Assignment_5

Niharika

2024-04-07

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

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

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

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

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.3.3
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.3
```

```
## Warning: package 'readr' was built under R version 4.3.3
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
## Warning: package 'forcats' was built under R version 4.3.3
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ lubridate 1.9.3     ✔ tibble    3.2.1
## ✔ purrr     1.0.2     ✔ tidyr     1.3.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ purrr::lift()   masks caret::lift()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

```
library(ISLR)
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.3.3
```

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

```
library(dbscan)
```

```
## Warning: package 'dbscan' was built under R version 4.3.3
```

```
## 
## Attaching package: 'dbscan'
## 
## The following object is masked from 'package:stats':
## 
##     as.dendrogram
```

```
library(cluster)
```

```
## Warning: package 'cluster' was built under R version 4.3.3
```

```
library(klustR)
```

```
## Warning: package 'klustR' was built under R version 4.3.3
```

```
library(ggplot2)
library(dplyr)
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 4.3.3
```

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

```
cereals.data <- read.csv("C:/Users/nihar/Downloads/Cereals.csv")

# Omit the null values
cereals.data <- na.omit(cereals.data)

# check the dimensions of the dataset
dim(cereals.data)
```

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

```
head(cereals.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
## 6    Apple_Cinnamon_Cheerios   G    C      110       2   2    180   1.5  10.5
## 7                Apple_Jacks   K    C      110       2   0    125   1.0  11.0
##   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
## 6     10     70       25     1      1 0.75 29.50954
## 7     14     30       25     2      1 1.00 33.17409
```

```
# To know the column names of the dataset
t(t(names(cereals.data)))# The 't' function creates a transpose of the dataframe
```

```
##       [,1]
##  [1,] "name"
##  [2,] "mfr"
##  [3,] "type"
##  [4,] "calories"
##  [5,] "protein"
##  [6,] "fat"
##  [7,] "sodium"
##  [8,] "fiber"
##  [9,] "carbo"
## [10,] "sugars"
## [11,] "potass"
## [12,] "vitamins"
## [13,] "shelf"
## [14,] "weight"
## [15,] "cups"
## [16,] "rating"
```

Assignment Task A #“Apply hierarchical clustering to the data using Euclidean distance to the normalized measurements. Use Agnes to compare the clustering from single linkage, complete linkage, average linkage, and Ward. Choose the best method.”

```
clust.data <- cereals.data[ ,4:16]
dim(clust.data)
```

```
## [1] 74 13
```

```
head(clust.data)
```

```
##   calories protein fat sodium fiber carbo sugars potass vitamins shelf weight
## 1       70       4   1    130  10.0   5.0      6    280       25     3      1
## 2      120       3   5     15   2.0   8.0      8    135        0     3      1
## 3       70       4   1    260   9.0   7.0      5    320       25     3      1
## 4       50       4   0    140  14.0   8.0      0    330       25     3      1
## 6      110       2   2    180   1.5  10.5     10     70       25     1      1
## 7      110       2   0    125   1.0  11.0     14     30       25     2      1
##   cups   rating
## 1 0.33 68.40297
## 2 1.00 33.98368
## 3 0.33 59.42551
## 4 0.50 93.70491
## 6 0.75 29.50954
## 7 1.00 33.17409
```

```
summary(clust.data)
```

```
##     calories      protein          fat        sodium          fiber       
##  Min.   : 50   Min.   :1.000   Min.   :0   Min.   :  0.0   Min.   : 0.000  
##  1st Qu.:100   1st Qu.:2.000   1st Qu.:0   1st Qu.:135.0   1st Qu.: 0.250  
##  Median :110   Median :2.500   Median :1   Median :180.0   Median : 2.000  
##  Mean   :107   Mean   :2.514   Mean   :1   Mean   :162.4   Mean   : 2.176  
##  3rd Qu.:110   3rd Qu.:3.000   3rd Qu.:1   3rd Qu.:217.5   3rd Qu.: 3.000  
##  Max.   :160   Max.   :6.000   Max.   :5   Max.   :320.0   Max.   :14.000  
##      carbo           sugars           potass          vitamins     
##  Min.   : 5.00   Min.   : 0.000   Min.   : 15.00   Min.   :  0.00  
##  1st Qu.:12.00   1st Qu.: 3.000   1st Qu.: 41.25   1st Qu.: 25.00  
##  Median :14.50   Median : 7.000   Median : 90.00   Median : 25.00  
##  Mean   :14.73   Mean   : 7.108   Mean   : 98.51   Mean   : 29.05  
##  3rd Qu.:17.00   3rd Qu.:11.000   3rd Qu.:120.00   3rd Qu.: 25.00  
##  Max.   :23.00   Max.   :15.000   Max.   :330.00   Max.   :100.00  
##      shelf           weight           cups            rating     
##  Min.   :1.000   Min.   :0.500   Min.   :0.2500   Min.   :18.04  
##  1st Qu.:1.250   1st Qu.:1.000   1st Qu.:0.6700   1st Qu.:32.45  
##  Median :2.000   Median :1.000   Median :0.7500   Median :40.25  
##  Mean   :2.216   Mean   :1.031   Mean   :0.8216   Mean   :42.37  
##  3rd Qu.:3.000   3rd Qu.:1.000   3rd Qu.:1.0000   3rd Qu.:50.52  
##  Max.   :3.000   Max.   :1.500   Max.   :1.5000   Max.   :93.70  
```

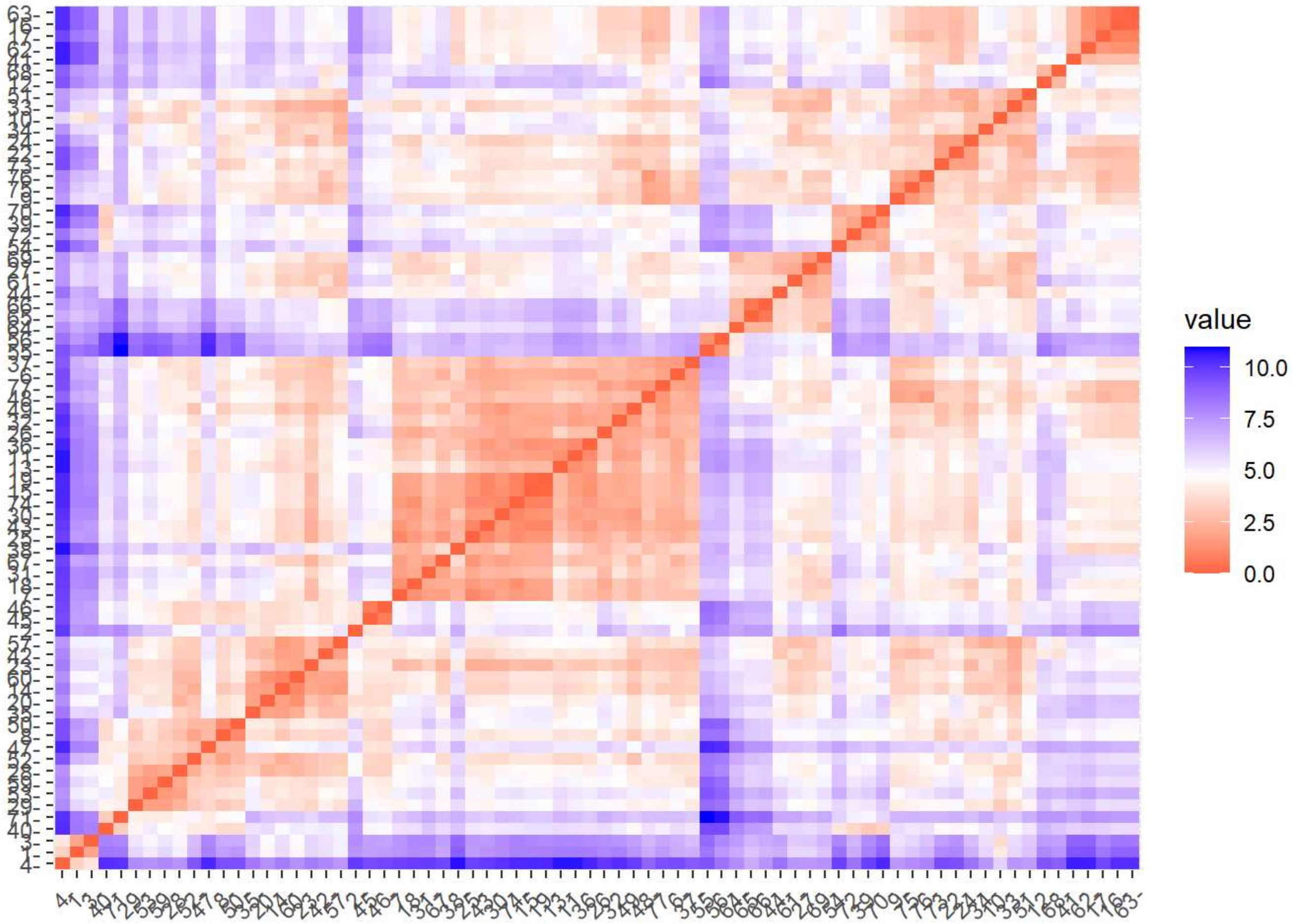
```
scaled.data <- scale(clust.data)
head(scaled.data)
```

```
##     calories   protein        fat     sodium       fiber      carbo     sugars
## 1 -1.8659155  1.3817478  0.0000000 -0.3910227  3.22866747 -2.5001396 -0.2542051
## 2  0.6537514  0.4522084  3.9728810 -1.7804186 -0.07249167 -1.7292632  0.2046041
## 3 -1.8659155  1.3817478  0.0000000  1.1795987  2.81602258 -1.9862220 -0.4836096
## 4 -2.8737823  1.3817478 -0.9932203 -0.2702057  4.87924705 -1.7292632 -1.6306324
## 6  0.1498180 -0.4773310  0.9932203  0.2130625 -0.27881412 -1.0868662  0.6634132
## 7  0.1498180 -0.4773310 -0.9932203 -0.4514312 -0.48513656 -0.9583868  1.5810314
##       potass   vitamins      shelf     weight       cups     rating
## 1  2.5605229 -0.1818422  0.9419715 -0.2008324 -2.0856582  1.8549038
## 2  0.5147738 -1.3032024  0.9419715 -0.2008324  0.7567534 -0.5977113
## 3  3.1248675 -0.1818422  0.9419715 -0.2008324 -2.0856582  1.2151965
## 4  3.2659536 -0.1818422  0.9419715 -0.2008324 -1.3644493  3.6578436
## 6 -0.4022862 -0.1818422 -1.4616799 -0.2008324 -0.3038480 -0.9165248
## 7 -0.9666308 -0.1818422 -0.2598542 -0.2008324  0.7567534 -0.6553998
```

```
dim(scaled.data)
```

```
## [1] 74 13
```

```
distance <- get_dist(scaled.data)
fviz_dist(distance)
```



```
# Consider the Euclidean distance to know the distance between each point
dist <- dist(scaled.data, method = "euclidean")

# cluster using different linkage methods
# Single linkage
hc_single <- agnes(dist,method = "single")

# Complete linkage
hc_complete <- agnes(dist,method = "complete")

# average linkage
hc_average <- agnes(dist,method = "average")

# Ward's linkage
hc_ward <- agnes(dist,method = "ward")

#comparing the above links values with Agglomerative coefficient(ac)
print(hc_single$ac)
```

```
## [1] 0.6067859
```

```
print(hc_complete$ac)
```

```
## [1] 0.8353712
```

```
print(hc_average$ac)
```

```
## [1] 0.7766075
```

