Hierarchical Clustering

Dutt Thakkar

2023-05-15

Loading the dataset

```
data=read.csv("/Users/duttthakkar/Desktop/Cereals.csv")
head(data)
```

```
##                         name mfr type calories protein fat sodium fiber carbo
## 1                  100%_Bran   N    C       70       4   1    130  10.0   5.0
## 2          100%_Natural_Bran   Q    C      120       3   5     15   2.0   8.0
## 3                   All-Bran   K    C       70       4   1    260   9.0   7.0
## 4 All-Bran_with_Extra_Fiber   K    C       50       4   0    140  14.0   8.0
## 5             Almond_Delight   R    C      110       2   2    200   1.0  14.0
## 6   Apple_Cinnamon_Cheerios   G    C      110       2   2    180   1.5  10.5
##   sugars potass vitamins shelf weight cups   rating
## 1      6    280       25     3      1 0.33 68.40297
## 2      8    135        0     3      1 1.00 33.98368
## 3      5    320       25     3      1 0.33 59.42551
## 4      0    330       25     3      1 0.50 93.70491
## 5      8     NA       25     3      1 0.75 34.38484
## 6     10     70       25     1      1 0.75 29.50954
```

Viewing the summary and structure of that dataset

```
summary(data)
```

```
##      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  
## 
```

```
str(data)
```

```
## 'data.frame':    77 obs. of  16 variables:
##  $ name    : chr  "100%_Bran" "100%_Natural_Bran" "All-Bran" "All-Bran_with_Extra_
Fiber" ...
##  $ mfr     : chr  "N" "Q" "K" "K" ...
##  $ type    : chr  "C" "C" "C" "C" ...
##  $ calories: int  70 120 70 50 110 110 110 130 90 90 ...
##  $ protein : int  4 3 4 4 2 2 2 3 2 3 ...
##  $ fat     : int  1 5 1 0 2 2 0 2 1 0 ...
##  $ sodium  : int  130 15 260 140 200 180 125 210 200 210 ...
##  $ fiber   : num  10 2 9 14 1 1.5 1 2 4 5 ...
##  $ carbo   : num  5 8 7 8 14 10.5 11 18 15 13 ...
##  $ sugars  : int  6 8 5 0 8 10 14 8 6 5 ...
##  $ potass  : int  280 135 320 330 NA 70 30 100 125 190 ...
##  $ vitamins: int  25 0 25 25 25 25 25 25 25 25 ...
##  $ shelf   : int  3 3 3 3 3 1 2 3 1 3 ...
##  $ weight  : num  1 1 1 1 1 1 1 1.33 1 1 ...
##  $ cups    : num  0.33 1 0.33 0.5 0.75 0.75 1 0.75 0.67 0.67 ...
##  $ rating  : num  68.4 34 59.4 93.7 34.4 ...
```

Loading required packages

```
library(caret)
```

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

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

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(ggcorrplot)
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.0     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ lubridate 1.9.2     ✔ tibble    3.1.8
## ✔ purrr     1.0.1     ✔ tidyr     1.3.0
## ── Conflicts ──────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ purrr::lift()   masks caret::lift()
## ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force all conf
licts to become errors
```

```
library(tidyr)
library(dplyr)
library(e1071)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve
3WBa
```

```
library(cluster)
library(cowplot)
```

```
##
## Attaching package: 'cowplot'
##
## The following object is masked from 'package:lubridate':
##
##     stamp
```

```
library(pander)
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
##
## The following object is masked from 'package:purrr':
##
##     cross
##
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(FactoMineR)
```

Data Preprocessing. Remove all cereals with missing values.

```
dim(data)
```

```
## [1] 77 16
```

```
c_d2=na.omit(data)
dim(c_d2)
```

```
## [1] 74 16
```

There were 4 missing values in the dataset

Assigning row names to the cereal column

```
c_d3 = as.data.frame(c_d2)
row.names(c_d3) = c_d3[,1]
c_d4 = c_d3[,-1]
```

Only selecting numerical values and removing catergorical variables

```
c_d5 = c_d4[, c(3:11,13:15)]
```

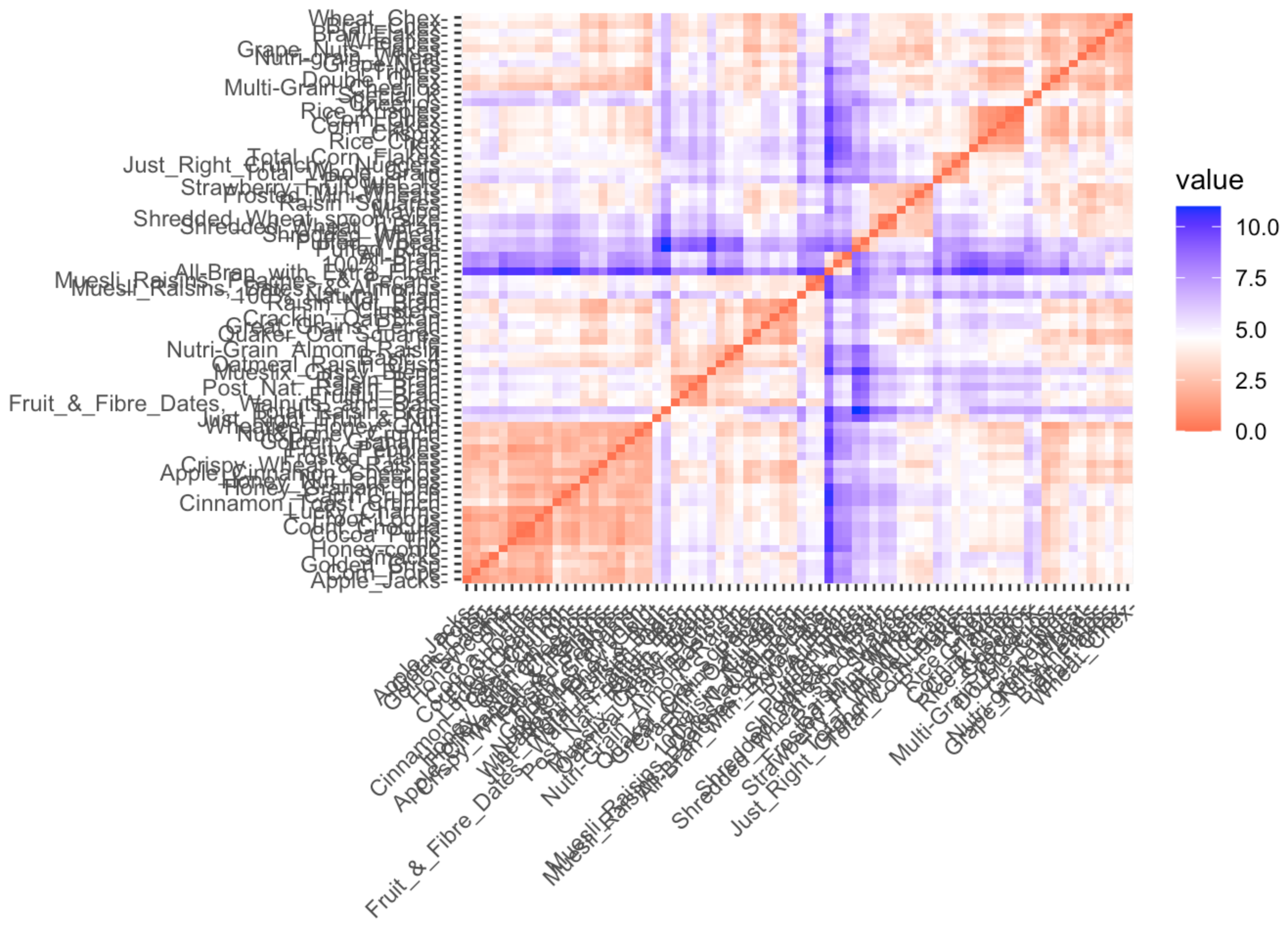
Normalizing the data using the scale function

