Assignment-5 Hierarchical Clustering

Tanmoy

12/5/2020

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

summary(cars)

## speed dist   
## Min. : 4.0 Min. : 2.00   
## 1st Qu.:12.0 1st Qu.: 26.00   
## Median :15.0 Median : 36.00   
## Mean :15.4 Mean : 42.98   
## 3rd Qu.:19.0 3rd Qu.: 56.00   
## Max. :25.0 Max. :120.00

## Including Plots

You can also embed plots, for example:



Note that the echo = FALSE parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Hierarchical clustering

rm(list = ls())  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(ISLR)  
library(factoextra)

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

library(cluster)   
library(NbClust)   
  
#Loadinng the data  
DFCereals<-read.csv("Cereals.csv")  
  
#EDA of the data set  
summary(DFCereals)

## name mfr type calories   
## Length:77 Length:77 Length:77 Min. : 50.0   
## Class :character Class :character Class :character 1st Qu.:100.0   
## Mode :character Mode :character Mode :character Median :110.0   
## Mean :106.9   
## 3rd Qu.:110.0   
## Max. :160.0   
##   
## protein fat sodium fiber   
## Min. :1.000 Min. :0.000 Min. : 0.0 Min. : 0.000   
## 1st Qu.:2.000 1st Qu.:0.000 1st Qu.:130.0 1st Qu.: 1.000   
## Median :3.000 Median :1.000 Median :180.0 Median : 2.000   
## Mean :2.545 Mean :1.013 Mean :159.7 Mean : 2.152   
## 3rd Qu.:3.000 3rd Qu.:2.000 3rd Qu.:210.0 3rd Qu.: 3.000   
## Max. :6.000 Max. :5.000 Max. :320.0 Max. :14.000   
##   
## carbo sugars potass vitamins   
## Min. : 5.0 Min. : 0.000 Min. : 15.00 Min. : 0.00   
## 1st Qu.:12.0 1st Qu.: 3.000 1st Qu.: 42.50 1st Qu.: 25.00   
## Median :14.5 Median : 7.000 Median : 90.00 Median : 25.00   
## Mean :14.8 Mean : 7.026 Mean : 98.67 Mean : 28.25   
## 3rd Qu.:17.0 3rd Qu.:11.000 3rd Qu.:120.00 3rd Qu.: 25.00   
## Max. :23.0 Max. :15.000 Max. :330.00 Max. :100.00   
## NA's :1 NA's :1 NA's :2   
## shelf weight cups rating   
## Min. :1.000 Min. :0.50 Min. :0.250 Min. :18.04   
## 1st Qu.:1.000 1st Qu.:1.00 1st Qu.:0.670 1st Qu.:33.17   
## Median :2.000 Median :1.00 Median :0.750 Median :40.40   
## Mean :2.208 Mean :1.03 Mean :0.821 Mean :42.67   
## 3rd Qu.:3.000 3rd Qu.:1.00 3rd Qu.:1.000 3rd Qu.:50.83   
## Max. :3.000 Max. :1.50 Max. :1.500 Max. :93.70   
##

colMeans(is.na(DFCereals))

## name mfr type calories protein fat sodium   
## 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000   
## fiber carbo sugars potass vitamins shelf weight   
## 0.00000000 0.01298701 0.01298701 0.02597403 0.00000000 0.00000000 0.00000000   
## cups rating   
## 0.00000000 0.00000000

#Median imputation of missing data   
preProcess\_1<-preProcess(DFCereals, method = c("medianImpute"))  
ImputedDF<-predict(preProcess\_1, DFCereals)  
  
#No more NULL values presnt  
colMeans(is.na(ImputedDF))

## name mfr type calories protein fat sodium fiber   
## 0 0 0 0 0 0 0 0   
## carbo sugars potass vitamins shelf weight cups rating   
## 0 0 0 0 0 0 0 0

#Scaling the DF  
ImputedDF<-subset(ImputedDF, select= -c(1,2,3))  
ImputedDF<-scale(ImputedDF)  
  
  
# Compute with agnes and with different linkage methods  
hc\_single <- agnes(ImputedDF, method = "single")  
hc\_complete<-agnes(ImputedDF, method="complete")  
hc\_average <- agnes(ImputedDF, method = "average")  
hc\_ward <- agnes(ImputedDF, method = "ward")  
  
# Compare Agglomerative coefficients  
print(hc\_single$ac)

## [1] 0.6029274

print(hc\_complete$ac)

## [1] 0.8353216

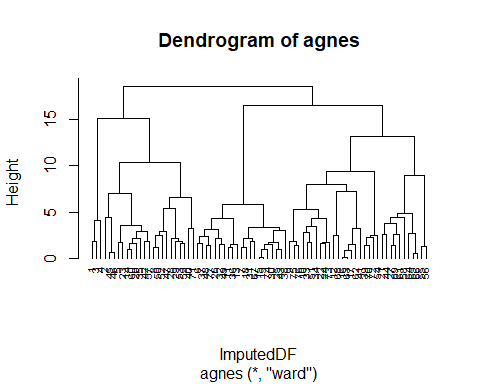
print(hc\_average$ac)

## [1] 0.7777555

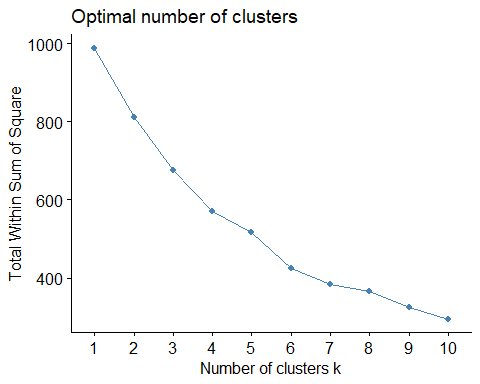
print(hc\_ward$ac)