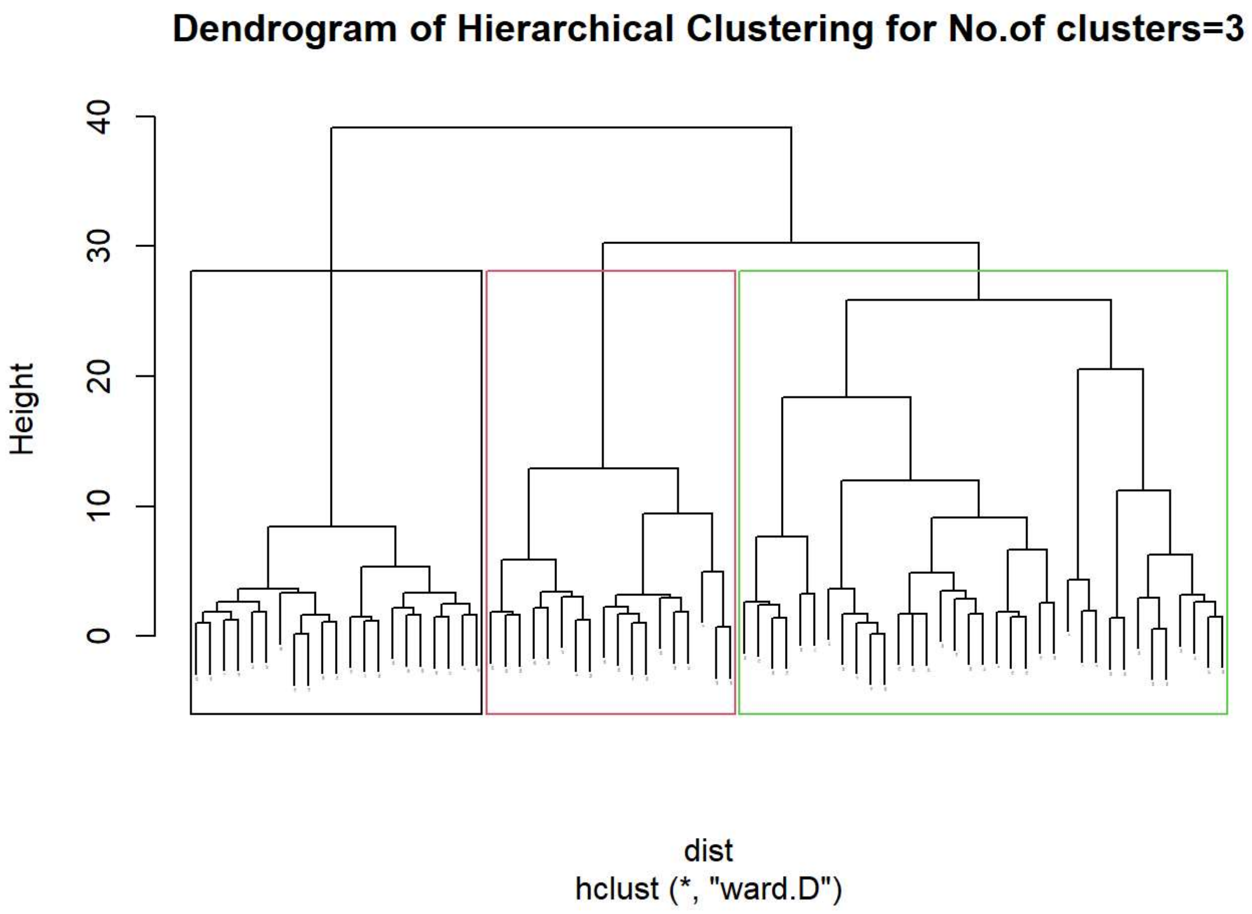
```
print(hc_ward$ac)
```

```
## [1] 0.9046042
```

From the output we can see that the agglomerative coefficient for wards method is high. so we need to consider that.

Assignment Task B #“How many clusters would you choose?” I choose (3-8)

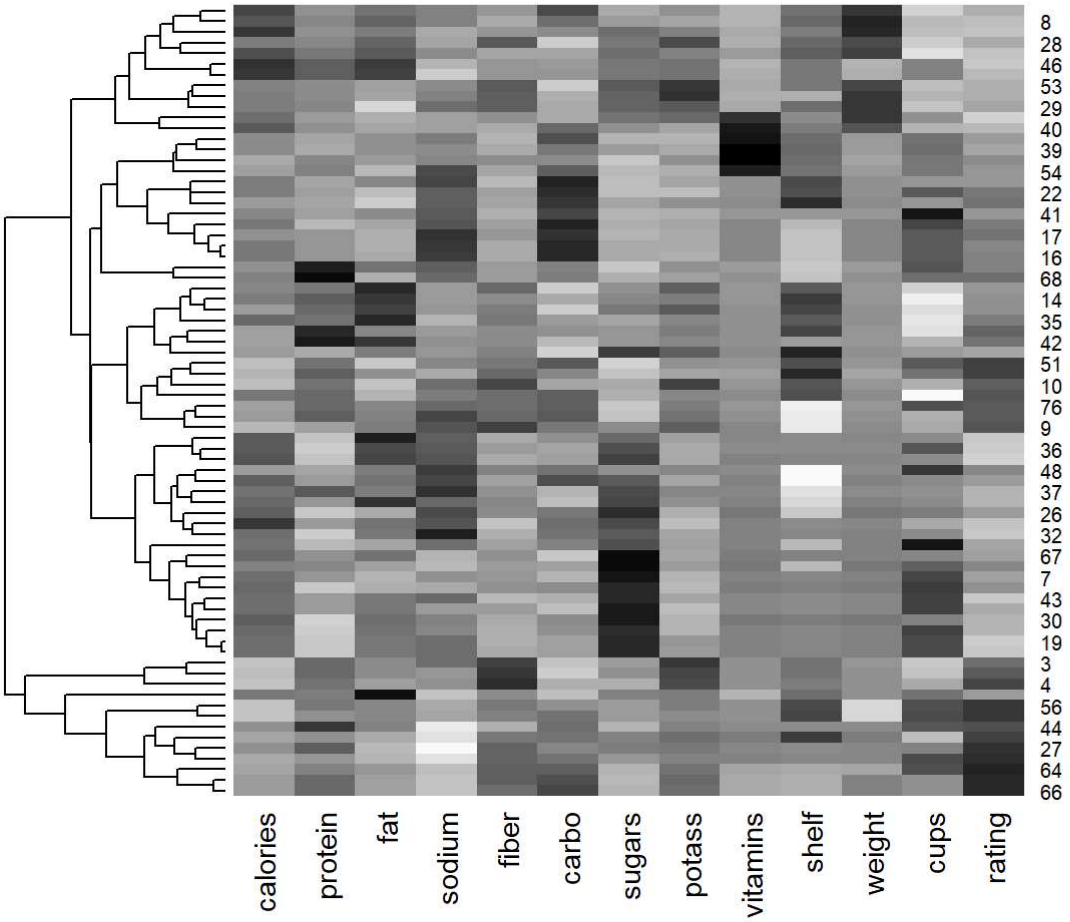
```
# Cluster the data using Hierarchical clustering
hc3 <- hclust(dist, method ="ward.D")

# Plot the dendrogram
plot(hc3, cex = 0.1, main = "Dendrogram of Hierarchical Clustering for No.of clusters=3")

# Split the tree
data3 = rect.hclust(hc3, k=3 , border = 1:3)
```