```
c_d5 = scale(c_d5)
head(c_d5)
```

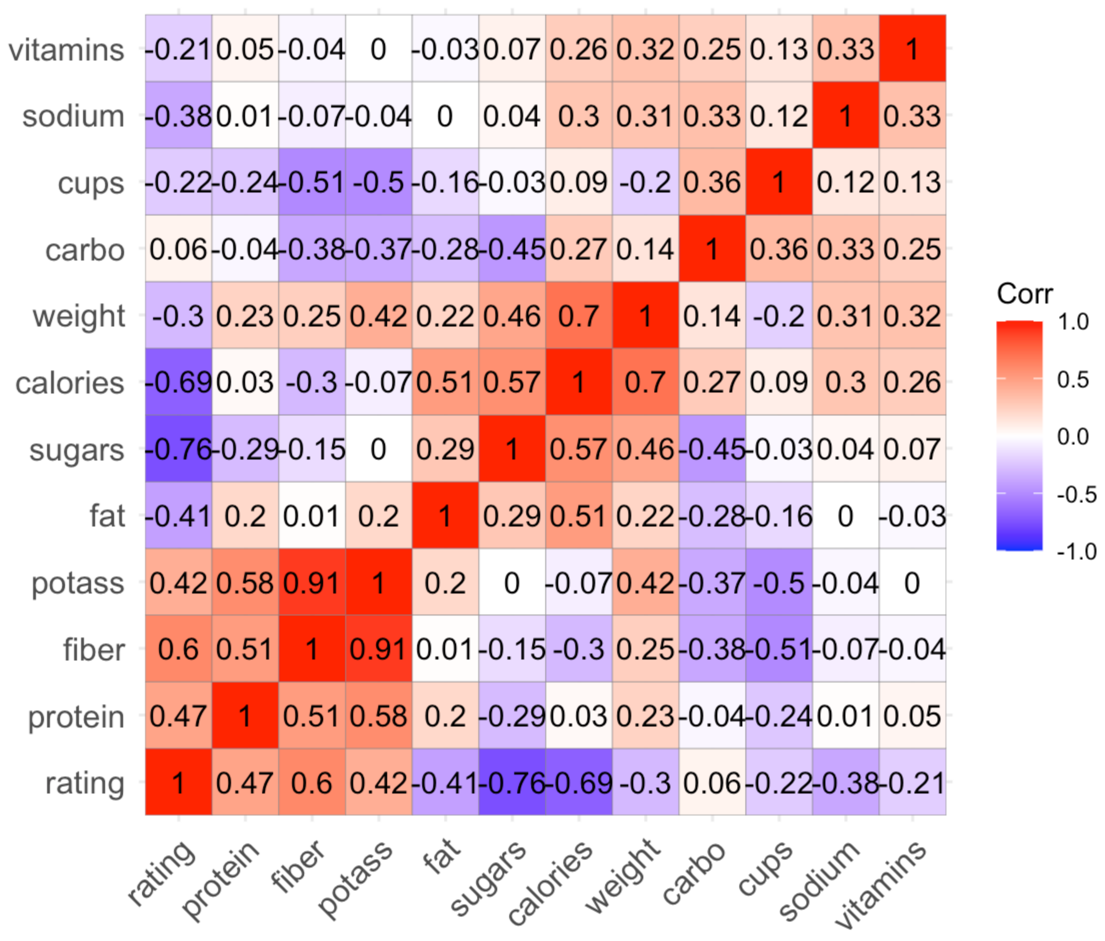
```
##                            calories    protein        fat     sodium
## 100%_Bran                -1.8659155  1.3817478  0.0000000 -0.3910227
## 100%_Natural_Bran         0.6537514  0.4522084  3.9728810 -1.7804186
## All-Bran                 -1.8659155  1.3817478  0.0000000  1.1795987
## All-Bran_with_Extra_Fiber -2.8737823  1.3817478 -0.9932203 -0.2702057
## Apple_Cinnamon_Cheerios   0.1498180 -0.4773310  0.9932203  0.2130625
## Apple_Jacks               0.1498180 -0.4773310 -0.9932203 -0.4514312
##                                 fiber      carbo     sugars     potass
## 100%_Bran                 3.22866747 -2.5001396 -0.2542051  2.5605229
## 100%_Natural_Bran        -0.07249167 -1.7292632  0.2046041  0.5147738
## All-Bran                  2.81602258 -1.9862220 -0.4836096  3.1248675
## All-Bran_with_Extra_Fiber  4.87924705 -1.7292632 -1.6306324  3.2659536
## Apple_Cinnamon_Cheerios  -0.27881412 -1.0868662  0.6634132 -0.4022862
## Apple_Jacks              -0.48513656 -0.9583868  1.5810314 -0.9666308
##                            vitamins     weight       cups     rating
## 100%_Bran                -0.1818422 -0.2008324 -2.0856582  1.8549038
## 100%_Natural_Bran        -1.3032024 -0.2008324  0.7567534 -0.5977113
## All-Bran                 -0.1818422 -0.2008324 -2.0856582  1.2151965
## All-Bran_with_Extra_Fiber -0.1818422 -0.2008324 -1.3644493  3.6578436
## Apple_Cinnamon_Cheerios  -0.1818422 -0.2008324 -0.3038480 -0.9165248
## Apple_Jacks              -0.1818422 -0.2008324  0.7567534 -0.6553998
```

Question 1 (part A): Apply hierarchical clustering to the data using Euclidean distance to the normalized measurements and looking at the correaltion values by plotting the corrplot

