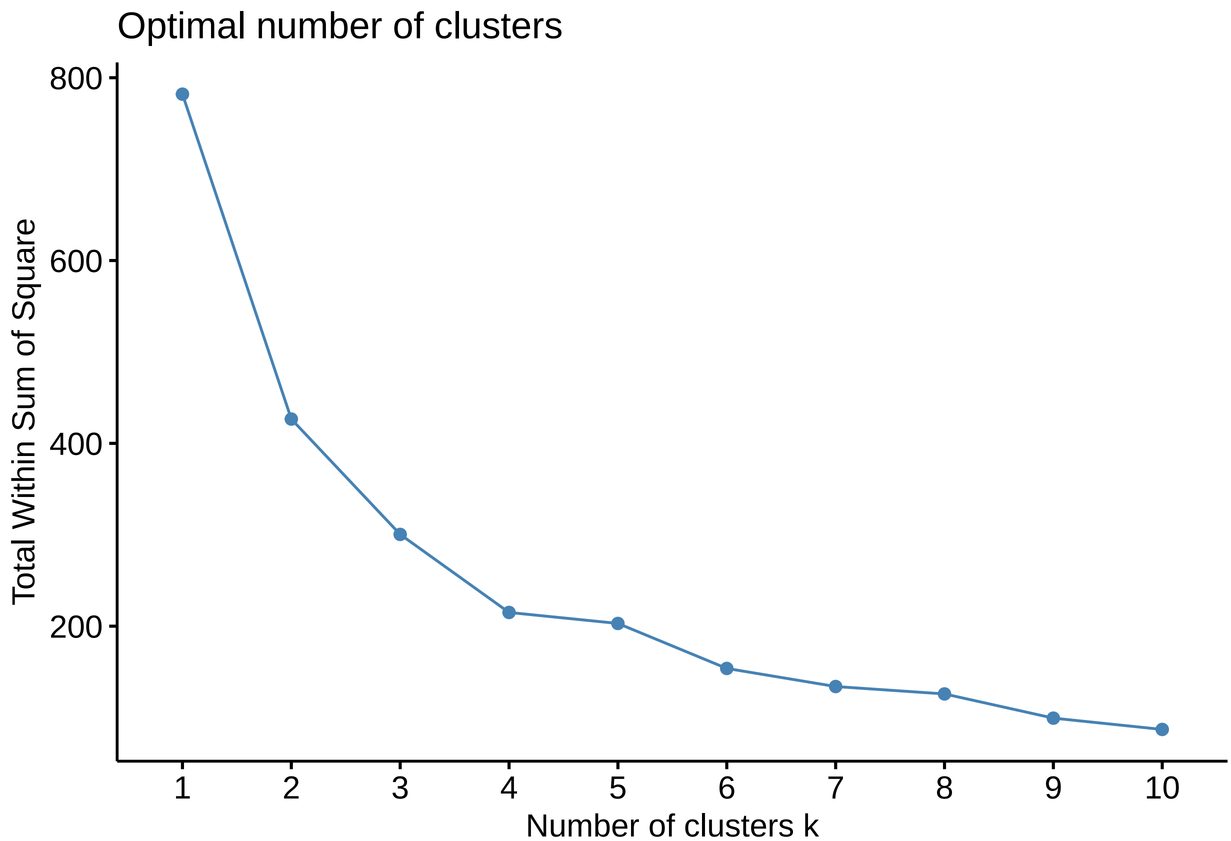
```r
clusters_index <- predict(k4)
dist(k4@centers)
```

```
##          1        2        3
## 2 2.608581
## 3 5.395288 3.872647
## 4 4.664586 2.864999 3.010141
```

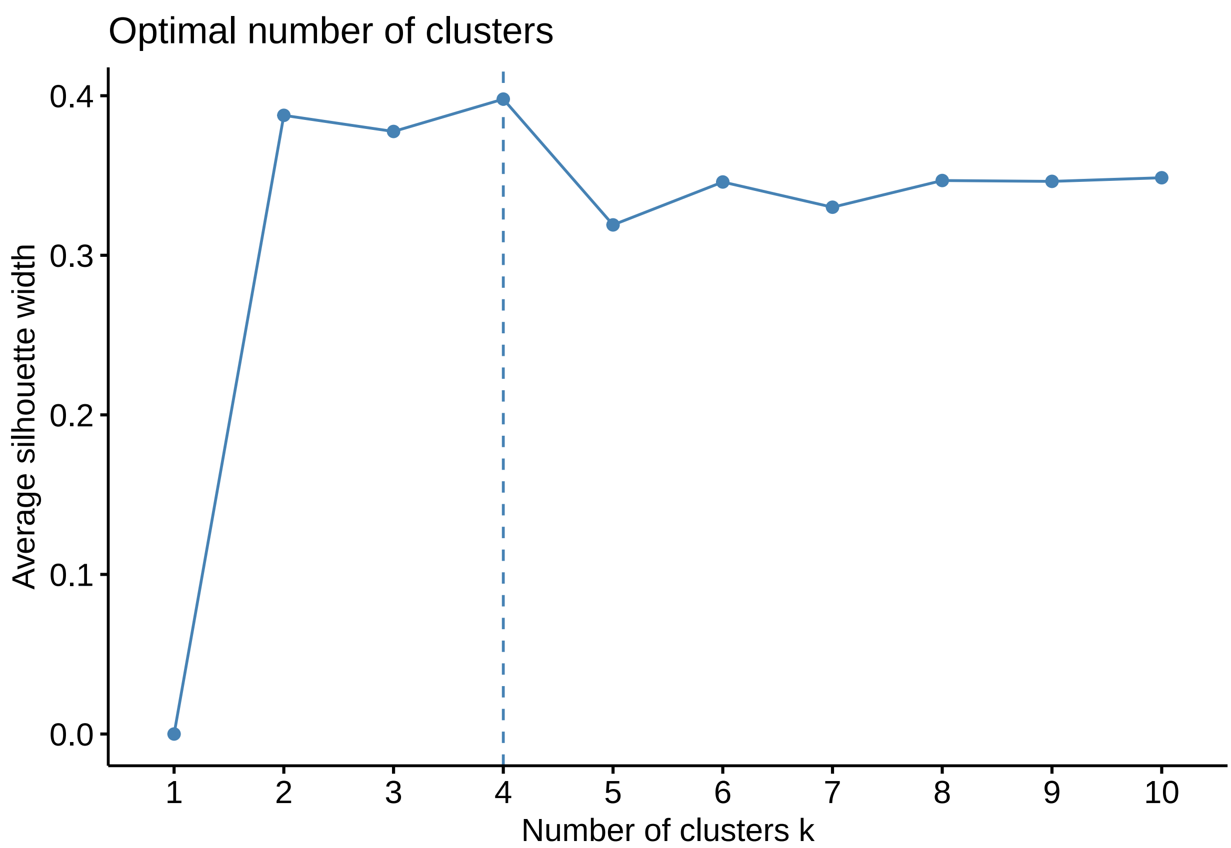
```r
image(k4)
points(pharmaceutical_data, col= clusters_index, pch = 19, cex=.3)
```




```r
set.seed(123)
pharmaceutical_data <- Auto[,c(1,6)]
pharmaceutical_data <- scale (pharmaceutical_data)
fviz_nbclust(pharmaceutical_data, kmeans, method = "wss")
```

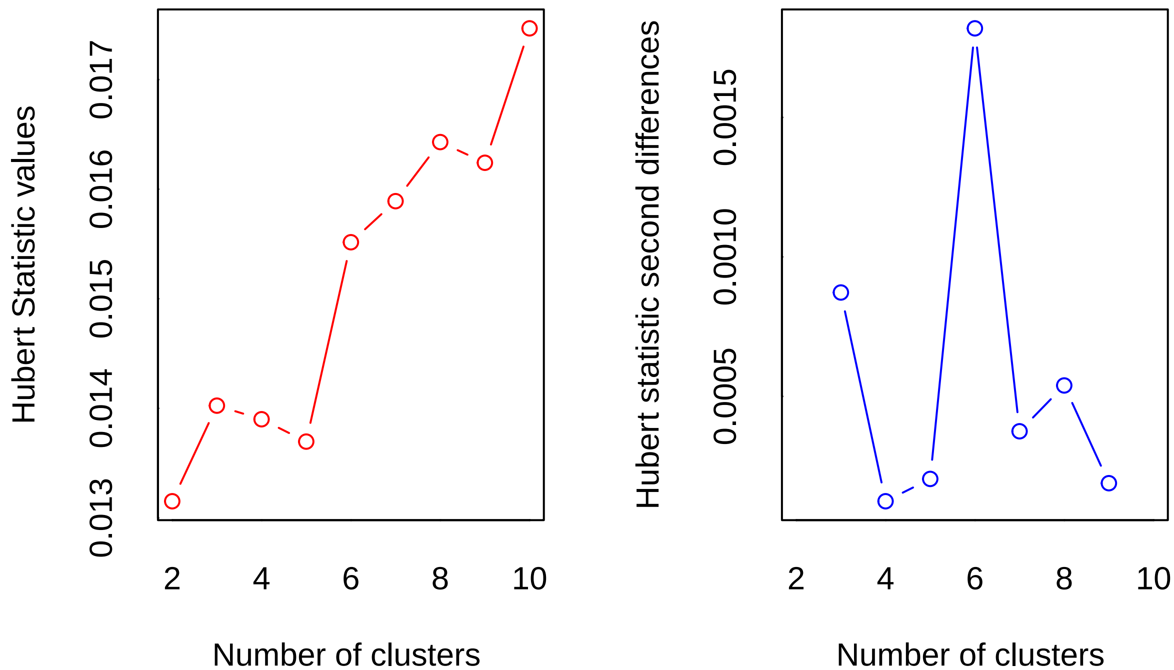



```
fviz_nbclust (pharmaceutical_data, kmeans, method = "silhouette")
```

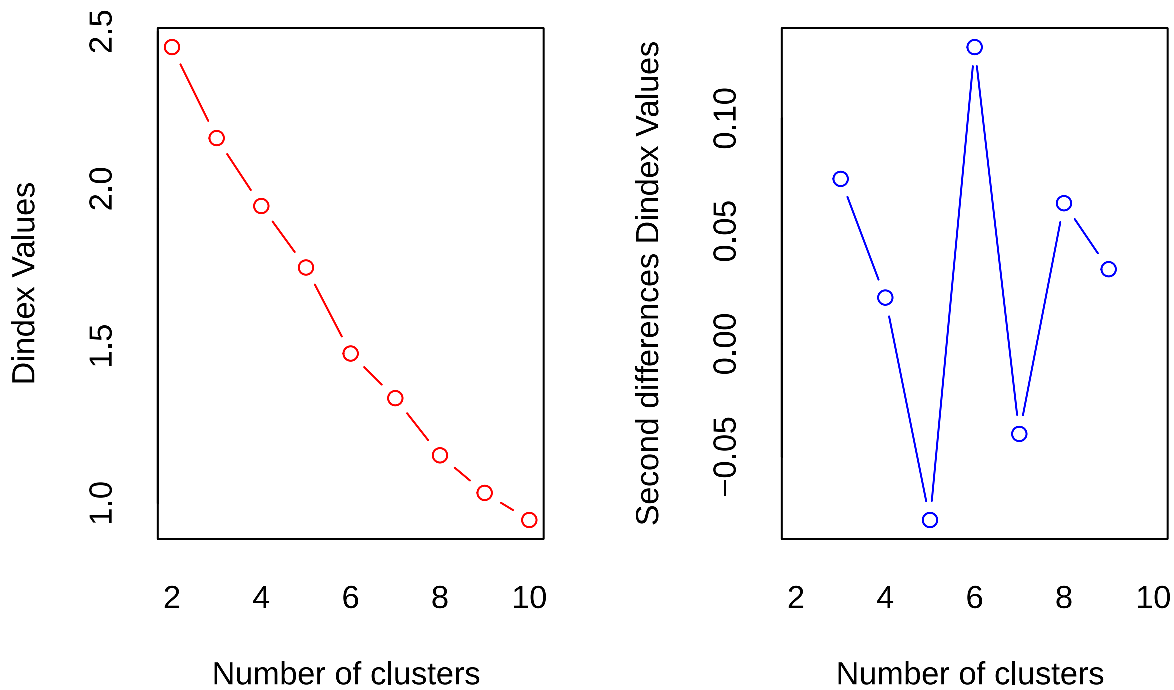



```
nc <- NbClust(pharmaceutical_data.norm, distance="euclidean", min.nc=2, max.nc=10, method="average")
```

```