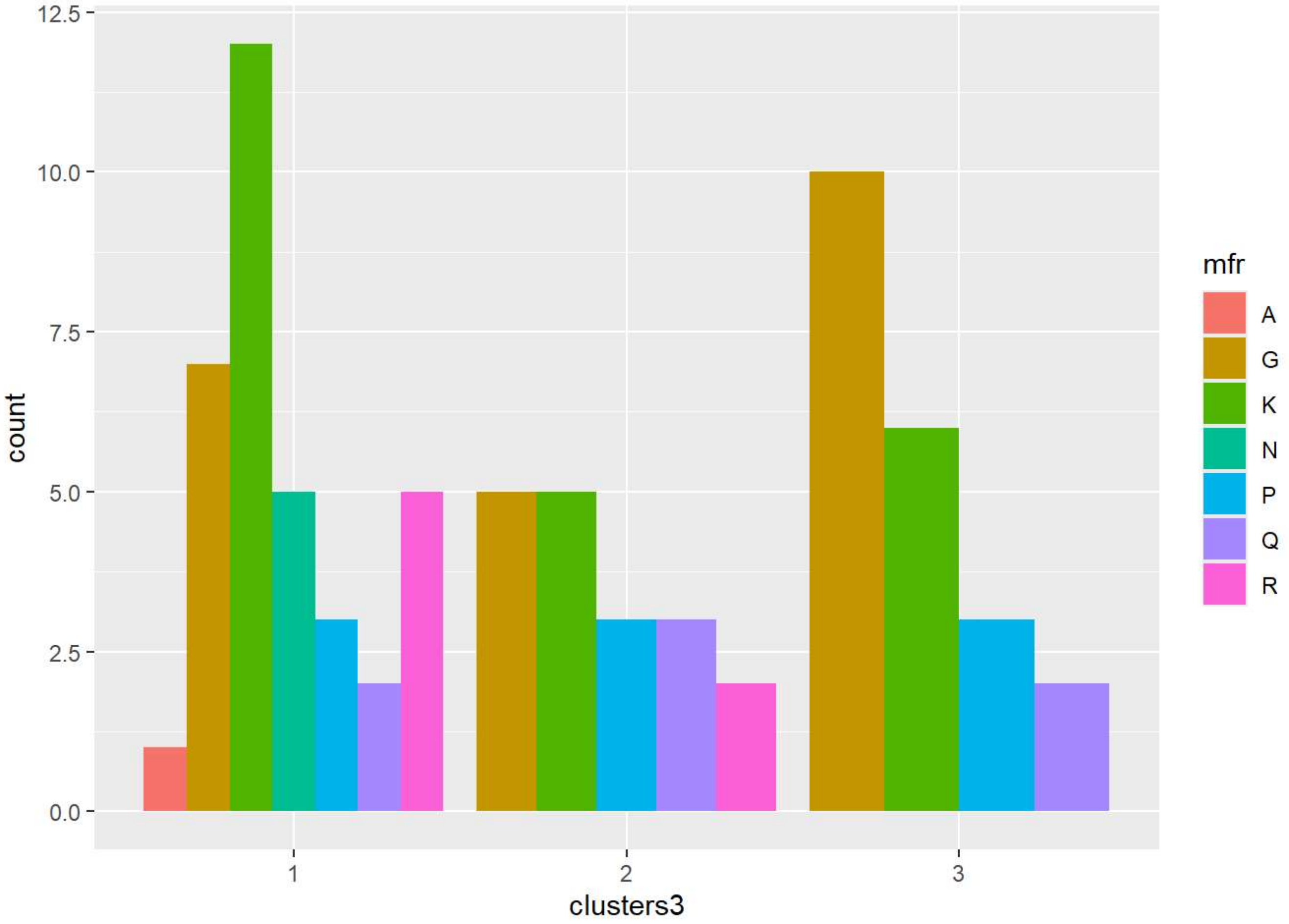
```
clusters3 <- cutree(hc3, k=3)

# To know the cluster size
data_with_clusters3 <- cbind(Cluster = data3)
data_with_clusters3
```

```
##      Cluster
## [1,] integer,21
## [2,] integer,18
## [3,] integer,35
```

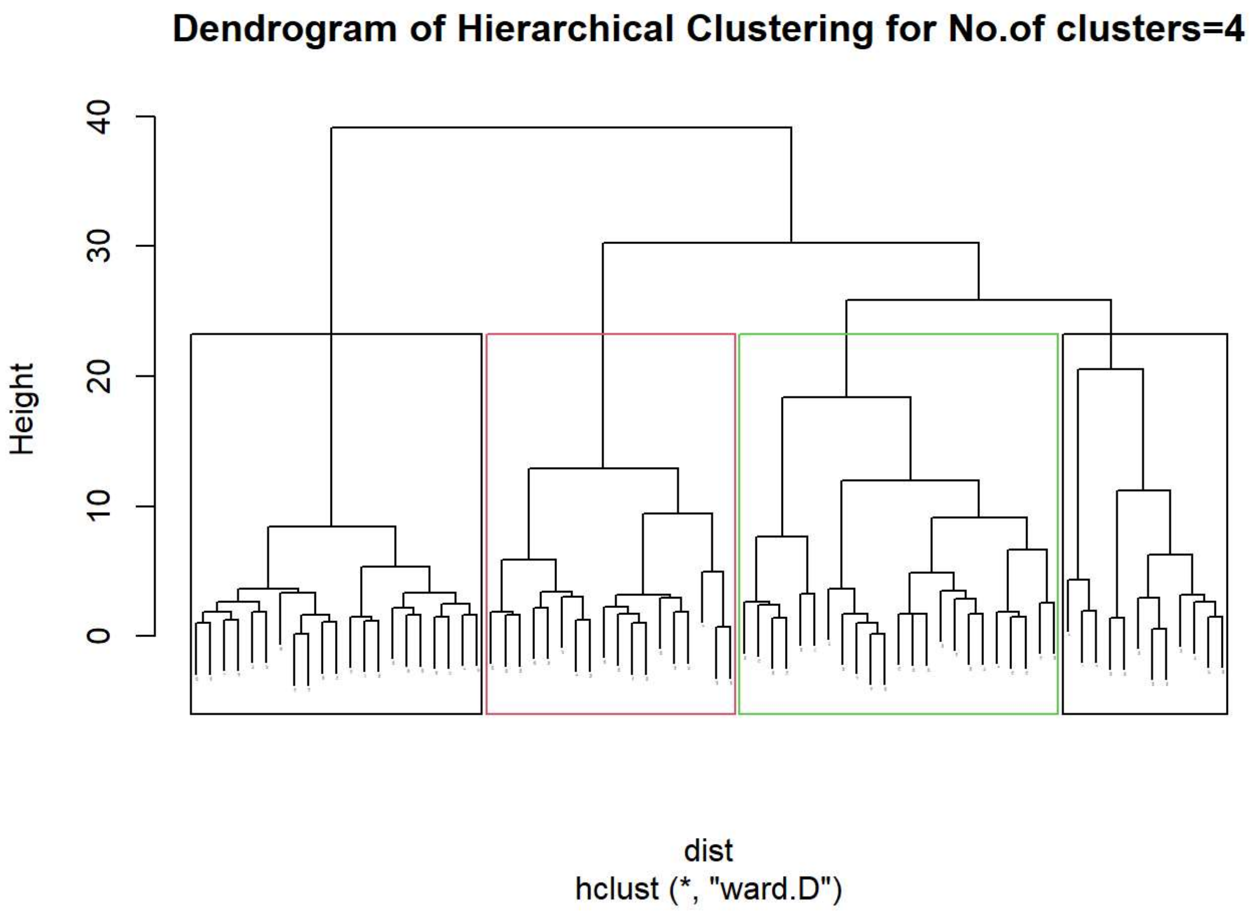
```
# add the clusters to the table
clustered <- cbind(cereals.data, clusters3)







# heatmap for the data
heatmap(as.matrix(scaled.data), Colv = NA, hclustfun = hclust,
        col=rev(paste("gray",1:99,sep="")))
```



```
# plot the data
ggplot(clustered, aes(factor(clusters3), fill= mfr))+ geom_bar(position='dodge')+labs(x='clusters3')
```