## [1] 0.9027089

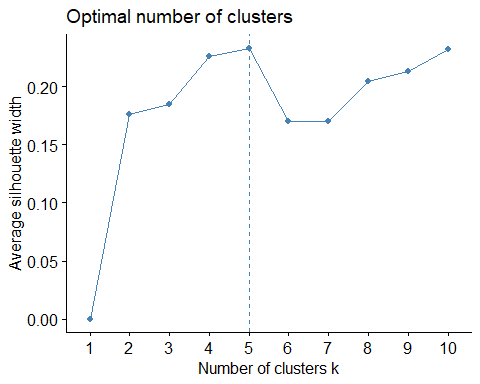
# The approach used by Ward describes the best clustering mechanism of the four approaches tested  
#visualize the dendrogram  
pltree(hc\_ward, cex = 0.6, hang = -1, main = "Dendrogram of agnes")

 Q> Comment on differences between hierarchical Clustering and K-means

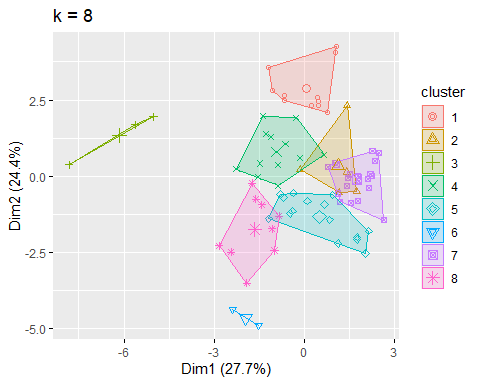
set.seed(123)  
#Finding optimal number of clusters - Elbow Method  
fviz\_nbclust(ImputedDF, kmeans, method = "wss")



#Determining Optimal Cluster by Average Silhouette Method  
fviz\_nbclust(ImputedDF, kmeans, method = "silhouette")



#Silhouette method shows that 8 numbers of clusters would be optimum.   
  
k8 <- kmeans(ImputedDF, centers = 8, nstart = 25)  
fviz\_cluster(k8, geom = "point", data = ImputedDF) + ggtitle("k = 8")



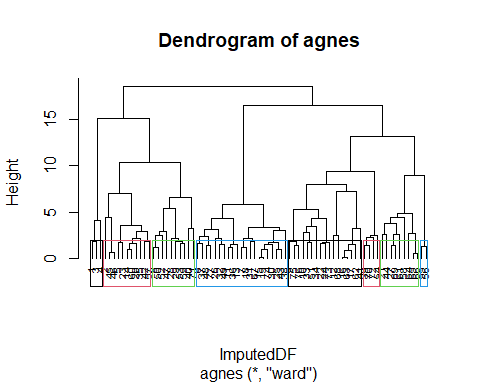
# slicing the dendogram on the longest path, 8 is the optimal level of clusters.  
# Cut tree into 8 groups  
sub\_grp <- cutree(hc\_ward, k = 8)  
C2 <- as.data.frame(cbind(ImputedDF,sub\_grp))  
C2