```
distance_table <- get_dist(c_d5)
fviz_dist(distance_table)
```



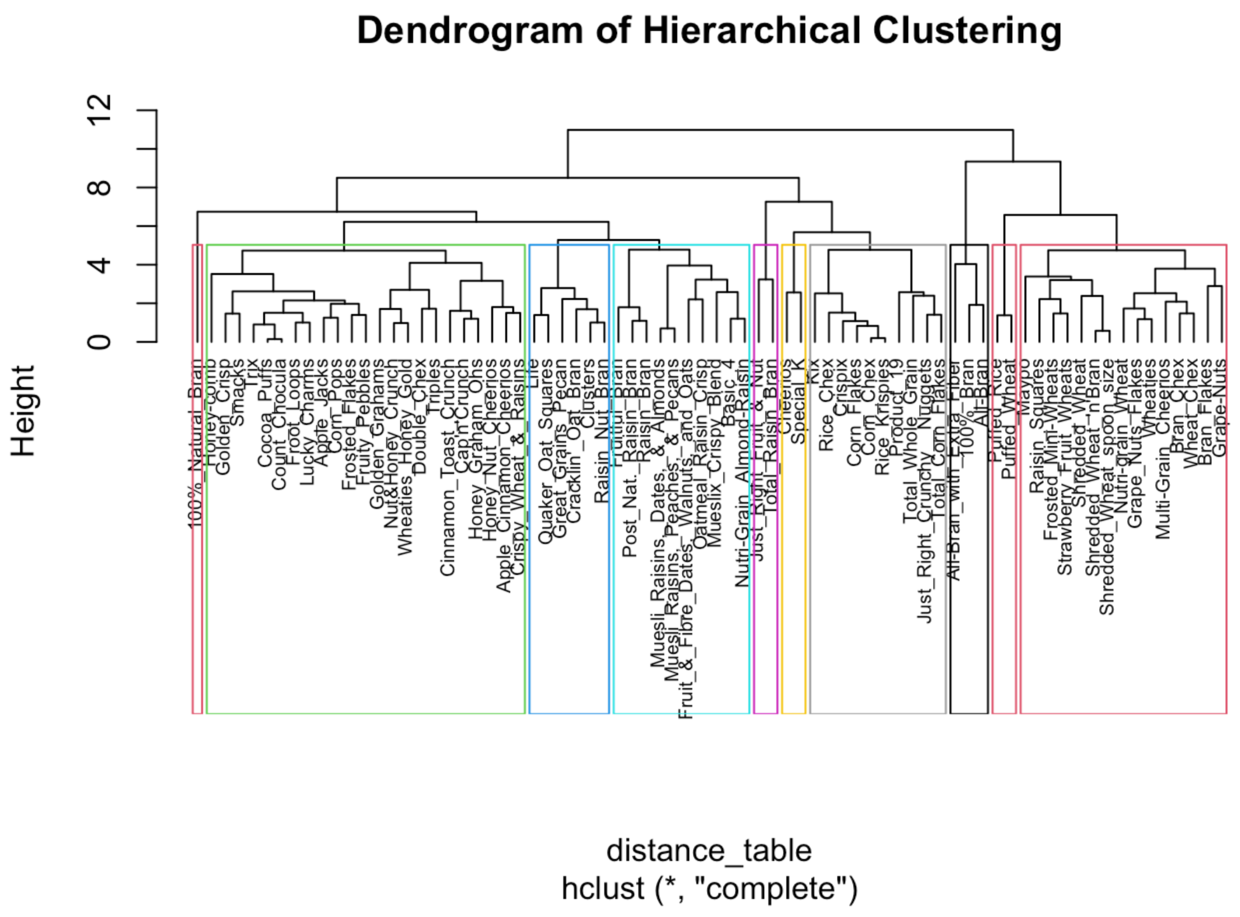
```
corr_plot = cor(c_d5)
ggcorrplot(corr_plot, outline.color = "grey50", lab = TRUE, hc.order = TRUE, type = "
full")
```



```
#Sugar and calories are highly negatively correlated with rating. Also, Potass is hig
hly positively correlated with fiber and Protien.
```

Question 1 (part B): comparing hierarchical clustering with different linkages: single, average, complete and ward.

```
# Hierarchical clustering using Complete Linkage
hc1 <- hclust(distance_table, method = "complete" )
# Plot the obtained dendrogram
plot(hc1, cex = 0.6, hang = -1, main = "Dendrogram of Hierarchical Clustering")
rect.hclust(hc1, k = 10, border = 2:10)
```



#

Computing with AGNES and with different linkage methods

```
hc_single <- agnes(distance_table, method = "single")
print(hc_single$ac)
```

```
## [1] 0.6072384
```

```
hc_complete <- agnes(distance_table, method = "complete")
print(hc_complete$ac)
```

```
## [1] 0.8469328
```

```
hc_average <- agnes(distance_table, method = "average")
print(hc_average$ac)
```