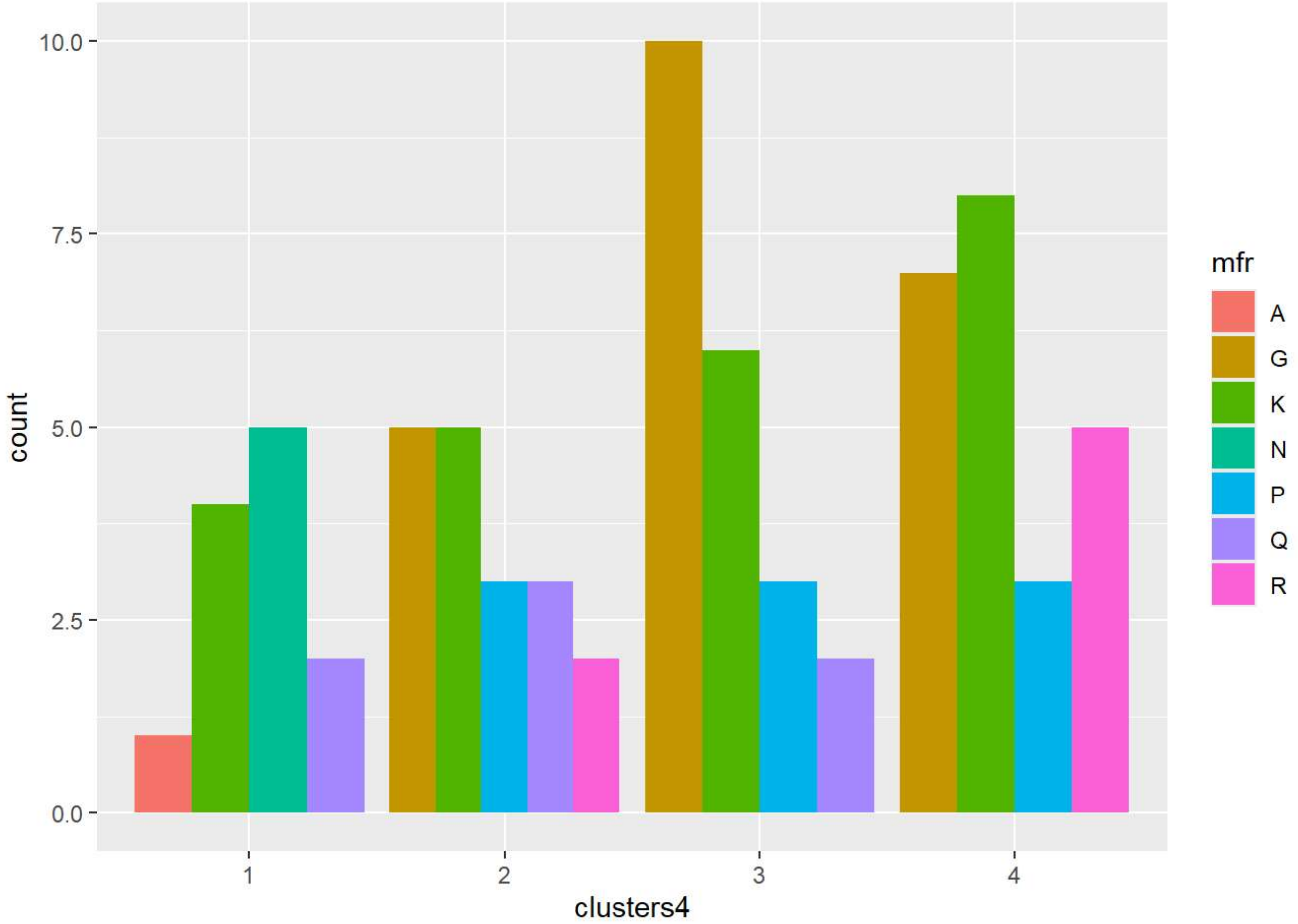
```
# Cluster the data using Hierarchical clustering
hc4 <- hclust(dist, method ="ward.D")

# Plot the dendrogram
plot(hc4, cex = 0.1, main = "Dendrogram of Hierarchical Clustering for No.of clusters=4")

# Split the tree
data4 = rect.hclust(hc4, k=4 , border = 1:3)
```



```
clusters4 <- cutree(hc4, k=4)

# To know the cluster size
data_with_clusters <- cbind(Cluster = data4)
data_with_clusters
```

```
##      Cluster
## [1,] integer,21
## [2,] integer,18
## [3,] integer,23
## [4,] integer,12
```

```
# add the clusters to the table
clustered <- cbind(cereals.data, clusters4)

# plot the graph
ggplot(clustered, aes(factor(clusters4), fill= mfr))+ geom_bar(position='dodge')+labs(x='cluster
s4')
```



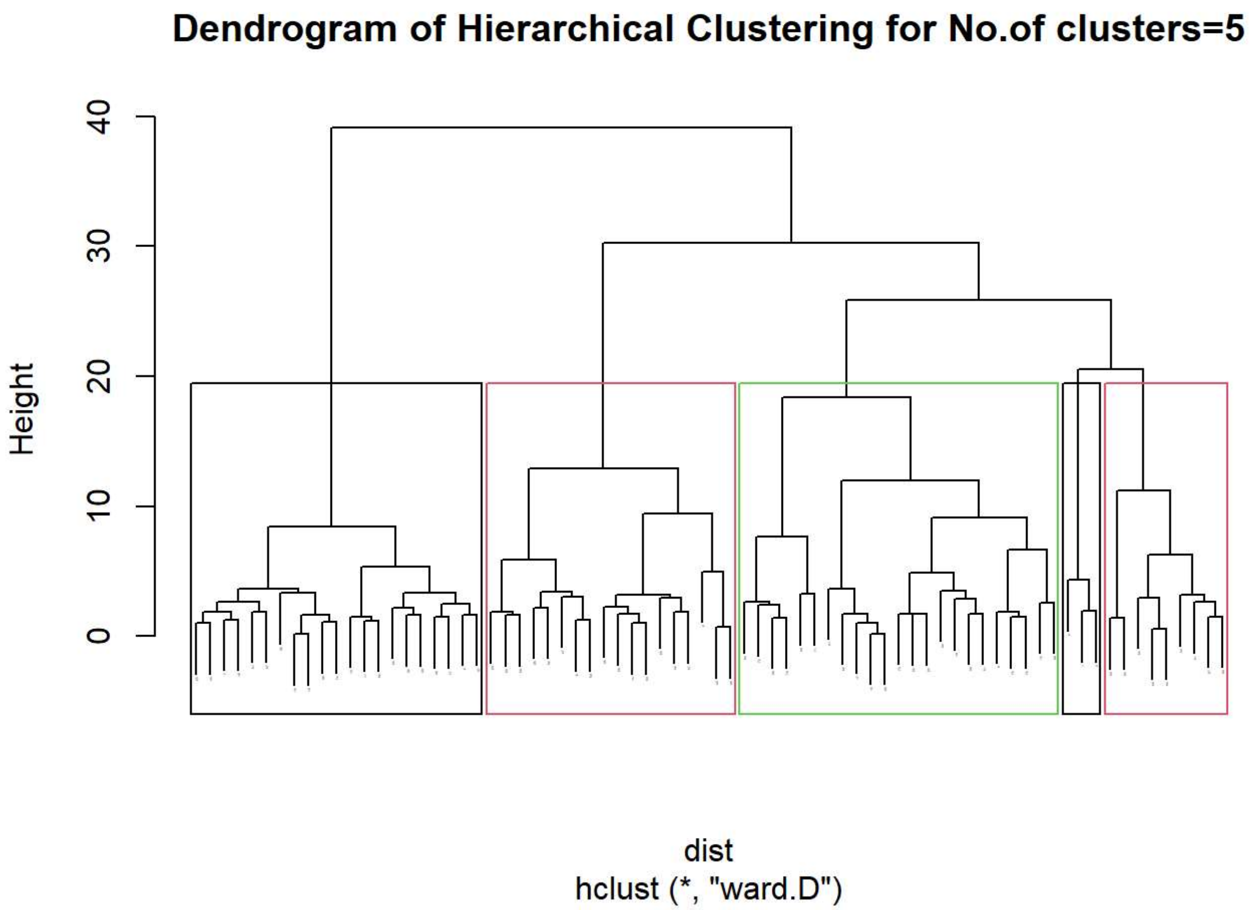
Assignment C #“Comment on the structure of the clusters and on their stability. Hint: To check stability, partition the data and see how well clusters formed based on one part apply to the other part. To do this:

Ans.To check stability of clusters, the data set will be split into a 70/30 partition. The 70% will be used to create cluster assignments again, and then the remaining 30% will be assigned based on their closest centroid.

```
set.seed(111780)
# Split the data into 70% partition A and 30% partition B
cerealIndex <- createDataPartition(cereals.data$protein, p=0.3, list =
F)
cereal_preprocessed_PartitionB <- cereals.data[cerealIndex, ]
cereal_preprocessed_PartitionA <- cereals.data[-cerealIndex,]
```

#1. Cluster partition A #2. Use the cluster centroids from A to assign each record in partition B (each record is assigned to the cluster with the closest centroid). #3. Assess how consistent the cluster assignments are compared to the assignments based on all the data”

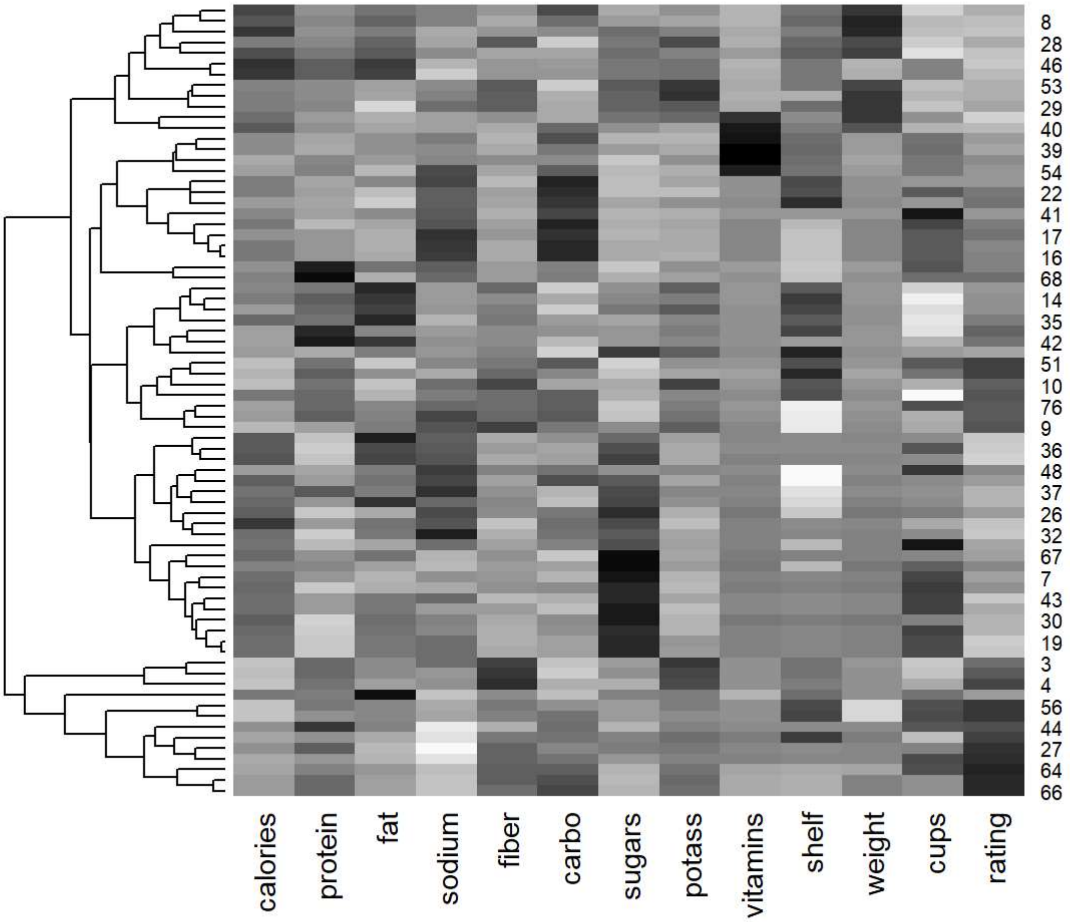
```
# Cluster the data using Hierarchical clustering
hc5 <- hclust(dist, method ="ward.D")

# Plot the dendrogram
plot(hc5, cex = 0.1, main = "Dendrogram of Hierarchical Clustering for No.of clusters=5")

# Split the tree
data5 = rect.hclust(hc5, k=5 , border = 1:3)
```



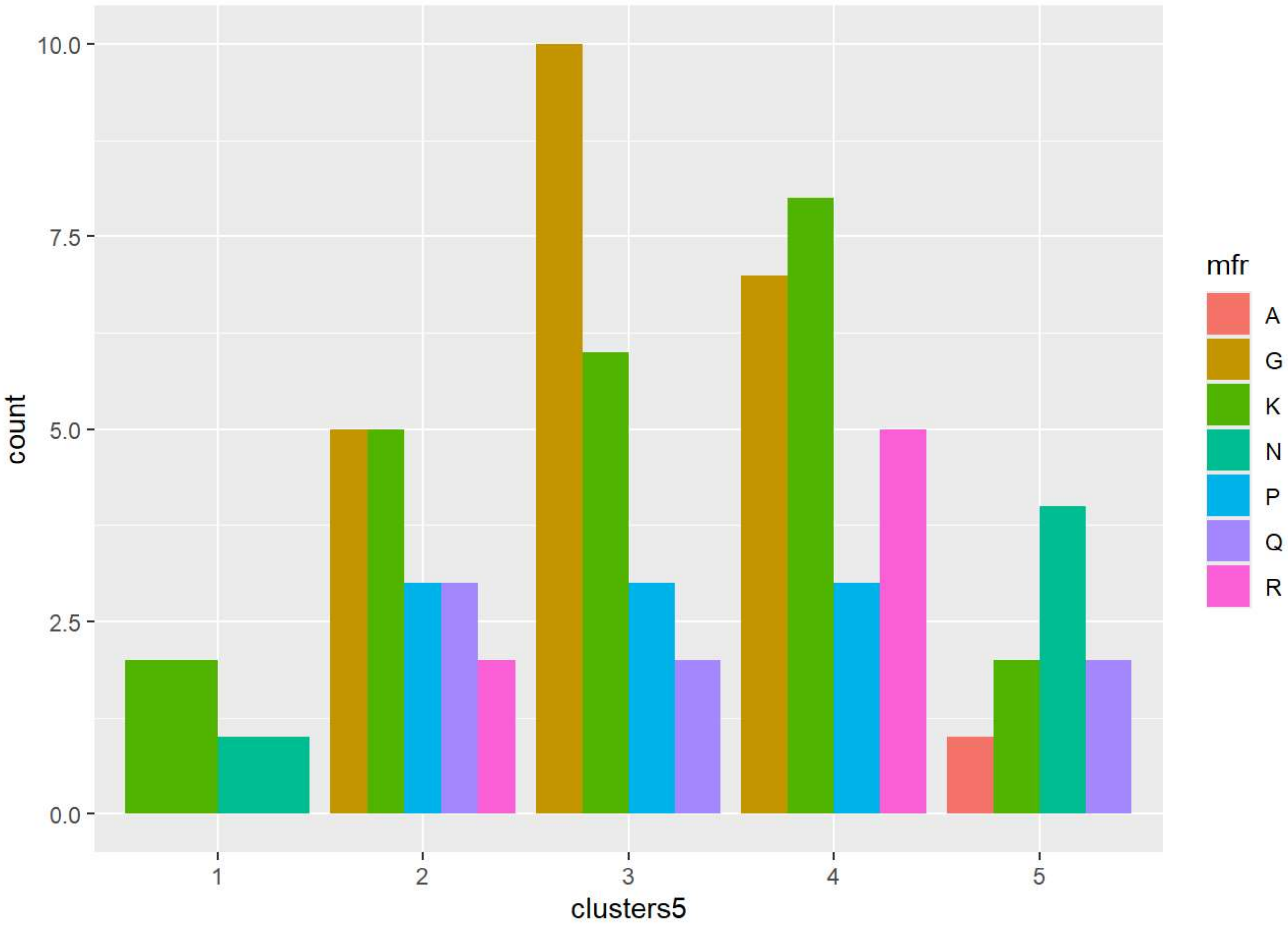
```
clusters5 <- cutree(hc5, k=5)

# To know the cluster size
data_with_clusters <- cbind(Cluster = data5)
data_with_clusters
```

```
##      Cluster
## [1,] integer,21
## [2,] integer,18
## [3,] integer,23
## [4,] integer,3
## [5,] integer,9
```