## calories protein fat sodium fiber carbo  
## 1 -1.8929836 1.3286071 -0.01290349 -0.353984399 3.29284661 -2.52434055  
## 2 0.6732089 0.4151897 3.96137277 -1.725770770 -0.06375361 -1.75148082  
## 3 -1.8929836 1.3286071 -0.01290349 1.196730628 2.87327158 -2.00910073  
## 4 -2.9194605 1.3286071 -1.00647256 -0.234698628 4.97114672 -1.75148082  
## 5 0.1599704 -0.4982277 0.98066557 0.481016000 -0.48332864 -0.20576136  
## 6 0.1599704 -0.4982277 0.98066557 0.242444457 -0.27354112 -1.10743104  
## 7 0.1599704 -0.4982277 -1.00647256 -0.413627285 -0.48332864 -0.97862109  
## 8 1.1864474 0.4151897 0.98066557 0.600301771 -0.06375361 0.82471828  
## 9 -0.8665066 -0.4982277 -0.01290349 0.481016000 0.77539645 0.05185855  
## 10 -0.8665066 0.4151897 -1.00647256 0.600301771 1.19497147 -0.46338127  
## 11 0.6732089 -1.4116451 0.98066557 0.719587543 -0.90290366 -0.72100118  
## 12 0.1599704 3.1554419 0.98066557 1.554587942 -0.06375361 0.56709837  
## 13 0.6732089 -1.4116451 1.97423464 0.600301771 -0.90290366 -0.46338127  
## 14 0.1599704 0.4151897 0.98066557 -0.234698628 -0.06375361 -0.46338127  
## 15 0.1599704 -1.4116451 -0.01290349 0.242444457 -0.90290366 -0.72100118  
## 16 0.1599704 -0.4982277 -1.00647256 1.435302171 -0.90290366 1.85519792  
## 17 -0.3532681 -0.4982277 -1.00647256 1.554587942 -0.48332864 1.59757801  
## 18 0.1599704 -1.4116451 -1.00647256 -0.831127485 -0.48332864 -0.46338127  
## 19 0.1599704 -1.4116451 -0.01290349 0.242444457 -0.90290366 -0.72100118  
## 20 0.1599704 0.4151897 1.97423464 -0.234698628 0.77539645 -1.23624100  
## 21 -0.3532681 0.4151897 -1.00647256 -0.950413256 -0.48332864 1.59757801  
## 22 0.1599704 -0.4982277 -1.00647256 0.719587543 -0.48332864 1.59757801  
## 23 -0.3532681 -0.4982277 -0.01290349 -0.234698628 -0.06375361 -0.97862109  
## 24 -0.3532681 -0.4982277 -1.00647256 0.361730229 -0.48332864 0.82471828  
## 25 0.1599704 -0.4982277 -0.01290349 -0.413627285 -0.48332864 -0.97862109  
## 26 0.1599704 -1.4116451 -1.00647256 0.481016000 -0.48332864 -0.20576136  
## 27 -0.3532681 0.4151897 -1.00647256 -1.904699427 0.35582142 -0.20576136  
## 28 0.6732089 0.4151897 0.98066557 0.003872915 1.19497147 -0.72100118  
## 29 0.6732089 0.4151897 -1.00647256 0.958159085 1.19497147 -0.20576136  
## 30 0.1599704 -1.4116451 -0.01290349 -0.294341514 -0.90290366 -0.46338127  
## 31 -0.3532681 -0.4982277 -1.00647256 -1.367913456 -0.90290366 -0.97862109  
## 32 0.1599704 -1.4116451 -0.01290349 1.435302171 -0.90290366 0.05185855  
## 33 -0.3532681 0.4151897 -0.01290349 -0.234698628 0.35582142 0.05185855  
## 34 0.1599704 0.4151897 -1.00647256 0.123158686 0.35582142 0.56709837  
## 35 0.6732089 0.4151897 1.97423464 -1.010056142 0.35582142 -0.46338127  
## 36 0.6732089 -1.4116451 0.98066557 0.719587543 -0.48332864 -0.72100118  
## 37 0.1599704 0.4151897 -0.01290349 1.077444857 -0.27354112 -0.84981113  
## 38 0.1599704 -1.4116451 -1.00647256 0.242444457 -0.90290366 -0.20576136  
## 39 0.1599704 -0.4982277 -0.01290349 0.123158686 -0.48332864 0.56709837  
## 40 1.6996859 0.4151897 -0.01290349 0.123158686 -0.06375361 1.33995810  
## 41 0.1599704 -0.4982277 -0.01290349 1.196730628 -0.90290366 1.59757801  
## 42 -0.3532681 1.3286071 0.98066557 -0.115412857 -0.06375361 -0.72100118  
## 43 0.1599704 -0.4982277 -0.01290349 0.242444457 -0.90290366 -0.72100118  
## 44 -0.3532681 1.3286071 -0.01290349 -1.904699427 -0.90290366 0.30947846  
## 45 2.2129244 1.3286071 1.97423464 -0.771484599 0.35582142 0.30947846  
## 46 2.2129244 1.3286071 1.97423464 -0.115412857 0.35582142 0.30947846  
## 47 2.7261629 0.4151897 0.98066557 -0.115412857 0.35582142 0.56709837  
## 48 -0.3532681 -0.4982277 -0.01290349 0.719587543 -0.06375361 0.05185855  
## 49 0.6732089 -0.4982277 -0.01290349 0.361730229 -0.90290366 0.05185855  
## 50 1.6996859 0.4151897 0.98066557 0.719587543 0.35582142 1.59757801  
## 51 -0.8665066 0.4151897 -1.00647256 0.123158686 0.35582142 0.82471828  
## 52 1.1864474 0.4151897 0.98066557 0.123158686 -0.27354112 -0.33457131  
## 53 0.6732089 0.4151897 -0.01290349 0.481016000 1.61454650 -0.97862109  
## 54 -0.3532681 0.4151897 -1.00647256 1.912445256 -0.48332864 1.33995810  
## 55 -2.9194605 -1.4116451 -1.00647256 -1.904699427 -0.90290366 -0.46338127  
## 56 -2.9194605 -0.4982277 -1.00647256 -1.904699427 -0.48332864 -1.23624100  
## 57 -0.3532681 1.3286071 -0.01290349 -0.294341514 -0.06375361 -0.20576136  
## 58 -0.3532681 2.2420245 0.98066557 -1.904699427 0.22994891 -0.07695140  
## 59 0.6732089 0.4151897 -0.01290349 0.600301771 1.19497147 -0.20576136  
## 60 -0.3532681 0.4151897 0.98066557 -0.234698628 0.14603391 -1.10743104  
## 61 -0.8665066 -0.4982277 -1.00647256 -1.904699427 -0.06375361 0.05185855  
## 62 0.1599704 -1.4116451 -1.00647256 0.958159085 -0.90290366 2.11281784  
## 63 0.1599704 -0.4982277 -1.00647256 1.554587942 -0.90290366 1.85519792  
## 64 -1.3797451 -0.4982277 -1.00647256 -1.904699427 0.35582142 0.30947846  
## 65 -0.8665066 0.4151897 -1.00647256 -1.904699427 0.77539645 1.08233819  
## 66 -0.8665066 0.4151897 -1.00647256 -1.904699427 0.35582142 1.33995810  
## 67 0.1599704 -0.4982277 -0.01290349 -1.069699027 -0.48332864 -1.49386091  
## 68 0.1599704 3.1554419 -1.00647256 0.838873314 -0.48332864 0.30947846  
## 69 -0.8665066 -0.4982277 -1.00647256 -1.725770770 0.35582142 0.05185855  
## 70 0.1599704 -0.4982277 -0.01290349 0.481016000 -0.90290366 1.59757801  
## 71 1.6996859 0.4151897 -0.01290349 0.361730229 0.77539645 0.05185855  
## 72 -0.3532681 0.4151897 -0.01290349 0.481016000 0.35582142 0.30947846  
## 73 0.1599704 -0.4982277 -0.01290349 1.077444857 -0.90290366 1.59757801  
## 74 0.1599704 -1.4116451 -0.01290349 -0.234698628 -0.90290366 -0.46338127  
## 75 -0.3532681 0.4151897 -0.01290349 0.838873314 0.35582142 0.56709837  
## 76 -0.3532681 0.4151897 -0.01290349 0.481016000 0.35582142 0.56709837  
## 77 0.1599704 -0.4982277 -0.01290349 0.481016000 -0.48332864 0.30947846  
## sugars potass vitamins shelf weight cups  
## 1 -0.235869379 2.61265777 -0.1453172 0.9515734 -0.1967771 -2.11003399  
## 2 0.223926626 0.52608238 -1.2642598 0.9515734 -0.1967771 0.76901001  
## 3 -0.465767382 3.18826478 -0.1453172 0.9515734 -0.1967771 -2.11003399  
## 4 -1.615257394 3.33216653 -0.1453172 0.9515734 -0.1967771 -1.37953029  
## 5 0.223926626 -0.12147550 -0.1453172 0.9515734 -0.1967771 -0.30526014  
## 6 0.683722631 -0.40927901 -0.1453172 -1.4507595 -0.1967771 -0.30526014  
## 7 1.603314640 -0.98488601 -0.1453172 -0.2495930 -0.1967771 0.76901001  
## 8 0.223926626 0.02242625 -0.1453172 0.9515734 1.9962520 -0.30526014  
## 9 -0.235869379 0.38218063 -0.1453172 -1.4507595 -0.1967771 -0.64902659  
## 10 -0.465767382 1.31754201 -0.1453172 0.9515734 -0.1967771 -0.64902659  
## 11 1.143518636 -0.91293514 -0.1453172 -0.2495930 -0.1967771 -0.30526014  
## 12 -1.385359391 0.09437712 -0.1453172 -1.4507595 -0.1967771 1.84328015  
## 13 0.453824628 -0.76903339 -0.1453172 -0.2495930 -0.1967771 -0.30526014  
## 14 -0.005971377 0.09437712 -0.1453172 0.9515734 -0.1967771 -1.37953029  
## 15 1.373416638 -0.62513163 -0.1453172 -0.2495930 -0.1967771 0.76901001  
## 16 -0.925563386 -1.05683689 -0.1453172 -1.4507595 -0.1967771 0.76901001  
## 17 -1.155461389 -0.91293514 -0.1453172 -1.4507595 -0.1967771 0.76901001  
## 18 1.143518636 -1.12878776 -0.1453172 -0.2495930 -0.1967771 0.76901001  
## 19 1.373416638 -0.48122988 -0.1453172 -0.2495930 -0.1967771 0.76901001  
## 20 -0.005971377 0.88583676 -0.1453172 0.9515734 -0.1967771 -1.37953029  
## 21 -1.615257394 -0.12147550 -1.2642598 -0.2495930 -0.1967771 0.76901001  
## 22 -0.925563386 -0.98488601 -0.1453172 0.9515734 -0.1967771 0.76901001  
## 23 0.683722631 0.31022975 -0.1453172 0.9515734 -0.1967771 -0.30526014  
## 24 -0.465767382 -0.26537726 -0.1453172 0.9515734 -0.1967771 -0.30526014  
## 25 1.373416638 -0.98488601 -0.1453172 -0.2495930 -0.1967771 0.76901001  
## 26 0.913620633 -1.05683689 -0.1453172 -1.4507595 -0.1967771 -0.30526014  
## 27 -0.005971377 0.02242625 -0.1453172 -0.2495930 -0.1967771 -0.09040611  
## 28 0.683722631 1.46144376 -0.1453172 0.9515734 1.4646086 -0.64902659  
## 29 1.143518636 1.31754201 -0.1453172 0.9515734 1.9962520 -0.64902659  
## 30 1.143518636 -1.05683689 -0.1453172 -0.2495930 -0.1967771 -0.30526014  
## 31 1.833212643 -0.84098426 -0.1453172 -1.4507595 -0.1967771 0.25336034  
## 32 0.453824628 -0.76903339 -0.1453172 -0.2495930 -0.1967771 -0.30526014  
## 33 -0.465767382 -0.19342638 -0.1453172 0.9515734 -0.1967771 0.25336034  
## 34 -0.925563386 -0.12147550 -0.1453172 0.9515734 -0.1967771 -2.45380043  
## 35 -0.695665384 0.02242625 -0.1453172 0.9515734 -0.1967771 -2.11003399  
## 36 0.913620633 -0.76903339 -0.1453172 -0.2495930 -0.1967771 0.76901001  
## 37 0.683722631 -0.12147550 -0.1453172 -1.4507595 -0.1967771 -0.30526014  
## 38 0.913620633 -0.91293514 -0.1453172 -1.4507595 -0.1967771 2.18704660  
## 39 -0.235869379 -0.55318076 3.2115106 0.9515734 -0.1967771 0.76901001  
## 40 0.453824628 -0.04952463 3.2115106 0.9515734 1.7968857 -0.30526014  