```
## [1] 0.7881955
```

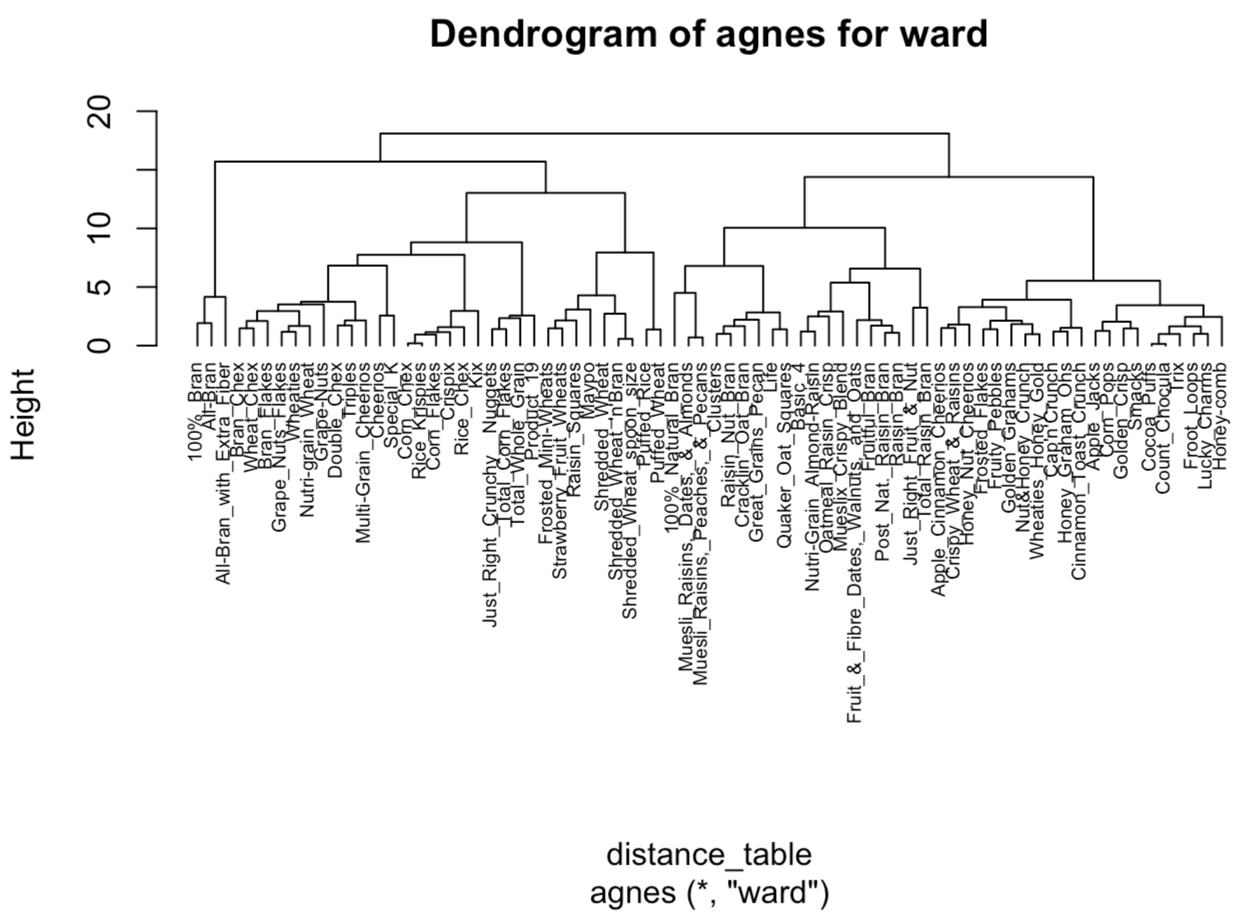
```
hc_ward <- agnes(distance_table, method = "ward")
print(hc_ward$ac)
```

```
## [1] 0.9087265
```

```
#These results confirm that the Ward linkage, which provides 90.87% accuracy, is the
optimal agglomerative (AGNES) linkage to use.
```

Visualizing the dendogram

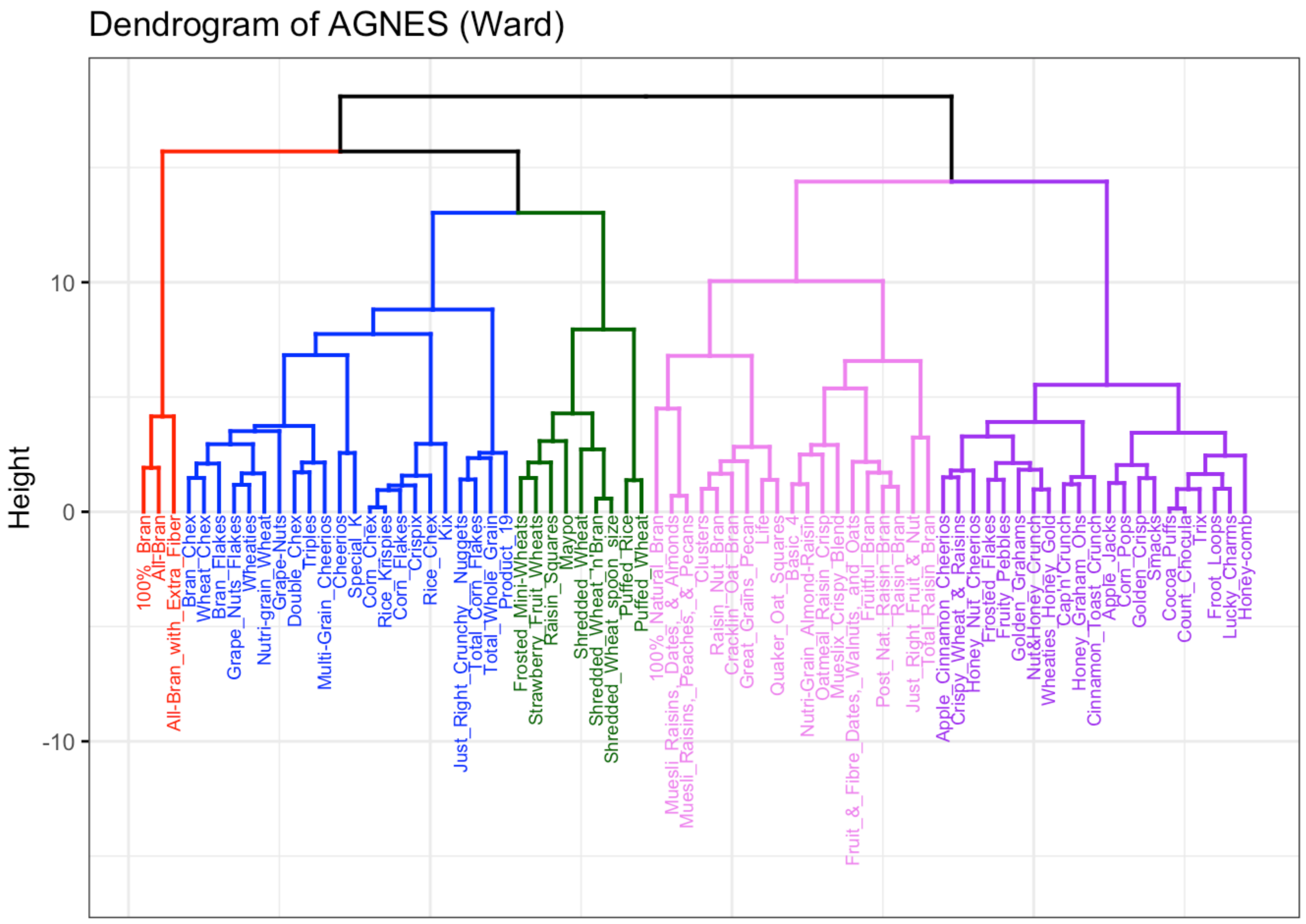
```
hc_Ward <- agnes(distance_table, method = "ward")
pltree(hc_Ward, cex = 0.6, hang = -1, main = "Dendrogram of agnes for ward")
```



Question 2: How many cluster would you choose?

