FML5

Varsha Karunya Mekala

2023-11-27

#Let us load all the required packages

#Now let us import the cereals data

```
getwd()
```

```
## [1] "/Users/varshamekala/Desktop"
```

```
setwd("/Users/varshamekala/Desktop/assignments")
cereals_data = read.csv('Cereals.csv')
```

#Data Preprocessing. Remove all the cereals with missing values

```
# assigning rows to the cereal names
rownames(cereals_data)=cereals_data$name

# Remove the cereal name column
cereals_data = cereals_data[,-1]

# The data contains three category variables: shelf, kind, and mfr. Removing
them
cereals_data = cereals_data[,c(-1,-2,-12)]

# Normalization
normalized_cereals_data=scale(cereals_data)

# There are 4 missing values in the entire dataframe
sum(is.na(normalized_cereals_data)) # 4
```

```
## [1] 4
```

```
# Remove all the cereals with missing values
normalized_cereals_data=as.data.frame(na.omit(normalized_cereals_data))

dim(normalized_cereals_data)
```

```
## [1] 74 12
```

#After removing missing values and normalizing the columns (scaling), there are 74 rows and 12 columns.

---

1. Using the Euclidean distance to the normalized measurements, apply hierarchical clustering to the data. To compare the clustering from single linkage, complete linkage, average linkage, and Ward, use Agnes. Choose the best method.

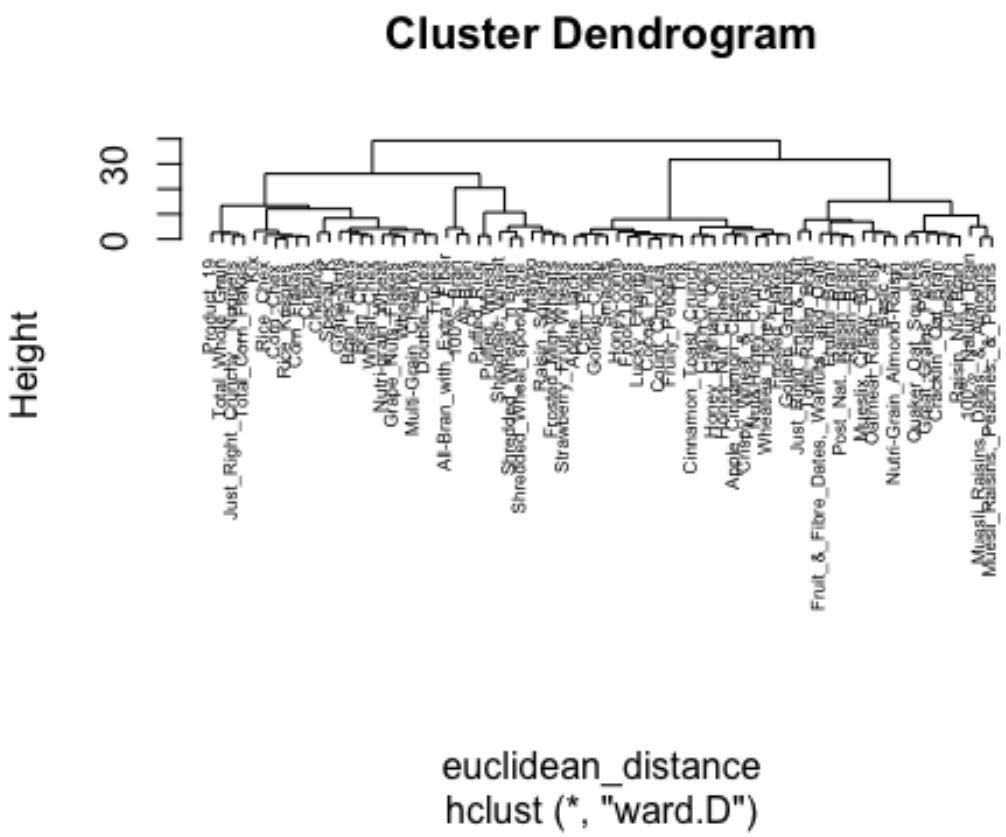
Solution:

#Compared to other links, Ward Linkage has the highest agglomerative coefficient (0.9088247), indicating that it is the best linkage method.

```
# Using AGNES to compare the clustering from the single linkage, complete lin
kage, average linkage, and Ward methodologies
linkages =c("average", "single", "complete", "ward")
names(linkages) <- c("average", "single", "complete", "ward")

# formula for computing the agglomerative coefficient
agglomerative_coef_calc <- function(linkage_method) {
  agnes(normalized_cereals_data, method = linkage_method)$ac
}

# calculating each linkage method's agglomerative coefficient
map_dbl(linkages, agglomerative_coef_calc)

##   average    single  complete      ward
## 0.7888569 0.6091225 0.8508357 0.9088247
```

#Hierarchical Clustering using ward linkage and Euclidean Distance:

```
# Dissimilarity matrix
euclidean_distance=dist(normalized_cereals_data,method="euclidean")

# Using ward linkage and Euclidean distance to apply hierarchical clustering
hierarchical_clustering_w_euclidean_ward=hclust(euclidean_distance,method="wa
rd.D")

# Ward linkage methodology: a visual representation of the Dendrogram
plot(hierarchical_clustering_w_euclidean_ward,cex=0.5,hang=0.1)
```



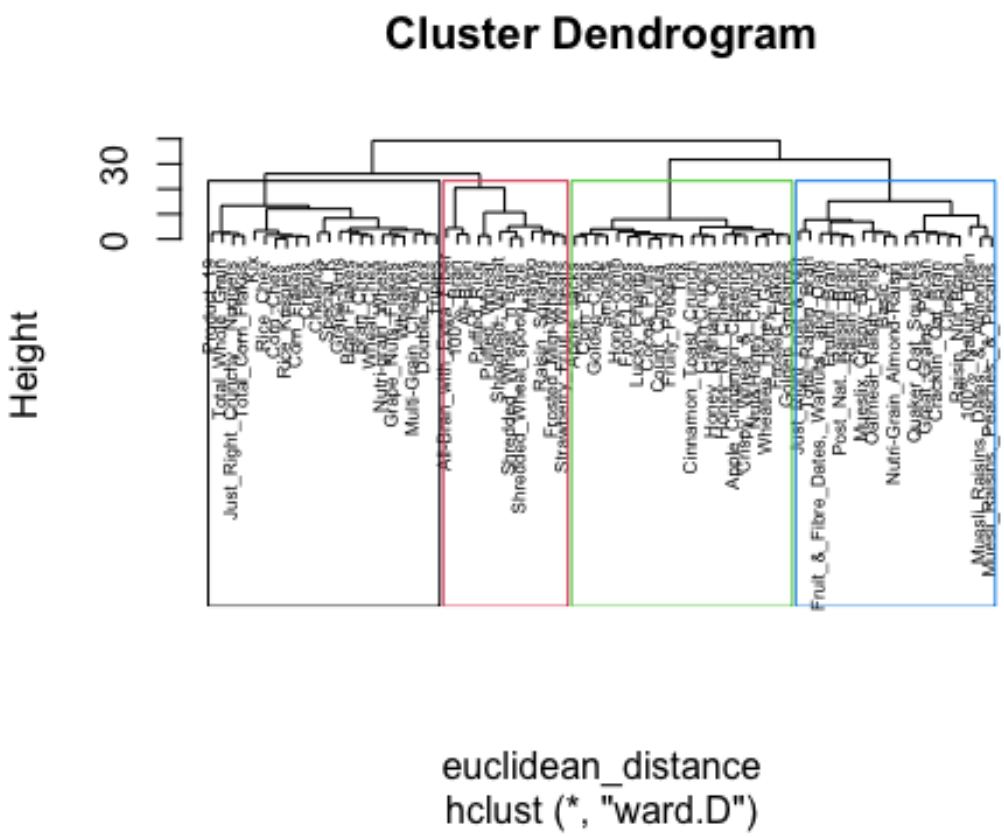
---

2. How many clusters would you choose?

Solution:

#The most apparent and meaningful division between groups seems to be achieved with a cutoff of 25. There are four clusters when the cutoff value is 25.

```
plot(hierarchical_clustering_w_euclidean_ward,cex=0.5,hang=0.1)
rect.hclust(hierarchical_clustering_w_euclidean_ward,k=4,border=1:4)
```



#Add the clusters assignment to the Cereals data

```
clusters4=cutree(hierarchical_clustering_w_euclidean_ward,k=4)
table(clusters4)

## clusters4
##  1  2  3  4
## 12 19 21 22

data_w_clusters_assignment=cbind.data.frame(normalized_cereals_data,clusters4
)
```