## 41 -0.925563386 -0.84098426 -0.1453172 -0.2495930 -0.1967771 2.91755030  
## 42 -0.235869379 -0.04952463 -0.1453172 -0.2495930 -0.1967771 -0.64902659  
## 43 1.143518636 -0.62513163 -0.1453172 -0.2495930 -0.1967771 0.76901001  
## 44 -0.925563386 -0.04952463 -0.1453172 -0.2495930 -0.1967771 0.76901001  
## 45 0.913620633 1.02973851 -0.1453172 0.9515734 -0.1967771 0.76901001  
## 46 0.913620633 1.02973851 -0.1453172 0.9515734 -0.1967771 0.76901001  
## 47 1.373416638 0.88583676 -0.1453172 0.9515734 3.1259942 -0.64902659  
## 48 -0.235869379 -0.12147550 -0.1453172 -1.4507595 -0.1967771 0.76901001  
## 49 0.453824628 -0.84098426 -0.1453172 -0.2495930 -0.1967771 -0.64902659  
## 50 -0.005971377 0.45413150 -0.1453172 0.9515734 1.9962520 -0.64902659  
## 51 -1.155461389 -0.12147550 -0.1453172 0.9515734 -0.1967771 0.76901001  
## 52 0.683722631 0.31022975 -0.1453172 0.9515734 1.4646086 -1.37953029  
## 53 1.603314640 2.32485427 -0.1453172 0.9515734 1.9962520 -0.64902659  
## 54 -0.925563386 -0.76903339 3.2115106 0.9515734 -0.1967771 0.76901001  
## 55 -1.615257394 -1.20073864 -1.2642598 0.9515734 -3.5195485 0.76901001  
## 56 -1.615257394 -0.69708251 -1.2642598 0.9515734 -3.5195485 0.76901001  
## 57 -0.235869379 0.16632800 -0.1453172 0.9515734 -0.1967771 -1.37953029  
## 58 -0.005971377 0.16632800 -1.2642598 -1.4507595 -0.1967771 -0.64902659  
## 59 1.143518636 2.03705077 -0.1453172 -0.2495930 1.9962520 -0.30526014  
## 60 0.223926626 0.59803325 -0.1453172 0.9515734 -0.1967771 -1.37953029  
## 61 -0.235869379 0.16632800 -0.1453172 0.9515734 -0.1967771 -1.37953029  
## 62 -1.155461389 -0.98488601 -0.1453172 -1.4507595 -0.1967771 1.32763048  
## 63 -0.925563386 -0.91293514 -0.1453172 -1.4507595 -0.1967771 0.76901001  
## 64 -1.615257394 -0.04952463 -1.2642598 -1.4507595 -1.3265194 0.76901001  
## 65 -1.615257394 0.59803325 -1.2642598 -1.4507595 -0.1967771 -0.64902659  
## 66 -1.615257394 0.31022975 -1.2642598 -1.4507595 -0.1967771 -0.64902659  
## 67 1.833212643 -0.84098426 -0.1453172 -0.2495930 -0.1967771 -0.30526014  
## 68 -0.925563386 -0.62513163 -0.1453172 -1.4507595 -0.1967771 0.76901001  
## 69 -0.465767382 -0.12147550 -0.1453172 -0.2495930 -0.1967771 0.76901001  
## 70 -0.925563386 -0.91293514 3.2115106 0.9515734 -0.1967771 0.76901001  
## 71 1.603314640 1.89314902 3.2115106 0.9515734 3.1259942 0.76901001  
## 72 -0.925563386 0.16632800 3.2115106 0.9515734 -0.1967771 0.76901001  
## 73 -0.925563386 -0.55318076 -0.1453172 0.9515734 -0.1967771 -0.30526014  
## 74 1.143518636 -1.05683689 -0.1453172 -0.2495930 -0.1967771 0.76901001  
## 75 -0.925563386 0.23827887 -0.1453172 -1.4507595 -0.1967771 -0.64902659  
## 76 -0.925563386 0.16632800 -0.1453172 -1.4507595 -0.1967771 0.76901001  
## 77 0.223926626 -0.55318076 -0.1453172 -1.4507595 -0.1967771 -0.30526014  
## rating sub\_grp  
## 1 1.83218758 1  
## 2 -0.61805706 2  
## 3 1.19309856 1  
## 4 3.63338491 1  
## 5 -0.58949895 2  
## 6 -0.93656251 3  
## 7 -0.67568989 3  
## 8 -0.40058570 4  
## 9 0.45948710 5  
## 10 0.75801873 5  
## 11 -1.75285455 3  
## 12 0.57657347 5  
## 13 -1.62608831 3  
## 14 -0.16127646 2  
## 15 -1.41872637 3  
## 16 -0.08689833 5  
## 17 0.22763247 5  
## 18 -0.48998167 3  
## 19 -1.44292556 3  
## 20 -0.15781928 2  
## 21 1.55674959 6  
## 22 0.30112138 5  
## 23 -0.46197591 2  
## 24 0.11853896 5  
## 25 -0.74449406 3  
## 26 -0.79942345 3  
## 27 1.11618949 6  
## 28 -0.12448367 4  
## 29 -0.11747555 4  
## 30 -1.04218972 3  
## 31 -0.52773607 3  
## 32 -1.34272615 3  
## 33 0.66996501 5  
## 34 0.76209027 5  
## 35 0.22395859 2  
## 36 -1.48031505 3  
## 37 -0.82531855 3  
## 38 -0.99117283 3  
## 39 -0.43723896 7  
## 40 -0.44095292 4  
## 41 -0.24379018 5  
## 42 0.18952903 2  
## 43 -1.13411138 3  
## 44 0.86744227 6  
## 45 -0.39358784 2  
## 46 -0.60694559 2  
## 47 -0.87934079 4  
## 48 -0.18222306 3  
## 49 -0.90703767 3  
## 50 -0.14048156 4  
## 51 1.20857002 5  
## 52 -0.86955299 4  
## 53 -0.34349055 4  
## 54 -0.08273233 7  
## 55 1.28782197 8  
## 56 1.44796198 8  
## 57 0.48736586 2  
## 58 0.58108630 6  
## 59 -0.24250288 4  
## 60 -0.21088091 2  
## 61 0.90177096 6  
## 62 -0.04746624 5  
## 63 -0.14988985 5  
## 64 1.82029290 6  
## 65 2.26429773 6  
## 66 2.14533086 6  
## 67 -0.81408243 3  
## 68 0.74502768 5  
## 69 1.18871964 6  
## 70 -0.27236281 7  
## 71 -1.00182464 4  
## 72 0.28426404 7  
## 73 -0.25339630 5  
## 74 -1.06158592 3  
## 75 0.50698324 5  
## 76 0.63545985 5  
## 77 -0.46116700 3