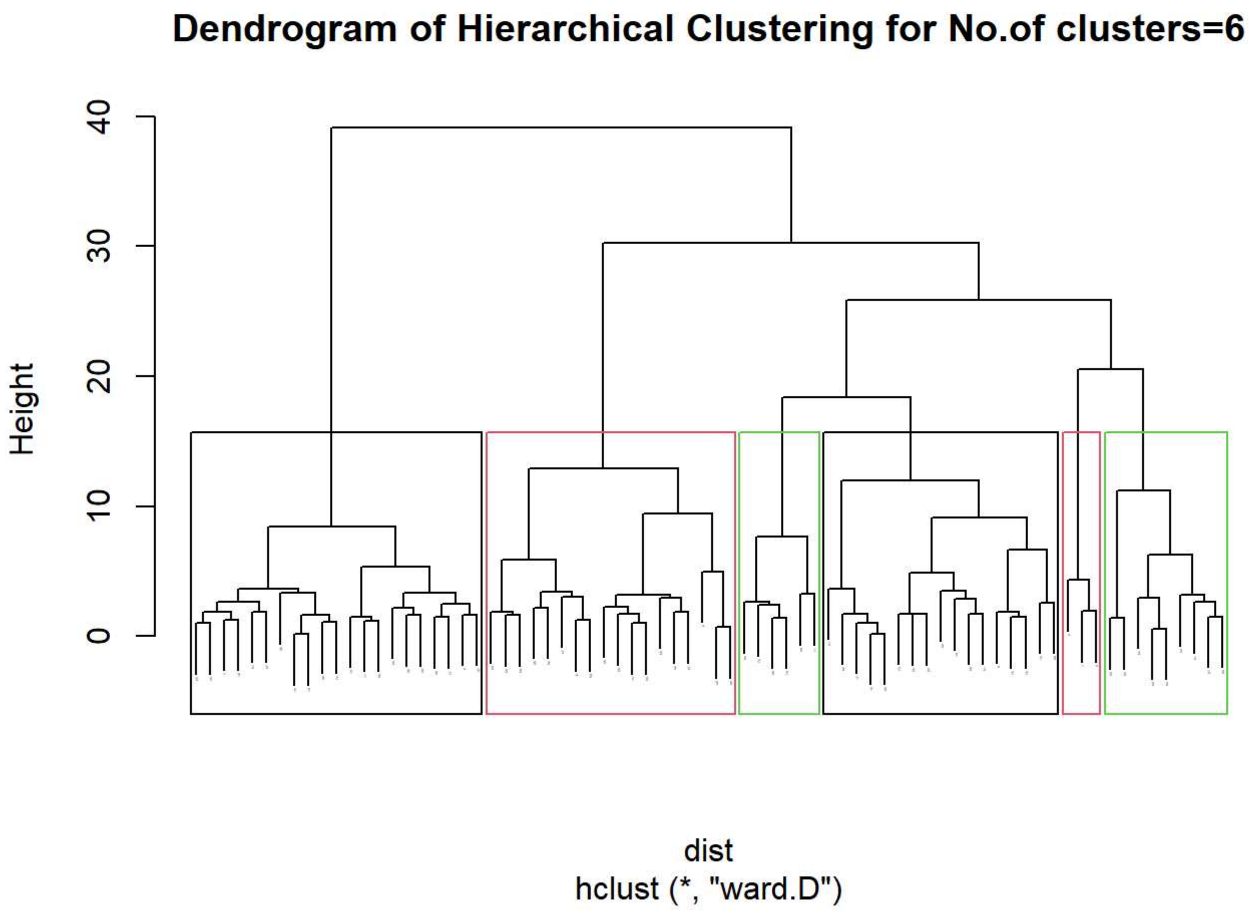
```
# add the data
clustered <- cbind(cereals.data, clusters5)



# heatmap for the data
heatmap(as.matrix(scaled.data), Colv = NA, hclustfun = hclust,
        col=rev(paste("gray",1:99,sep="")))
```



```
# plot the data
ggplot(clustered, aes(factor(clusters5), fill= mfr))+ geom_bar(position='dodge')+labs(x='clusters5')
```



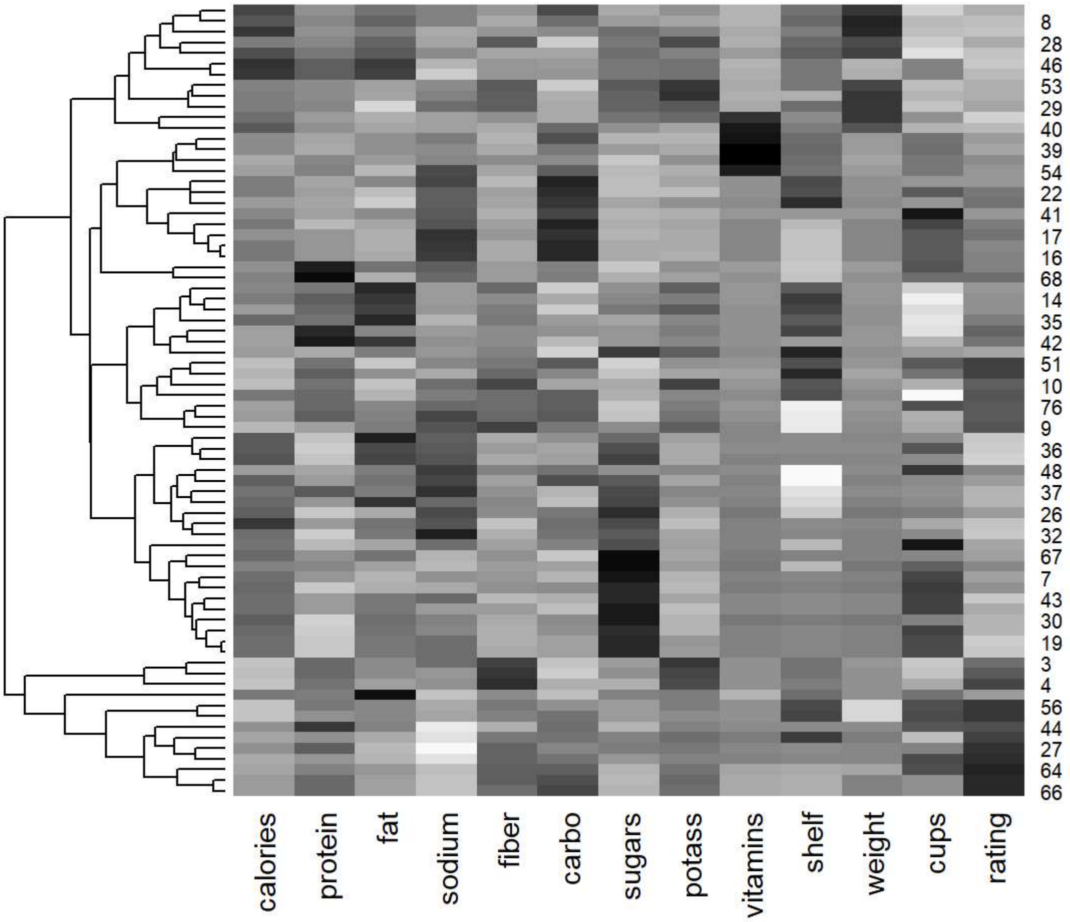
```
# Cluster the data using Hierarchical clustering
hc6 <- hclust(dist, method ="ward.D")

# Plot the dendrogram
plot(hc6, cex = 0.1, main = "Dendrogram of Hierarchical Clustering for No.of clusters=6")

# Split the tree
data6 = rect.hclust(hc6, k=6 , border = 1:3)
```