---

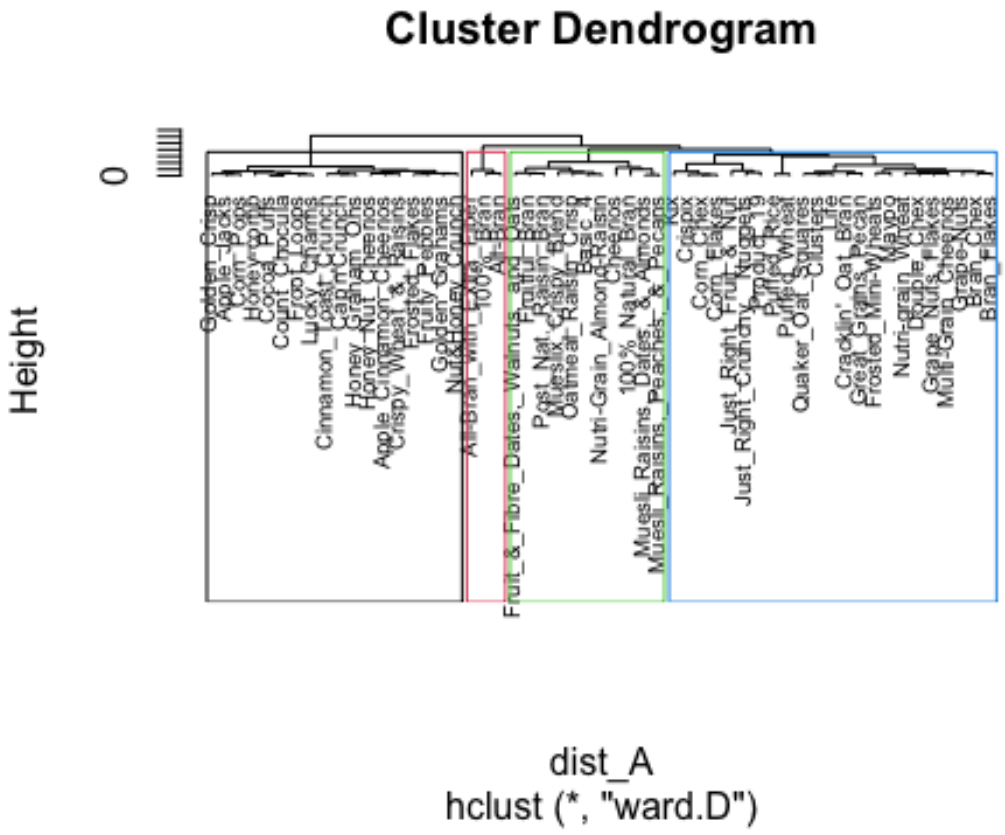
3. Comment on the structure of the clusters and on their stability. Hint: To check stability, partition the data and see how well clusters formed based on one part apply to the other.

#Cluster partition A

```
# Data clustering with 1 to 55 rows in A and the remaining rows (74%) and 26%
in B
cereals_A <-normalized_cereals_data[1:55,]
cereals_B <-normalized_cereals_data[56:74,]
```

#Assign each record in partition B to the cluster with the closest centroid using the cluster centroids from A.

```
dist_A=dist(cereals_A,method="euclidean")
hierarchical_ward_A=hclust(dist_A,method="ward.D")
plot(hierarchical_ward_A,cex=0.6,hang=-1)
rect.hclust(hierarchical_ward_A,k=4,border=1:4)
```




```
clusters4_A=cutree(hierarchical_ward_A,k=4)
table(clusters4_A)

## clusters4_A
##  1  2  3  4
##  3 11 18 23

data_w_clusters_assignment_A=cbind.data.frame(cereals_A,clusters4_A)
```

#Get the means of all columns in all 4 clusters

```
cluster1=colMeans(data_w_clusters_assignment_A[data_w_clusters_assignment_A$c
lusters4_A=="1",])
cluster2=colMeans(data_w_clusters_assignment_A[data_w_clusters_assignment_A$c
lusters4_A=="2",])
cluster3=colMeans(data_w_clusters_assignment_A[data_w_clusters_assignment_A$c
```

```
lusters4_A=="3",])
cluster4=colMeans(data_w_clusters_assignment_A[data_w_clusters_assignment_A$c
lusters4_A=="4",])
```