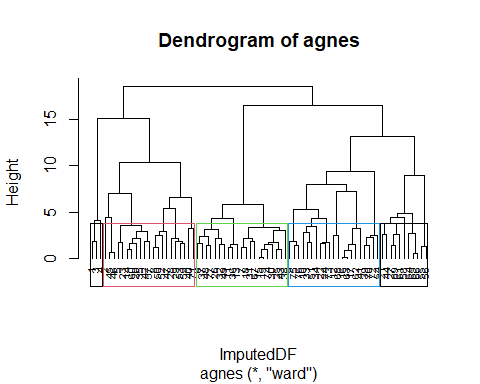
# Number of members in each cluster  
table(sub\_grp)

## sub\_grp  
## 1 2 3 4 5 6 7 8   
## 3 11 21 10 17 9 4 2

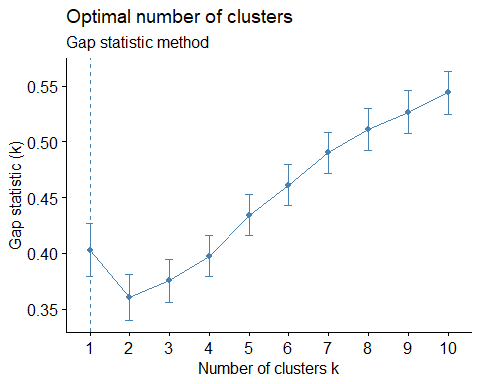
#plot dendrogram  
pltree(hc\_ward, cex = 0.6, hang = -1, main = "Dendrogram of agnes")  
rect.hclust(hc\_ward, k = 8, border = 1:4)



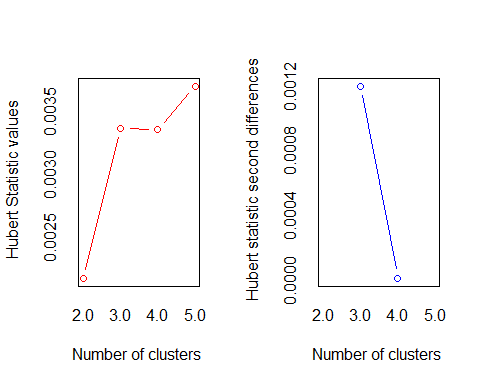
#From the Dendrogram it seems thatclustering at level of 5 would have been optimum.   
pltree(hc\_ward, cex = 0.6, hang = -1, main = "Dendrogram of agnes")  
rect.hclust(hc\_ward, k = 5, border = 1:4)



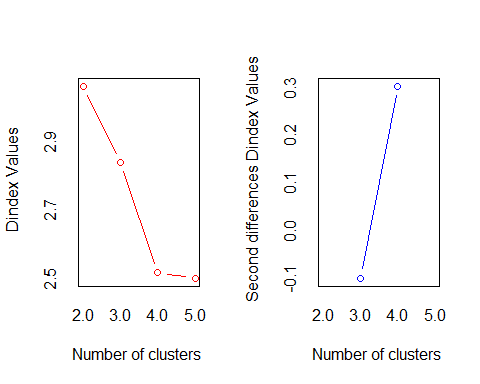
#Thus, Hierarchical clustering would be better approach than kmeans to find the suitable no. of clusters.  
  
# Gap statistic  
set.seed(42)  
fviz\_nbclust(ImputedDF, kmeans,  
 nstart = 25,  
 method = "gap\_stat",  
 nboot = 500  
) + # reduce it for lower computation time (but less precise results)  
 labs(subtitle = "Gap statistic method")



#The optimal number of clusters is the one that maximizes the gap statistic. This method suggests only 1 cluster (which is therefore a useless clustering)  
  
#Three methods do not necessarily lead to the same result. Here, all 3 approaches suggest a different number of clusters.  
  
#A fourth alternative is to use the NbClust() function, which provides 30 indices for choosing the best number of clusters.  
nbclust\_out <- NbClust(  
 data = ImputedDF,  
 distance = "euclidean",  
 min.nc = 2, # minimum number of clusters  
 max.nc = 5, # maximum number of clusters  
 method = "kmeans" # one of: "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median", "centroid", "kmeans"  
)

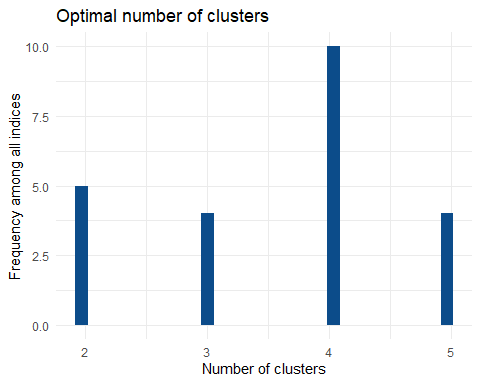


## \*\*\* : The Hubert index is a graphical method of determining the number of clusters.  
## In the plot of Hubert index, we seek a significant knee that corresponds to a   
## significant increase of the value of the measure i.e the significant peak in Hubert  
## index second differences plot.  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