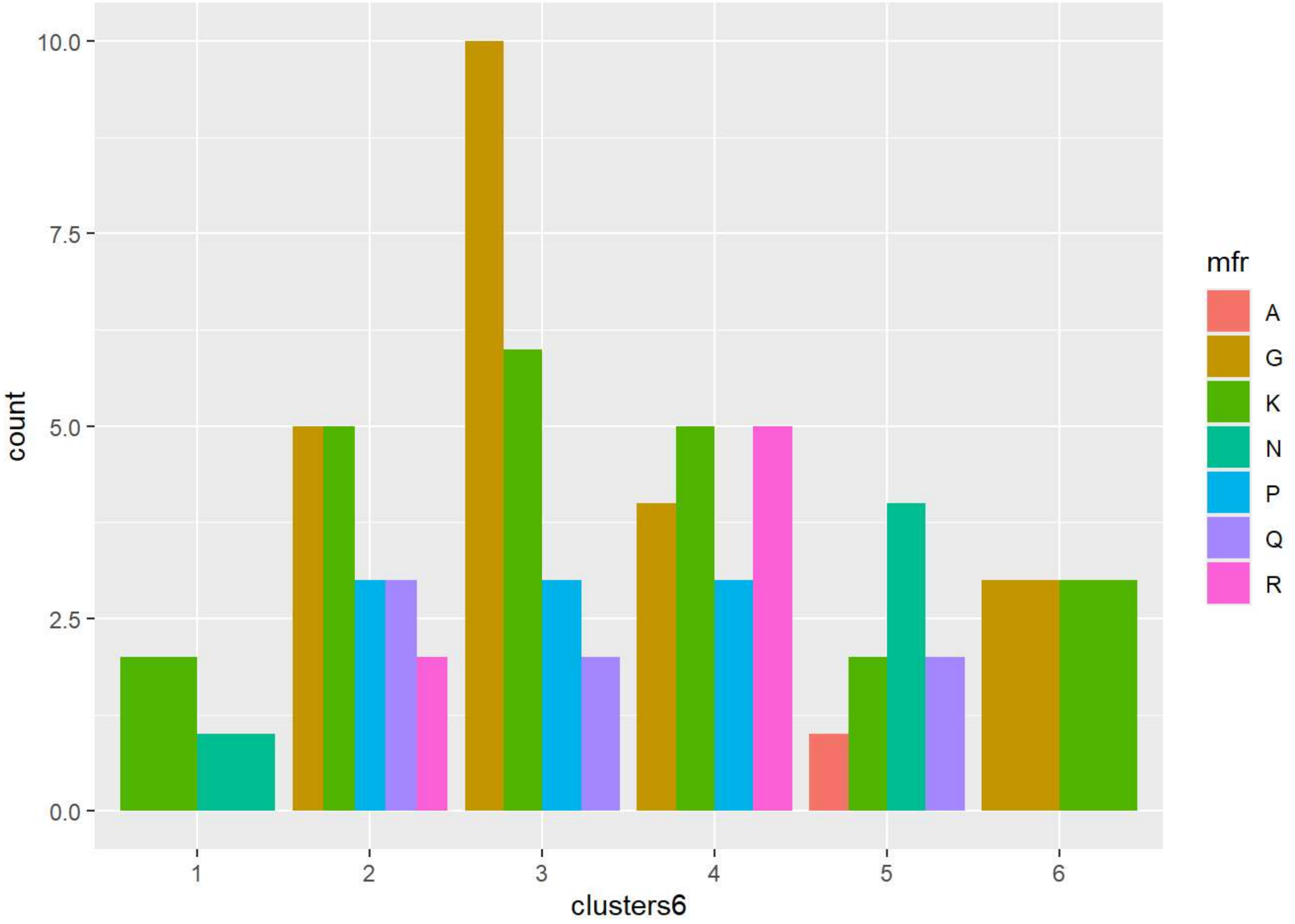
```
clusters6 <- cutree(hc6, k=6)

# To know the cluster size
data_with_clusters <- cbind(Cluster = data6)
data_with_clusters
```

```
##      Cluster
## [1,] integer,21
## [2,] integer,18
## [3,] integer,6
## [4,] integer,17
## [5,] integer,3
## [6,] integer,9
```

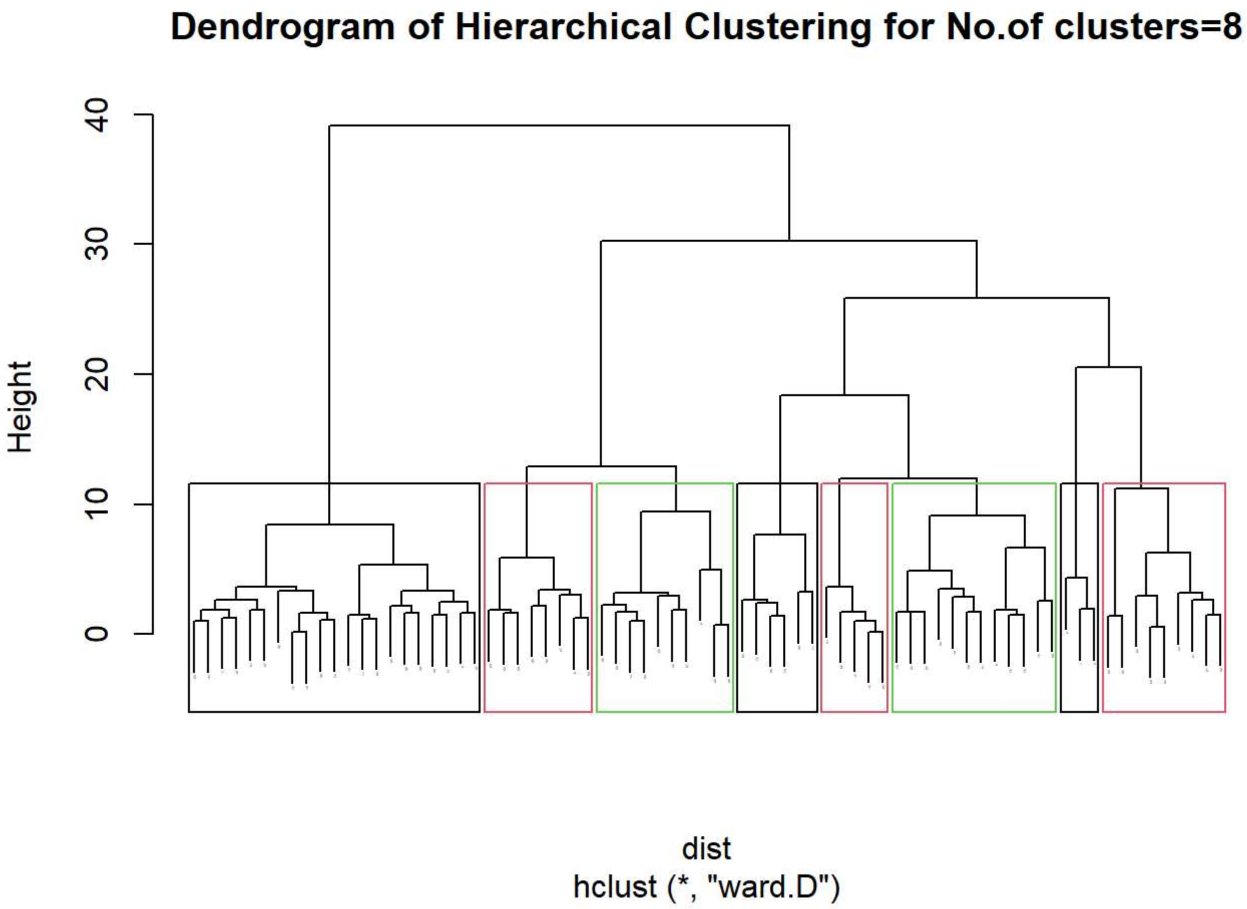
```
# add the data
clustered <- cbind(cereals.data, clusters6)



# heatmap for the data
heatmap(as.matrix(scaled.data), Colv = NA, hclustfun = hclust,
        col=rev(paste("gray",1:99,sep="")))
```



```
# plot the data
ggplot(clustered, aes(factor(clusters6), fill= mfr))+ geom_bar(position='dodge')+labs(x='clusters6')
```