## Warning in pf(beale, pp, df2): NaNs produced
```




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



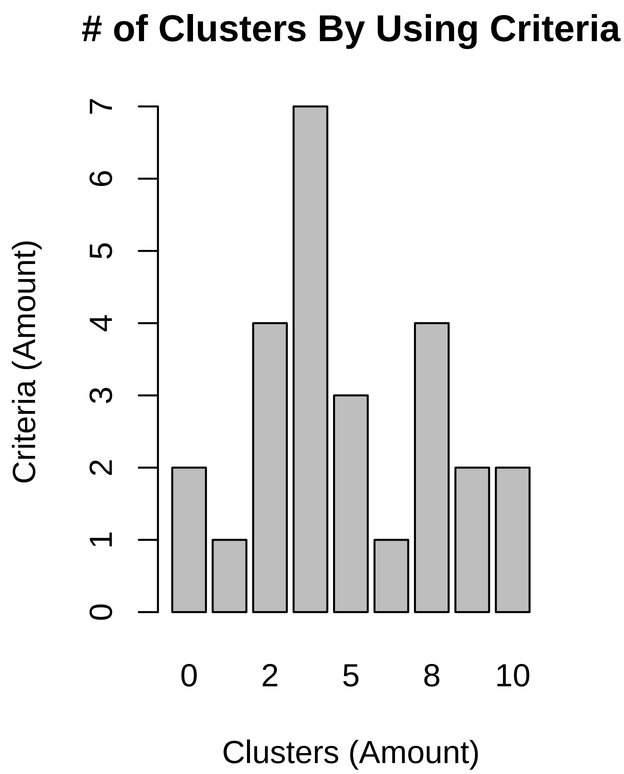

```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 4 proposed 2 as the best number of clusters
## * 7 proposed 3 as the best number of clusters
## * 3 proposed 5 as the best number of clusters
## * 1 proposed 6 as the best number of clusters
## * 4 proposed 8 as the best number of clusters
## * 2 proposed 9 as the best number of clusters
## * 2 proposed 10 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *******************************************************************
```

```
table(nc$Best.n[1,])
```

```
##
```

```
##  0  1  2  3  5  6  8  9 10
##  2  1  4  7  3  1  4  2  2
```

```
barplot(table(nc$Best.n[1,]), xlab="Clusters (Amount)", ylab="Criteria (Amount)", main="# of Clusters By