## \*\*\* : 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:   
## \* 5 proposed 2 as the best number of clusters   
## \* 4 proposed 3 as the best number of clusters   
## \* 10 proposed 4 as the best number of clusters   
## \* 4 proposed 5 as the best number of clusters   
##   
## \*\*\*\*\* Conclusion \*\*\*\*\*   
##   
## \* According to the majority rule, the best number of clusters is 4   
##   
##   
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

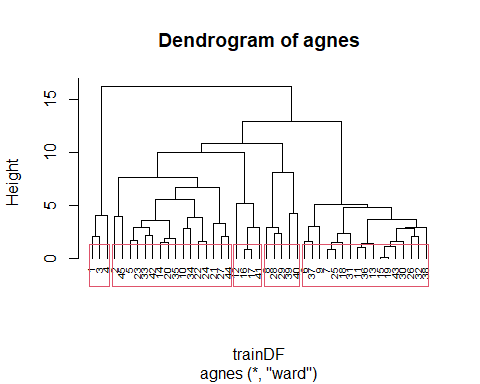
# create a dataframe of the optimal number of clusters  
nbclust\_plot <- data.frame(clusters = nbclust\_out$Best.nc[1, ])  
# select only indices which select between 2 and 5 clusters  
nbclust\_plot <- subset(nbclust\_plot, clusters >= 2 & clusters <= 5)  
  
# create plot  
ggplot(nbclust\_plot) +  
 aes(x = clusters) +  
 geom\_histogram(bins = 30L, fill = "#0c4c8a") +  
 labs(x = "Number of clusters", y = "Frequency among all indices", title = "Optimal number of clusters") +  
 theme\_minimal()



#Based on all 30 indices, the best number of clusters is 5 clusters.  
# I would choose 5 clusters.

Q> Comment on the structure of the clusters and on their stability.

library(caret)  
DF<-ImputedDF  
  
# Cluster partition   
trainDF<-DF[1:45,] # Partition A  
testDF<-DF[46:77,] # Partition B  
trainDF <- scale(trainDF)  
testDF<-scale(testDF)  
  
# The approach used earlier describes that ward is the best clustering mechanism  
hc\_train <- agnes(trainDF, method = "ward")  
  
#visualize the dendrogram  
pltree(hc\_train, cex = 0.6, hang = -1, main = "Dendrogram of agnes")   
rect.hclust(hc\_train, k = 5)



CWcut <- cutree(hc\_train, k = 5)  
CWtotal <- as.data.frame( cbind(trainDF,CWcut))  
head(CWtotal)

## calories protein fat sodium fiber carbo  
## 1 -2.2902757 1.3899279 -0.1190407 -0.5218048 2.74720044 -2.27995823  
## 2 0.7589079 0.4964028 3.4521808 -2.0963468 -0.07044104 -1.51239124  
## 3 -2.2902757 1.3899279 -0.1190407 1.2581124 2.39499525 -1.76824691  
## 4 -3.5099491 1.3899279 -1.0118461 -0.3848881 4.15602117 -1.51239124  
## 5 0.1490712 -0.3971223 0.7737647 0.4366121 -0.42264622 0.02274273  
## 6 0.1490712 -0.3971223 0.7737647 0.1627787 -0.24654363 -0.87275209  
## sugars potass vitamins shelf weight cups  
## 1 -0.44514757 2.45188569 -0.1330313 0.9770643 -0.3068967 -1.8861150  
## 2 0.05564345 0.51266701 -1.6296338 0.9770643 -0.3068967 0.7128963  
## 3 -0.69554308 2.98684257 -0.1330313 0.9770643 -0.3068967 -1.8861150  
## 4 -1.94752061 3.12058179 -0.1330313 0.9770643 -0.3068967 -1.2266644  
## 5 0.05564345 -0.08915948 -0.1330313 0.9770643 -0.3068967 -0.2568840  
## 6 0.55643446 -0.35663792 -0.1330313 -1.6092824 -0.3068967 -0.2568840  
## rating CWcut  
## 1 1.9353757 1  
## 2 -0.4509040 2  
## 3 1.3129704 1  
## 4 3.6895518 1  
## 5 -0.4230914 2  
## 6 -0.7610947 3

CWclust1 <- CWtotal[CWtotal$CWcut==1,]  
 colMeans(CWclust1)

## calories protein fat sodium fiber carbo sugars   
## -2.6968335 1.3899279 -0.4166425 0.1171398 3.0994056 -1.8535321 -1.0294038   
## potass vitamins shelf weight cups rating CWcut   
## 2.8531034 -0.1330313 0.9770643 -0.3068967 -1.6662981 2.3126327 1.0000000

CWclust2 <- CWtotal[CWtotal$CWcut==2,]  
 colMeans(CWclust2)

## calories protein fat sodium fiber carbo   
## 0.03472681 0.44055751 0.27156164 -0.62021364 0.01761026 0.15067056   
## sugars potass vitamins shelf weight cups   
## -0.50774645 0.15742221 -0.32010664 0.65377097 -0.30689671 -0.29325080   
## rating CWcut   
## 0.40553275 2.00000000

CWclust3 <- CWtotal[CWtotal$CWcut==3,]  
 colMeans(CWclust3)

## calories protein fat sodium fiber carbo sugars   
## 0.1490712 -0.8701650 -0.1190407 0.1144552 -0.5262360 -0.3685659 0.8952049   
## potass vitamins shelf weight cups rating CWcut   
## -0.6359169 -0.1330313 -0.7725232 -0.3068967 0.2861930 -0.8097474 3.0000000