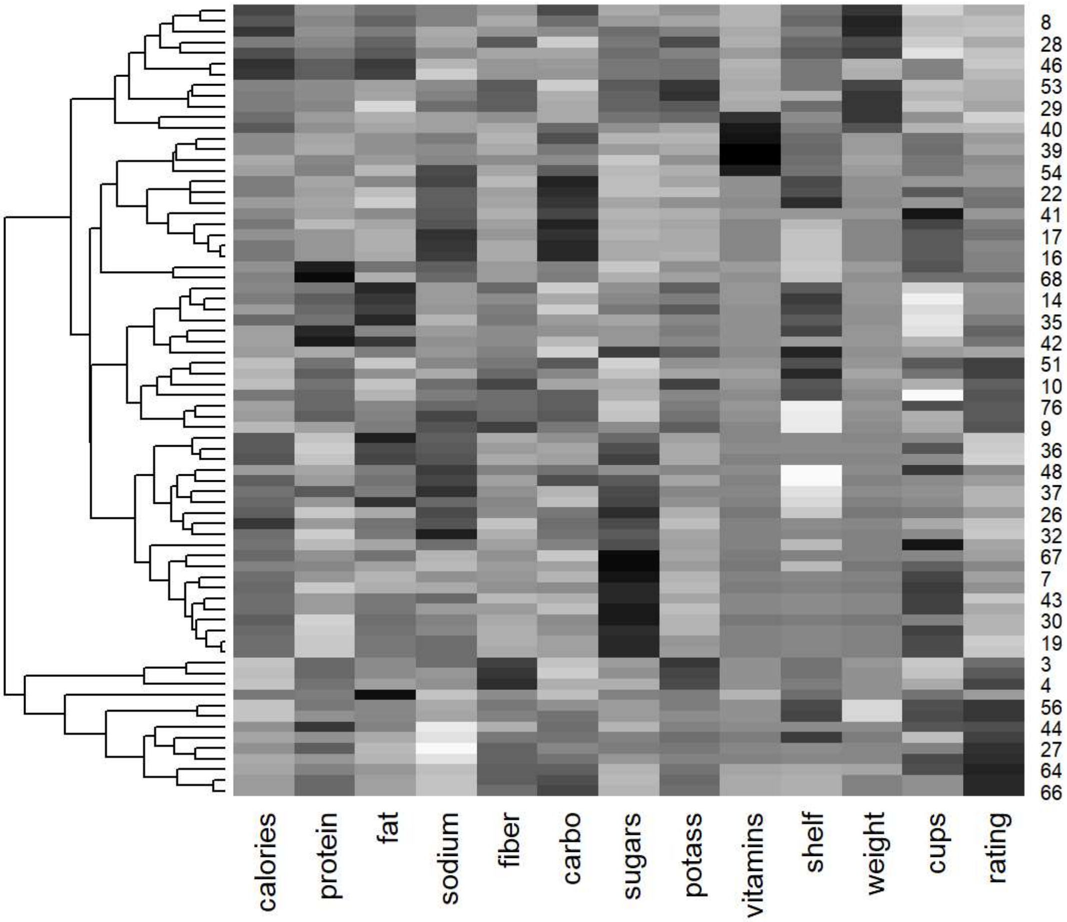
```
# Cluster the data using Hierarchical clustering
hc8 <- hclust(dist, method ="ward.D")

# Plot the dendrogram
plot(hc8, cex = 0.1, main = "Dendrogram of Hierarchical Clustering for No.of clusters=8")

# Split the tree
data8 = rect.hclust(hc8, k=8 , border = 1:3)
```



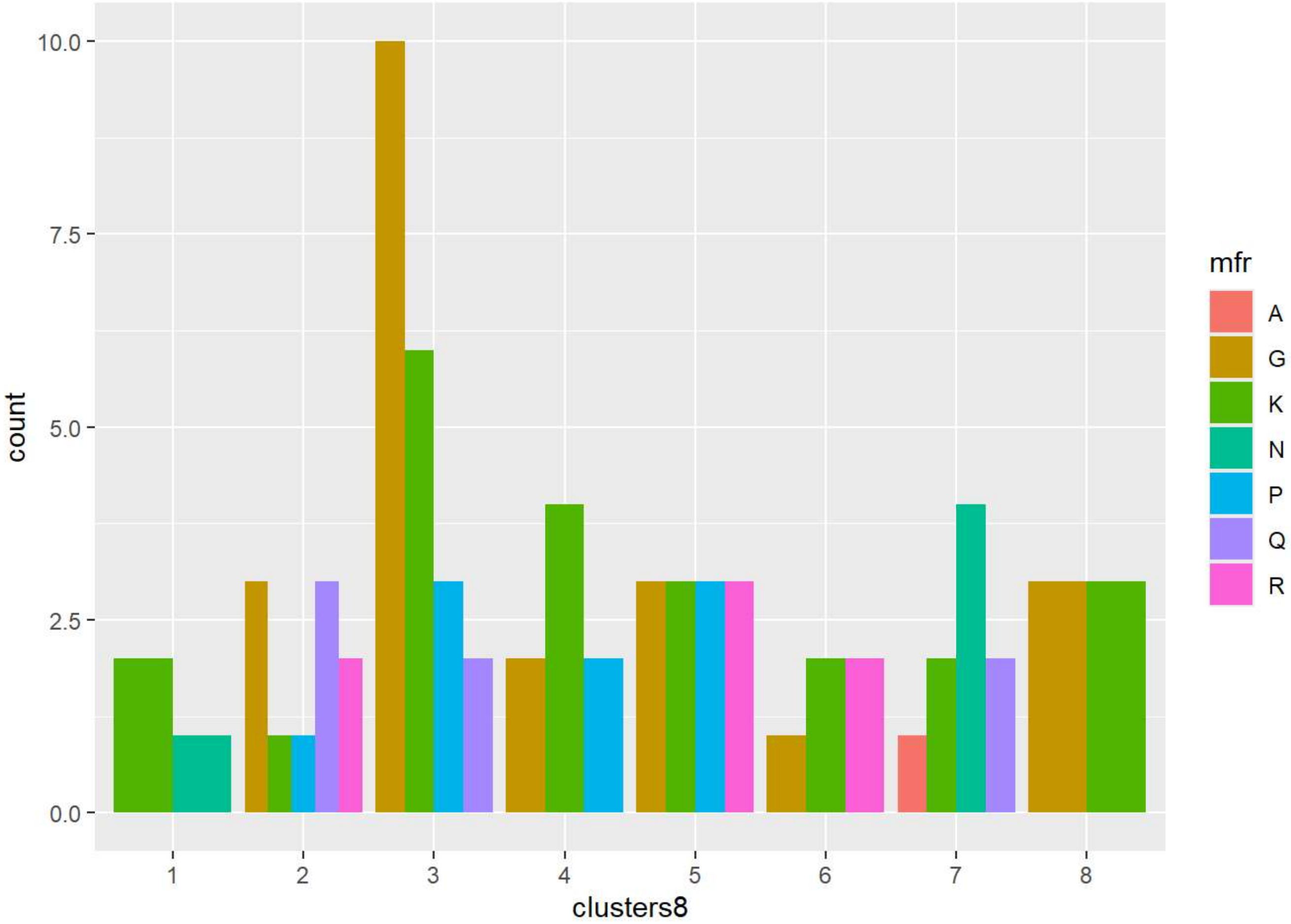
```
clusters8 <- cutree(hc8, k=8)

# To know the cluster size
data_with_clusters <- cbind(Cluster = data8)
data_with_clusters
```

```
##      Cluster
## [1,] integer,21
## [2,] integer,8
## [3,] integer,10
## [4,] integer,6
## [5,] integer,5
## [6,] integer,12
## [7,] integer,3
## [8,] integer,9
```

```
# add the data
clustered <- cbind(cereals.data, clusters8)



# heatmap for the data
heatmap(as.matrix(scaled.data), Colv = NA, hclustfun = hclust,
        col=rev(paste("gray",1:99,sep="")))
```



```
# plot the data
ggplot(clustered, aes(factor(clusters8), fill= mfr))+ geom_bar(position='dodge')+labs(x='cluster
s8')
```



Assignment Task D

In the context of selecting cereals for elementary school cafeterias to ensure a healthy diet, it’s important to consider the normalization of data. Normalizing the data may not be appropriate because it would scale the nutritional information based on the sample of cereals being analyzed. This approach could be misleading because the dataset might include cereals with extremely high sugar content and very low fiber, iron, and other essential nutrients. When normalized across the sample set, it becomes difficult to interpret the nutritional value accurately. For instance, a cereal with a normalized value of 0.999 for iron might appear to have nearly all the nutritional iron a child needs, but it could simply be the best among a set of unhealthy options. A more suitable approach for preprocessing the data would be to express it as a ratio to the daily recommended intake of calories, fiber, carbohydrates, etc., for a child. This method allows analysts to make better-informed decisions about clustering without allowing a few variables with larger values to dominate the distance calculations. By reviewing the clusters, analysts can assess the average nutritional values within each cluster to determine what percentage of a student’s daily recommended nutrition would come from each cereal. This approach enables staff to make informed decisions about selecting “healthy” cereals for the cafeteria.