```
# The largest difference in height can be used to determine the k value hence K =5 is
the best option.

fviz_dend(hc_ward, k = 5,main = "Dendrogram of AGNES (Ward)",cex = 0.5, k_colors = c
("red", "blue", "darkgreen", "violet", "purple"), color_labels_by_k = TRUE,labels_tra
ck_height = 16,ggtheme = theme_bw())
```

```
## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instea
d as
## of ggplot2 3.3.4.
## ℹ The deprecated feature was likely used in the factoextra package.
##   Please report the issue at <]8;;https://github.com/kassambara/factoextra/issuesh
ttps://github.com/kassambara/factoextra/issues]8;;>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



```
c_d6 <- cutree(hc_ward, k = 5)
Clustered_df <-as.data.frame(cbind ( c_d5, c_d6 ))
```

Question 3: Comment on the structure of the clusters and on their stability. Hint: To check stability, partition the data and see how well clusters formed based on one part apply to the other part

For the stability of the clusters, We will partition the data into A and B.

```
cereal_a = c_d5[1:55,]
cereal_b = c_d5[56:74,]
```

Computing the distances of cereal_a

```
distance_cereal_a = get_dist(cereal_a)
```

#Compute with AGNES and with different linkage methods for cereal_a

```
hc_single_cereal_a <- agnes(distance_cereal_a, method = "single")
print(hc_single_cereal_a$ac)
```

```
## [1] 0.6663587
```

```
hc_complete_cereal_a <- agnes(distance_cereal_a, method = "complete")
print(hc_complete_cereal_a$ac)
```

```
## [1] 0.8285192
```

```
hc_average_cereal_a <- agnes(distance_cereal_a, method = "average")
print(hc_average_cereal_a$ac)
```

```
## [1] 0.7646836
```

```
hc_ward_cereal_a <- agnes(distance_cereal_a, method = "ward")
print(hc_ward_cereal_a$ac)
```

```
## [1] 0.8891086
```

With 88.91% accuracy, it enables us to establish that the best linkage for cereal_a is Ward.

Computing the distances of cereal_a

```
distance_cereal_b = get_dist(cereal_b)
```

#Compute with AGNES and with different linkage methods for cereal_b

```
hc_single_cereal_b <- agnes(distance_cereal_b, method = "single")
print(hc_single_cereal_b$ac)
```

```
## [1] 0.4805129
```

```
hc_complete_cereal_b <- agnes(distance_cereal_b, method = "complete")
print(hc_complete_cereal_b$ac)
```

```
## [1] 0.71298
```

```
hc_average_cereal_b <- agnes(distance_cereal_b, method = "average")
print(hc_average_cereal_b$ac)
```

```
## [1] 0.6232053
```

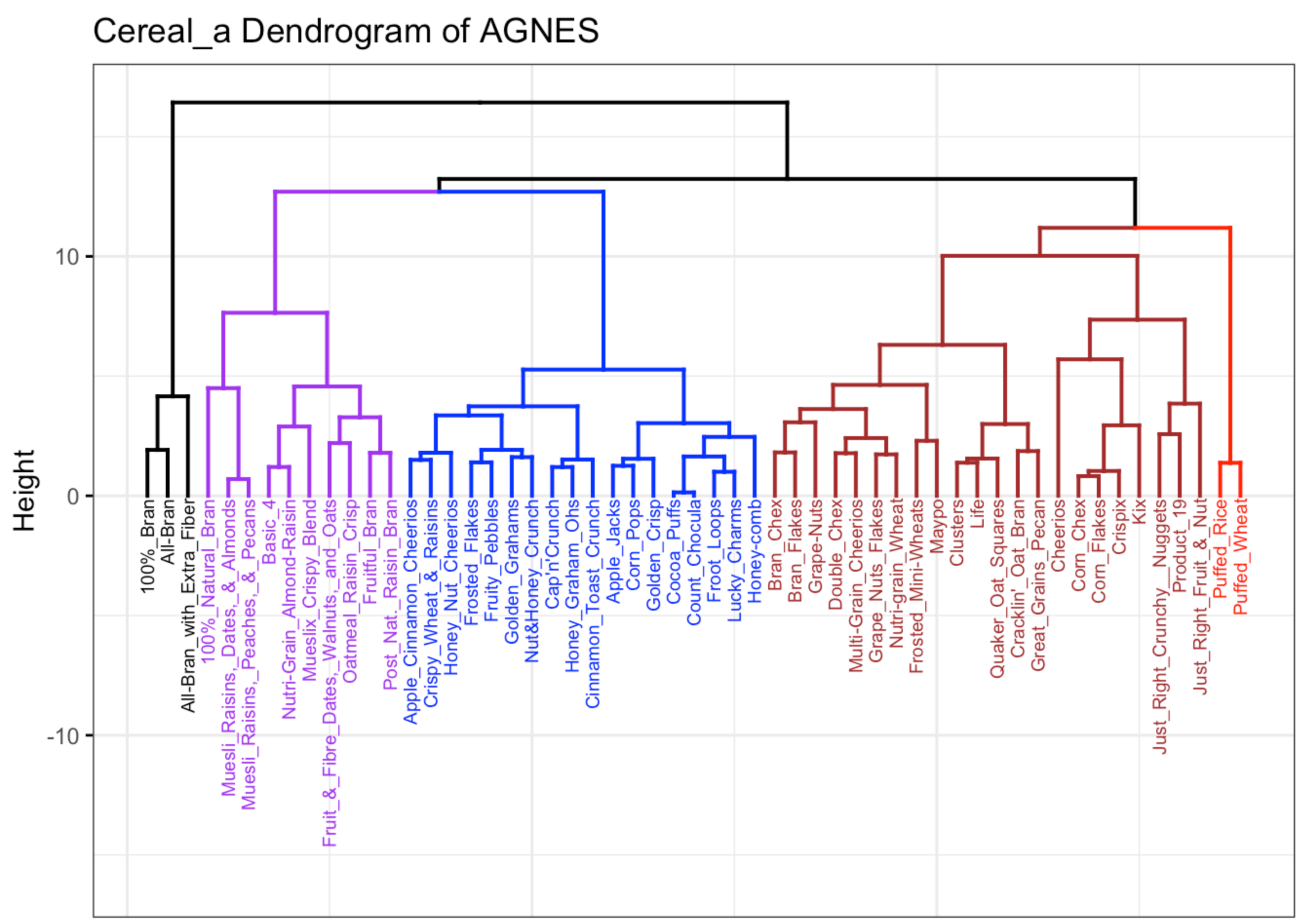
```
hc_ward_cereal_b <- agnes(distance_cereal_b, method = "ward")
print(hc_ward_cereal_b$ac)
```

```
## [1] 0.7710122
```