#Combining all the clusters into one dataframe

```
centroid_A=rbind(cluster1, cluster2, cluster3, cluster4)
```

#Calculate which cluster is closest to each data point in Partition B.

```
B_data_distance_from_clustersA=rowMins(distance(cereals_B,centroid_A[,-13]))
total_clusters_A_B=c(data_w_clusters_assignment_A$clusters4_A,B_data_distance
_from_clustersA)
data_w_clusters_assignment =cbind(data_w_clusters_assignment,total_clusters_A
_B)
```

Assess how consistent the cluster assignments are compared to the assignments based on all the data.

Solution:

#The consistency of cluster assignments based on the B partition is 68.4%, while the consistency of cluster assignments based on all data is 77.03%.

```
table(data_w_clusters_assignment$clusters4==data_w_clusters_assignment$total_
clusters_A_B)

##
## FALSE  TRUE
##    17    57

table(data_w_clusters_assignment$clusters4[56:74]==data_w_clusters_assignment
$total_clusters_A_B[56:74])

##
## FALSE  TRUE
##     6    13
```

I also tried using six clusters, however the partition data did not match as well as it did with four clusters, therefore the stability was pretty low.

---

4. 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."

Solution:

#Cluster 1 is the "Healthy Cereals" cluster, as seen in the table below. Cluster 1 has the highest fiber contest, rating, and protein and potassium levels among the other clusters. It

also has the lowest amounts of calories, fat, sodium, and sugar. Thus, cluster 1 represents the healthiest choice.

```
# Finding the four clusters' centroids and trying to find out which is health
y
cluster1_centroid=colMeans(data_w_clusters_assignment[data_w_clusters_assignm
ent$clusters4 == "1",])
cluster2_centroid=colMeans(data_w_clusters_assignment[data_w_clusters_assignm
ent$clusters4 == "2",])
cluster3_centroid=colMeans(data_w_clusters_assignment[data_w_clusters_assignm
ent$clusters4 == "3",])
cluster4_centroid=colMeans(data_w_clusters_assignment[data_w_clusters_assignm
ent$clusters4 == "4",])

# Combining all the cluster centroids
all_cluster_centroids_of_all_columns=rbind(cluster1_centroid, cluster2_centro
id, cluster3_centroid, cluster4_centroid)
all_cluster_centroids_of_all_columns
```

```
##                     calories    protein         fat      sodium      fiber
## cluster1_centroid -1.5080547  0.2629535 -0.75808029 -1.36294322  0.9152548
## cluster2_centroid  0.9433345  0.6074881  0.98066557 -0.04635267  0.4220701
## cluster3_centroid  0.2088503 -0.9331883 -0.01290349  0.10611786 -0.6631465
## cluster4_centroid -0.1666359  0.1245569 -0.46452580  0.79007459 -0.1972548
##                        carbo     sugars     potass   vitamins     weight
## cluster1_centroid -0.4186917 -0.9956591  0.6639622 -0.6115433 -0.8447175
## cluster2_centroid -0.1784772  0.5228713  0.7739062  0.1491414  1.0099135
## cluster3_centroid -0.6075921  1.0162649 -0.7283802 -0.1453172 -0.1967771
## cluster4_centroid  0.8997331 -0.8157229 -0.3425791  0.4650151 -0.1967771
##                         cups     rating clusters4 total_clusters_A_B
## cluster1_centroid -0.3768781  1.6415416         1           3.250000
## cluster2_centroid -0.5653466 -0.3057055         2           2.736842
## cluster3_centroid  0.2328980 -0.9969602         3           3.000000
## cluster4_centroid  0.4799337  0.2498970         4           3.909091
```