d <- dist(pharmaceutical_data.norm)
```

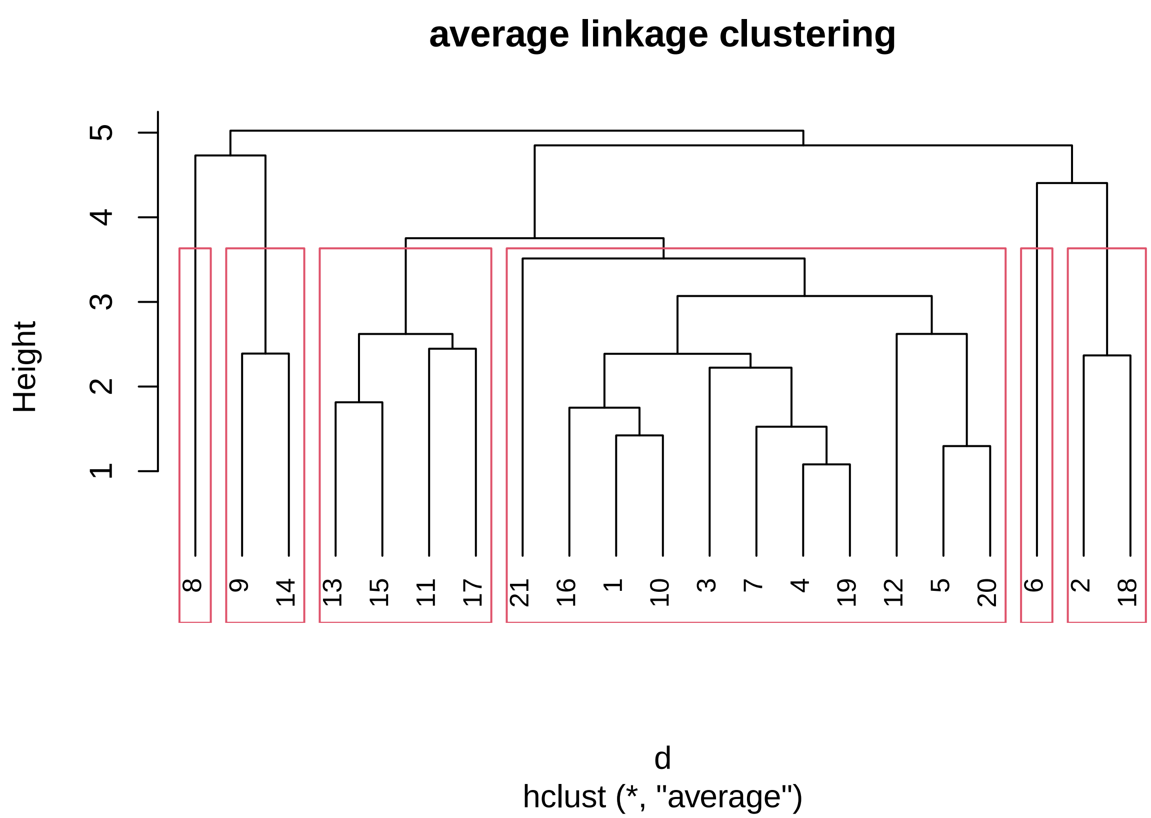



```
AverageClust <- hclust(d, method="average")

plot(AverageClust, hang = -1, cex=0.8, main="average linkage clustering")

clusters <- cutree(AverageClust, k=6)

aggregate(pharmaceutical_data.norm, by=list(cluster=clusters), median)
```

```
##   cluster Market_Cap      Beta    PE_Ratio        ROE        ROA Asset_Turnover
## 1       1 -0.1614497 -0.255956 -0.32874435 -0.2118159  0.2416121     -0.4612656
## 2       2 -0.4392513 -0.470180  2.70002464 -0.8349525 -0.9234951      0.2306328
## 3       3 -0.6953818  2.275783  0.14948233 -1.4514600 -1.7127612     -0.4612656
## 4       4 -0.9767669  1.263087  0.03299122 -0.1123792 -1.1677918     -0.4612656
## 5       5 -0.9668697  1.516261 -0.57398880 -0.8382671 -0.9892673     -1.8450624
## 6       6  1.6312072 -0.255956 -0.25823657  1.1471522  1.3597391      1.1531640
##     Leverage Rev_Growth Net_Profit_Margin
## 1 -0.3144900 -0.5277675         0.2902694
## 2 -0.1417034 -0.1168459        -1.4165148
## 3 -0.7496565 -1.4971443        -1.9956023
```

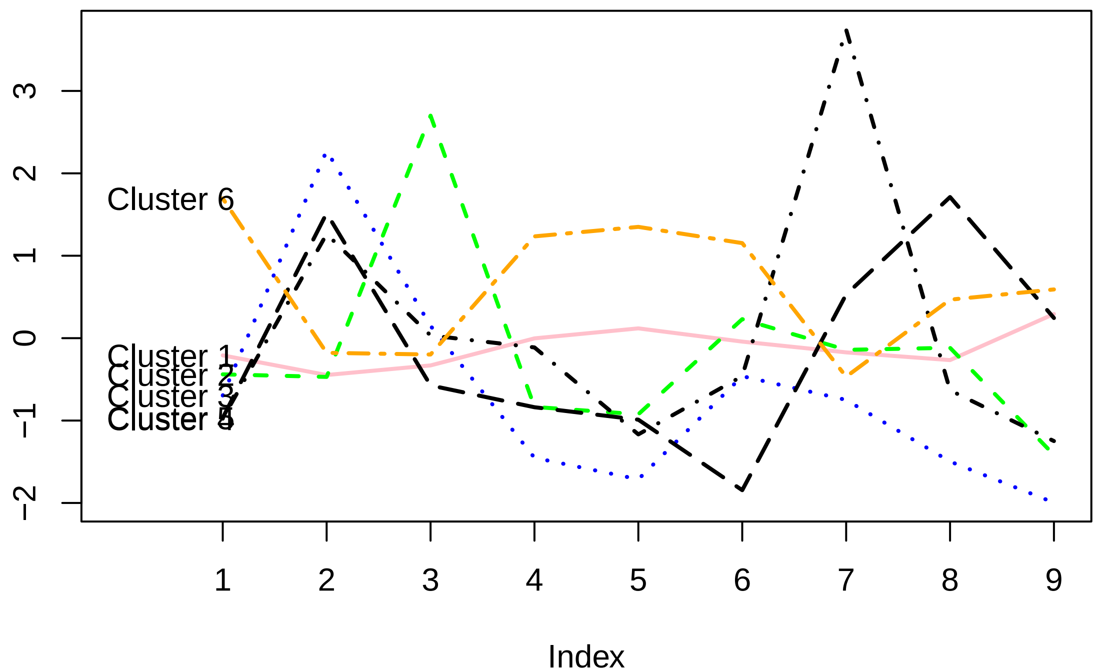
```
## 4  3.7427970 -0.6327607        -1.2488842
## 5  0.5302448  1.7123890         0.2445520
## 6 -0.4680782  0.5647048         0.5798132
```

```
rect.hclust(AverageClust, k=6)
```




```
clust.means <- function(x, res.clust, groups)
{
  if(!is.matrix(x))
    x <- as.matrix(x)
  means <- tapply(x, list(rep(cutree(res.clust, groups), ncol(x)),
                          col(x)),
                  mean)