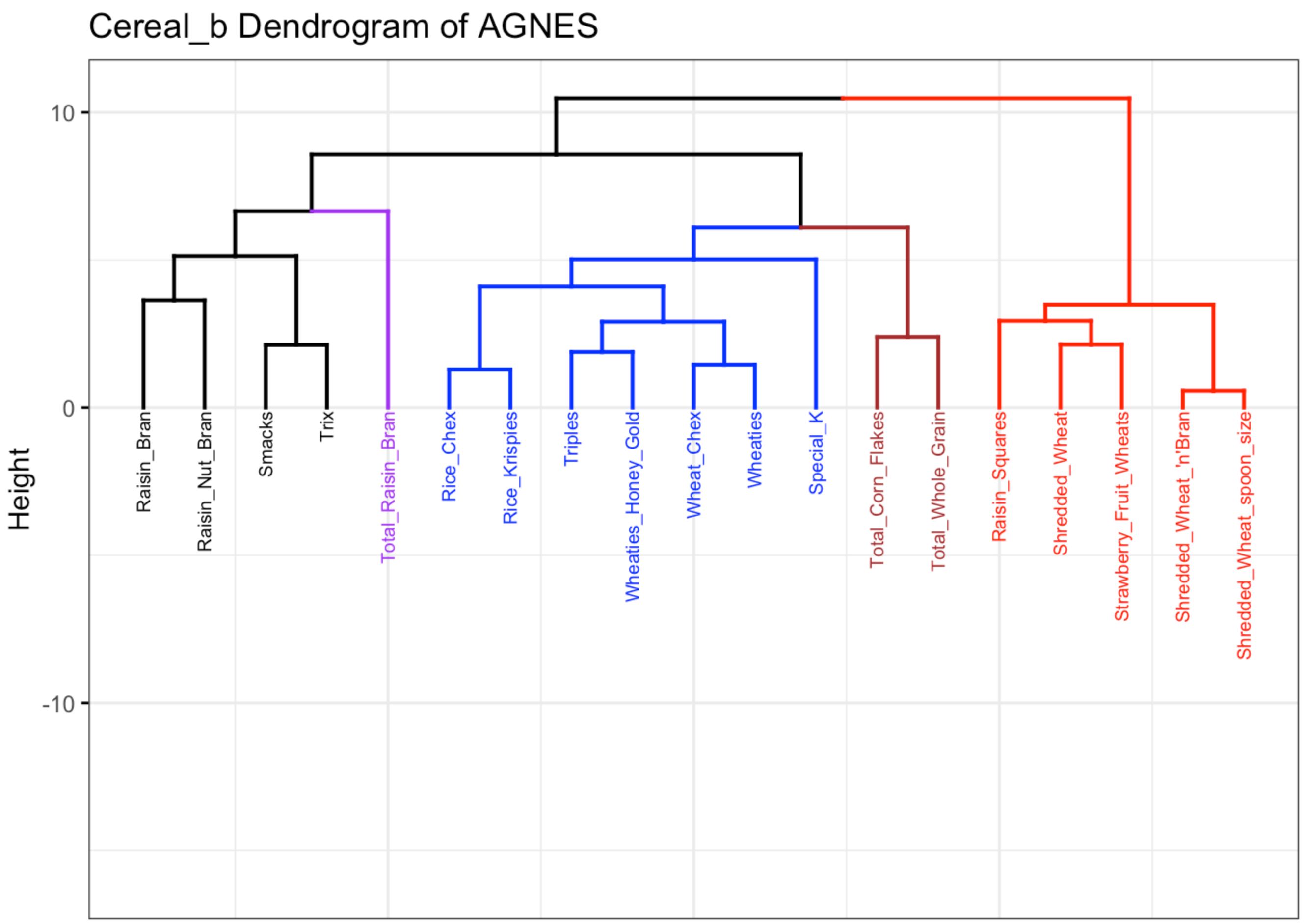
With 77.10% accuracy, it enables us to establish that the best linkage for cereal_a is Ward.

Plotting dendogram of cereal_a and cereal_b

```
fviz_dend(hc_ward_cereal_a, k = 5,main = "Cereal_a Dendrogram of AGNES",cex = 0.5, k_
colors = c("black", "purple", "blue", "brown", "red"), color_labels_by_k = TRUE,label
s_track_height = 16,ggtheme = theme_bw())
```



```
fviz_dend(hc_ward_cereal_b, k = 5,main = "Cereal_b Dendrogram of AGNES",cex = 0.5, k_
colors = c("black", "purple", "blue", "brown", "red"), color_labels_by_k = TRUE,label
s_track_height = 16,ggtheme = theme_bw())
```