#Cluster 1 has 12 cereals shown below and can be used in the daily cafeterias

```
data_w_clusters_assignment[data_w_clusters_assignment$clusters4 == '1',]
```

```
##                              calories    protein         fat     sodium
## 100%_Bran                 -1.8929836  1.3286071 -0.01290349 -0.3539844
## All-Bran                  -1.8929836  1.3286071 -0.01290349  1.1967306
## All-Bran_with_Extra_Fiber -2.9194605  1.3286071 -1.00647256 -0.2346986
## Frosted_Mini-Wheats       -0.3532681  0.4151897 -1.00647256 -1.9046994
## Maypo                     -0.3532681  1.3286071 -0.01290349 -1.9046994
## Puffed_Rice               -2.9194605 -1.4116451 -1.00647256 -1.9046994
## Puffed_Wheat              -2.9194605 -0.4982277 -1.00647256 -1.9046994
## Raisin_Squares            -0.8665066 -0.4982277 -1.00647256 -1.9046994
## Shredded_Wheat            -1.3797451 -0.4982277 -1.00647256 -1.9046994
## Shredded_Wheat_'n'Bran    -0.8665066  0.4151897 -1.00647256 -1.9046994
## Shredded_Wheat_spoon_size -0.8665066  0.4151897 -1.00647256 -1.9046994
```

```
## Strawberry_Fruit_Wheats   -0.8665066 -0.4982277 -1.00647256 -1.7257708
##                                 fiber       carbo       sugars      potass
## 100%_Bran                    3.29284661 -2.50878291 -0.234390576  2.57536849
## All-Bran                     2.87327158 -1.99692385 -0.462771138  3.14346448
## All-Bran_with_Extra_Fiber    4.97114672 -1.74099432 -1.604673946  3.28548848
## Frosted_Mini-Wheats          0.35582142 -0.20541712 -0.006010015  0.01893653
## Maypo                       -0.90290366  0.30644194 -0.919532261 -0.05207547
## Puffed_Rice                 -0.90290366 -0.46134666 -1.604673946 -1.18826745
## Puffed_Wheat                -0.48332864 -1.22913525 -1.604673946 -0.69118346
## Raisin_Squares              -0.06375361  0.05051241 -0.234390576  0.16096053
## Shredded_Wheat               0.35582142  0.30644194 -1.604673946 -0.05207547
## Shredded_Wheat_'n'Bran       0.77539645  1.07423054 -1.604673946  0.58703252
## Shredded_Wheat_spoon_size    0.35582142  1.33016007 -1.604673946  0.30298453
## Strawberry_Fruit_Wheats      0.35582142  0.05051241 -0.462771138 -0.12308746
##                               vitamins     weight        cups    rating clus
ters4
## 100%_Bran                   -0.1453172 -0.1967771 -2.11003399 1.8321876
1
## All-Bran                    -0.1453172 -0.1967771 -2.11003399 1.1930986
1
## All-Bran_with_Extra_Fiber   -0.1453172 -0.1967771 -1.37953029 3.6333849
1
## Frosted_Mini-Wheats         -0.1453172 -0.1967771 -0.09040611 1.1161895
1
## Maypo                       -0.1453172 -0.1967771  0.76901001 0.8674423
1
## Puffed_Rice                 -1.2642598 -3.5195485  0.76901001 1.2878220
1
## Puffed_Wheat                -1.2642598 -3.5195485  0.76901001 1.4479620
1
## Raisin_Squares              -0.1453172 -0.1967771 -1.37953029 0.9017710
1
## Shredded_Wheat              -1.2642598 -1.3265194  0.76901001 1.8202929
1
## Shredded_Wheat_'n'Bran      -1.2642598 -0.1967771 -0.64902659 2.2642977
1
## Shredded_Wheat_spoon_size   -1.2642598 -0.1967771 -0.64902659 2.1453309
1
## Strawberry_Fruit_Wheats     -0.1453172 -0.1967771  0.76901001 1.1887196
1
##                             total_clusters_A_B
## 100%_Bran                                    1
## All-Bran                                     1
## All-Bran_with_Extra_Fiber                    1
## Frosted_Mini-Wheats                          4
## Maypo                                        4
## Puffed_Rice                                  4
## Puffed_Wheat                                 4
## Raisin_Squares                               4
## Shredded_Wheat                               4
```

```
## Shredded_Wheat_'n'Bran                    4
## Shredded_Wheat_spoon_size                 4
## Strawberry_Fruit_Wheats                   4
```

Should the data be normalized? If not, how should they be used in the cluster analysis?

To make sure that all the variables are on the same scale, normalizing the data is usually a good idea before doing cluster analysis. By doing this, it is possible to prevent any variable from overpowering the analysis due to its magnitude alone, as opposed to its actual significance. For instance, when analyzing the grams of sugar, fat, and fiber in cereals, it's crucial to scale each variable suitably to guarantee that each has an equivalent influence on the clustering.