  dimnames(means) <- list(NULL, dimnames(x)[[2]])
  return(as.data.frame(means))
}


PharmaCentroids<-clust.means(pharmaceutical_data.norm, AverageClust, 6)

y<-apply(as.matrix(PharmaCentroids), 2, as.double)

plot(c(0), xaxt = 'n', ylab = "", type = "l", ylim = c(min(y), max(y)), xlim = c(0,9))

axis(1, at = c(1:9), labels = names(pharmaceutical_data))
```

```
for (i in c(1:6))
  lines(y[i,],  lty = i, lwd = 2,
        col = ifelse(i %in% c(1),"pink",
                     (ifelse(i %in% c(2),"green",
                             (ifelse(i %in% c(3),"blue",
                                     (ifelse(i %in% c(4,5),"black","orange"))))))))

text(x = 0.5, y = PharmaCentroids[, 1], labels = paste("Cluster", c(1:6)))
```



C. I would not necessarily say there is a pattern in the clusters with repsect to the numerical variables. The median recommendation helps us understand if each cluster is a buy, hold or sell. It helps to read the cluster graphs in double checking if you feel these recommendations are reliable.

D. Cluster 1 is our "steady" cluster. It remains mostly in the middle of the graph the entire time. Cluster 2 is "Stable but low Profit margin" by having high PE Ratio but low profit margin. Cluster 3 is the "high risk, low reward". It shoots up for Beta but takes a dive afterwards. Cluster 4 is "high risk, good outcome". Cluster 5 is similar to 3. The risk is high but profit is average so it would be called "high risk, medium payout". Cluster 6 starts out with the highest market cap and very low risk. It finishes somewhere in the middle range for profit. I'd call this "the chosen cluster".