Question 3 (part B): Use the cluster centroids from A to assign each record in partition B (each record is assigned to the cluster with the closest centroid)

```
Clustered_df_A <-cutree (hc_ward_cereal_a, k=5)
Clusters_A <-as.data.frame(cbind(cereal_a, Clustered_df_A))
Clust_1 <- colMeans (Clusters_A [Clusters_A$ Clustered_df_A == "1" ,])
# The centroid of cluster 1 is represented by a vector of mean values for each column
of the data as a result.
```

```
Clustered_df_B <-cutree (hc_ward_cereal_b, k=5)
Clusters_B <-as.data.frame(cbind(cereal_b, Clustered_df_B))
Clust_2 <- colMeans (Clusters_B [Clusters_B$ Clustered_df_B == "1" ,])
# The centroid of cluster 2 is represented by a vector of mean values for each column
of the data as a result.
```

```
Centroid <-rbind(Clust_1, Clust_2)
Centroid
```

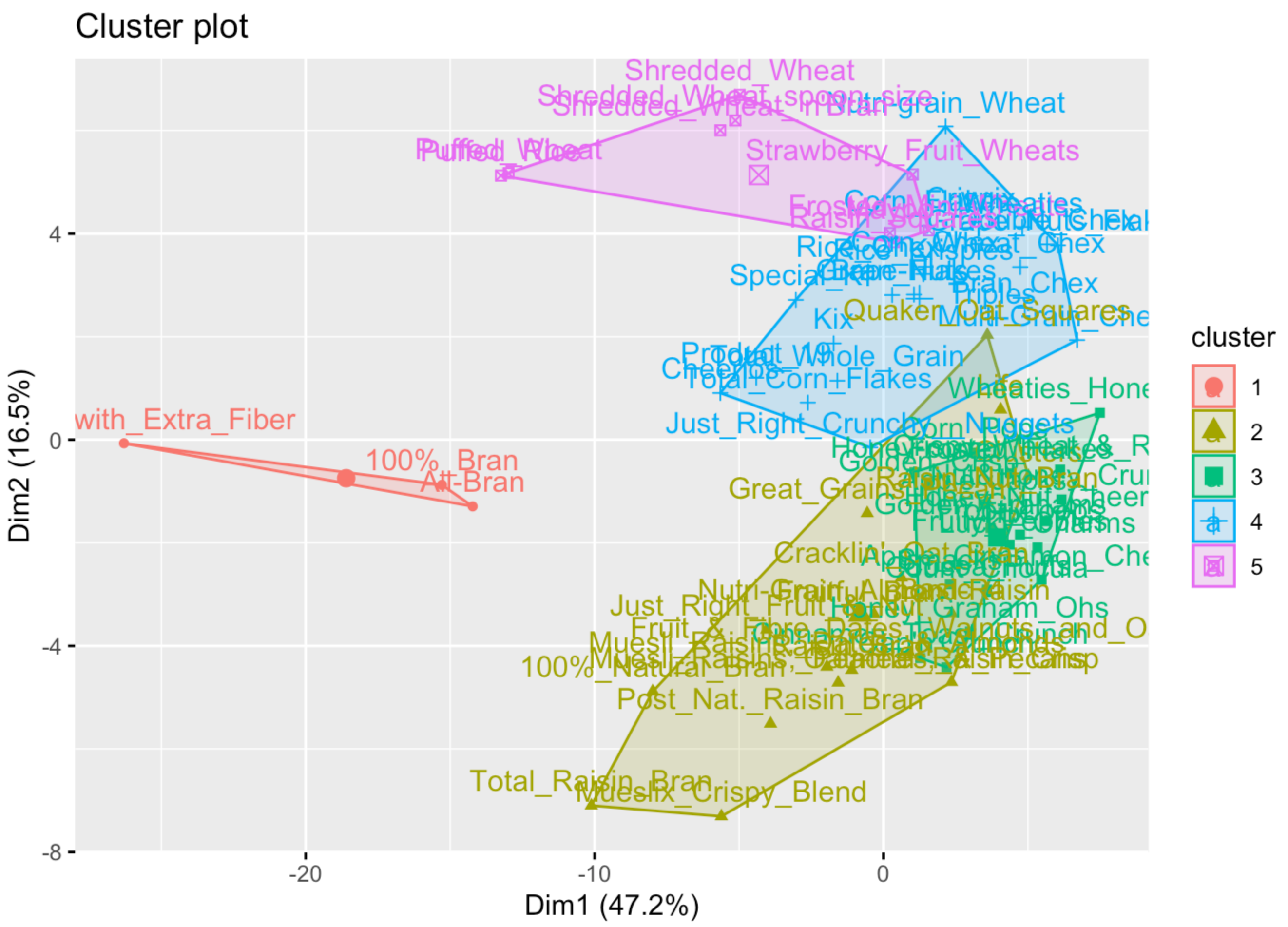
```
##          calories    protein        fat     sodium       fiber      carbo
## Clust_1 -2.201871  1.3817478 -0.3310734  0.1727901  3.64131237 -2.0718749
## Clust_2  0.149818 -0.2449462  0.2483051 -0.2702057 -0.02091106 -0.7977876
##             sugars    potass   vitamins     weight      cups     rating
## Clust_1 -0.7894824 2.9837813 -0.1818422 -0.2008324 -1.845255  2.2426479
## Clust_2  1.0648712 0.1796942 -0.1818422  0.3369228 -0.303848 -0.5618826
##         Clustered_df_A
## Clust_1              1
## Clust_2              1
```

Question 3 (part C): Assess how consistent the cluster assignments are compared to the assignments based on all the data.

```
# After reviewing the centroid, it shows that cluster 1 is high in protein, fiber, an
d potassium. It means that the cereals in cluster 1 is more healthier than cluster 2.
It can also be supported by looking at calories, fat, carbs, and sugar levels which a
re higher in cluster 2 as compared to cluster 1. Thus cereals in cluster 1 are health
ier.
```

#Q4: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?

```
#Visualizing the clusters in Scatter plot
fviz_cluster(list(data=distance_table, cluster = c_d6))
```



```
Healthy_cereal<- cbind(c_d2,c_d6)
mean(Healthy_cereal[Healthy_cereal$c_d6==1,"rating"])
```

```
## [1] 73.84446
```

```
mean(Healthy_cereal[Healthy_cereal$c_d6==2,"rating"])
```

```
## [1] 38.37137
```

```
mean(Healthy_cereal[Healthy_cereal$c_d6==3,"rating"])
```

```
## [1] 28.66112
```

```
mean(Healthy_cereal[Healthy_cereal$c_d6==4,"rating"])
```

```
## [1] 46.17608
```