CWclust4 <- CWtotal[CWtotal$CWcut==4,]  
 colMeans(CWclust4)

## calories protein fat sodium fiber carbo   
## 1.00284261 0.31769781 0.05952036 0.29969545 0.28176415 0.58562518   
## sugars potass vitamins shelf weight cups   
## 0.30603895 0.43242348 1.66289165 0.97706430 2.45517366 -0.18705985   
## rating CWcut   
## -0.14518909 4.00000000

CWclust5 <- CWtotal[CWtotal$CWcut==5,]  
 colMeans(CWclust5)

## calories protein fat sodium fiber carbo   
## -0.003387982 0.496402831 -0.342242063 1.531945766 -0.510697517 1.621840611   
## sugars potass vitamins shelf weight cups   
## -1.384130721 -0.607398956 -0.133031332 -1.285989048 -0.306896708 1.440231546   
## rating CWcut   
## 0.266307344 5.000000000

CWmeans1 <- rbind(colMeans(CWclust1),colMeans(CWclust2),colMeans(CWclust3),colMeans(CWclust4) ,colMeans(CWclust5))  
head(CWmeans1)

## calories protein fat sodium fiber carbo  
## [1,] -2.696833500 1.3899279 -0.41664251 0.1171398 3.09940562 -1.8535321  
## [2,] 0.034726813 0.4405575 0.27156164 -0.6202136 0.01761026 0.1506706  
## [3,] 0.149071198 -0.8701650 -0.11904072 0.1144552 -0.52623598 -0.3685659  
## [4,] 1.002842608 0.3176978 0.05952036 0.2996954 0.28176415 0.5856252  
## [5,] -0.003387982 0.4964028 -0.34224206 1.5319458 -0.51069752 1.6218406  
## sugars potass vitamins shelf weight cups  
## [1,] -1.0294038 2.8531034 -0.1330313 0.9770643 -0.3068967 -1.6662981  
## [2,] -0.5077464 0.1574222 -0.3201066 0.6537710 -0.3068967 -0.2932508  
## [3,] 0.8952049 -0.6359169 -0.1330313 -0.7725232 -0.3068967 0.2861930  
## [4,] 0.3060390 0.4324235 1.6628917 0.9770643 2.4551737 -0.1870599  
## [5,] -1.3841307 -0.6073990 -0.1330313 -1.2859890 -0.3068967 1.4402315  
## rating CWcut  
## [1,] 2.3126327 1  
## [2,] 0.4055327 2  
## [3,] -0.8097474 3  
## [4,] -0.1451891 4  
## [5,] 0.2663073 5

Q> The elementary public schools would like to choose a set of cereals to include in their daily cafeterias. Every day a different cereal is offered, but all cereals should support a healthy diet. For this goal, you are requested to find a cluster of “healthy cereals.” Should the data be normalized? If not, how should they be used in the cluster analysis?

#install.packages("hrbrthemes")  
library(GGally)

## Registered S3 method overwritten by 'GGally':  
## method from   
## +.gg ggplot2

library(ggplot2)  
library(hrbrthemes)

## NOTE: Either Arial Narrow or Roboto Condensed fonts are required to use these themes.

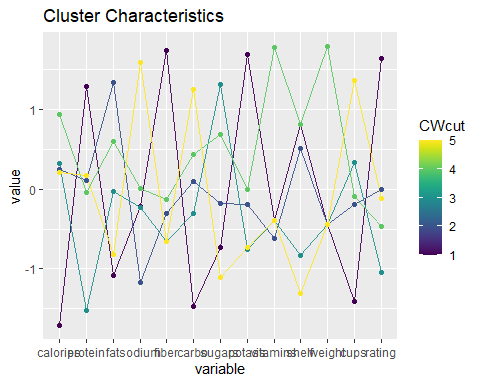
## Please use hrbrthemes::import\_roboto\_condensed() to install Roboto Condensed and

## if Arial Narrow is not on your system, please see https://bit.ly/arialnarrow

library(viridis)

## Loading required package: viridisLite

#ggparcoord(cbind(c(1:4),CWmeans),columns = 2:14,groupColumn = 1,showPoints = TRUE,title = " Charter of cluster",alphaLines = 0.9) + scale\_color\_viridis (discrete = TRUE)+theme\_ipsum()+theme(plot.title = element\_text(size = 10))   
ggparcoord(CWmeans1,  
 columns = 1:13, groupColumn = 14,  
 showPoints = TRUE,   
 title = "Cluster Characteristics",  
 alphaLines = 0.9  
) +   
 scale\_color\_viridis(discrete=FALSE)



# Based on the characteristics of the cluster, it is clear that Cluster 1 is the strongest with low calories, high protein, potassium, fiber top-rated.  
###In general,when we use the distance metric algorithm the data should be normalized,because the data characteristics are diverse. Therefore, it is necessary to standardize the data.

Q> How do you compare hierarchical clustering and k-means? What are they main advantages of hierarchical clustering compared to k-means?

Ans: Clustering is a subjective statistical analysis and there can be more than one appropriate algorithm, depending on the dataset at hand or the type of problem to be solved. So choosing between k-means and hierarchical clustering is not always easy. If the cluster size is known or if we know that there is a specific number of clusters in our dataset (for example if we would like to distinguish diseased and healthy patients depending on some characteristics but we do not know in which group patients belong to), we should probably opt for the k-means clustering as this technique is used when the number of groups is specified in advance. If the number of groups or clusters in the dataset is unknown (for instance in marketing when trying to distinguish clients without any prior belief on the number of different types of customers), then we should probably opt for the hierarchical clustering to determine in how many clusters the data should be divided.