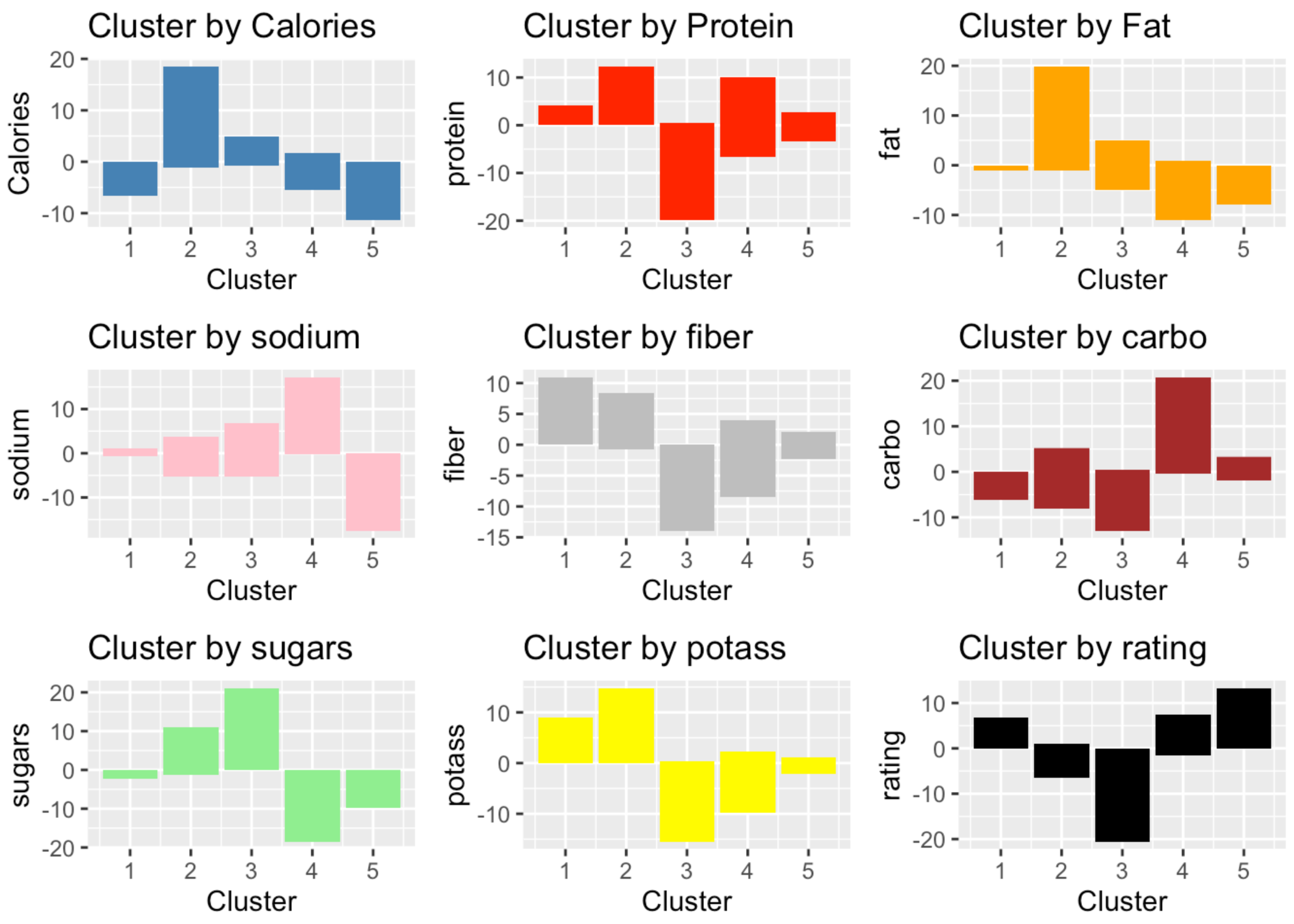
```
mean(Healthy_cereal[Healthy_cereal$c_d6==5,"rating"])
```

```
## [1] 63.0184
```

#It is evident that Cluster1 has the highest rating (73.84446), so we will select it as a nutritious cereal.

#lets also visualize the results by plotting a bar chart

```
calories <- ggplot(Clustered_df, aes(x = c_d6, y = calories)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(x = "Cluster", y = "Calories") +
  ggtitle("Cluster by Calories")

protein <- ggplot(Clustered_df, aes(x = c_d6, y = protein)) +
  geom_bar(stat = "identity", fill = "red") +
  labs(x = "Cluster", y = "protein") +
  ggtitle("Cluster by Protein")

fat <- ggplot(Clustered_df, aes(x = c_d6, y = fat)) +
  geom_bar(stat = "identity", fill = "orange") +
  labs(x = "Cluster", y = "fat") +
  ggtitle("Cluster by Fat")

sodium <- ggplot(Clustered_df, aes(x = c_d6, y = sodium)) +
  geom_bar(stat = "identity", fill = "pink") +
  labs(x = "Cluster", y = "sodium") +
  ggtitle("Cluster by sodium")

fiber <- ggplot(Clustered_df, aes(x = c_d6, y = fiber)) +
  geom_bar(stat = "identity", fill = "gray") +
  labs(x = "Cluster", y = "fiber") +
  ggtitle("Cluster by fiber")

carbo <- ggplot(Clustered_df, aes(x = c_d6,, y = carbo)) +
  geom_bar(stat = "identity", fill = "brown") +
  labs(x = "Cluster", y = "carbo") +
  ggtitle("Cluster by carbo")

sugars <- ggplot(Clustered_df, aes(x = c_d6,, y = sugars)) +
  geom_bar(stat = "identity", fill = "lightgreen") +
  labs(x = "Cluster", y = "sugars") +
  ggtitle("Cluster by sugars")

potass <- ggplot(Clustered_df, aes(x = c_d6,, y = potass)) +
  geom_bar(stat = "identity", fill = "yellow") +
  labs(x = "Cluster", y = "potass") +
  ggtitle("Cluster by potass")

rating <- ggplot(Clustered_df, aes(x = c_d6,, y = rating)) +
  geom_bar(stat = "identity", fill = "black") +
  labs(x = "Cluster", y = "rating") +
  ggtitle("Cluster by rating")
plot_grid(calories, protein, fat, sodium, fiber, carbo, sugars, potass, rating)
```



#

Here we can see that cluster 1 still has the best results. It is low in calories, sugar, and fat. It has higher content of fiber, potassium and protein. Thus we can conclude that cluster 1 can be a set of cereals to include in their daily cafeterias.