

FML Assignment-5

Rajeswari Theegala

2023-11-29

Load Data Set and Libraries #First, we will load all of the packages needed for this task. “ISLR”, “caret”, “dplyr”, “tidyverse”, “factoextra”, “ggplot2”, “proxy”, “NbClust”, “ppclust”, “dendextend”, “cluster”, and “tinytex” will be loaded for this problem.

```
library(cluster)
library(ISLR)
library(caret)
```

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

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

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

```
library(dplyr)
```

```
##
```

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

```
## The following objects are masked from 'package:stats':
```

```
##
```

```
##   filter, lag
```

```
## The following objects are masked from 'package:base':
```

```
##
```

```
##   intersect, setdiff, setequal, union
```

```
library(tidyverse)
```

```
## Warning: package 'lubridate' was built under R version 4.3.2
```

```
## -- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
```

```
## v forcats 1.0.0      v stringr 1.5.0
```

```
## v lubridate 1.9.3    v tibble 3.2.1
```

```
## v purrr 1.0.2       v tidyr 1.3.0
```

```
## v readr 2.1.4
```

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

```
library(factoextra)
```

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

```
library(ggplot2)
library(proxy)
```

```
##
## Attaching package: 'proxy'
##
## The following objects are masked from 'package:stats':
##
##   as.dist, dist
##
## The following object is masked from 'package:base':
##
##   as.matrix
```

```
library(NbClust)
library(ppclust)
```

```
## Warning: package 'ppclust' was built under R version 4.3.2
```

```
library(dendextend)
```

```
##
## -----
## Welcome to dendextend version 1.17.1
## 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
```

```
library(tinytex)
```

```
## Warning: package 'tinytex' was built under R version 4.3.2
```

```
#The "cereal" data set will then be imported into the RStudio environment.
```

```
# Import data set from BlackBoard into the RStudio environment
```

```
cereal <- read.csv("C:/Users/dell/Downloads/Cereals.csv")
```

Review Data Structure #A summary of the data set will be displayed for inspection.Examine the first few rows of the data set

```
head(cereal)
```

```
##           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
```

```
# Look into the data set's structure.
```

```
str(cereal)
```

```
## '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 ...
```

```
# Investigate the summary of the data set
summary(cereal)
```

```
##      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
##
```

```
#Data Preprocessing
#The data will be scaled prior to removing the NA(Null) values from the data set.
# Create duplicate of data set for preprocessing
```

```
cereal_scaled <- cereal
# Scale the data set prior to placing it into a clustering algorithm
cereal_scaled[ , c(4:16)] <- scale(cereal[ , c(4:16)])
# Remove NA values from data set
cereal_preprocessed <- na.omit(cereal_scaled)

# Review the scaled data set with NA's removed
head(cereal_preprocessed)
```

```
##      name mfr type  calories  protein      fat
## 1      100%_Bran  N   C -1.8929836  1.3286071 -0.01290349
## 2    100%_Natural_Bran  Q   C  0.6732089  0.4151897  3.96137277
## 3      All-Bran  K   C -1.8929836  1.3286071 -0.01290349
```

```
## 4 All-Bran_with_Extra_Fiber K C -2.9194605 1.3286071 -1.00647256
## 6 Apple_Cinnamon_Cheerios G C 0.1599704 -0.4982277 0.98066557
## 7 Apple_Jacks K C 0.1599704 -0.4982277 -1.00647256
## sodium fiber carbo sugars potass vitamins shelf
## 1 -0.3539844 3.29284661 -2.5087829 -0.2343906 2.5753685 -0.1453172 0.9515734
## 2 -1.7257708 -0.06375361 -1.7409943 0.2223705 0.5160205 -1.2642598 0.9515734
## 3 1.1967306 2.87327158 -1.9969238 -0.4627711 3.1434645 -0.1453172 0.9515734
## 4 -0.2346986 4.97114672 -1.7409943 -1.6046739 3.2854885 -0.1453172 0.9515734
## 6 0.2424445 -0.27354112 -1.1011705 0.6791317 -0.4071355 -0.1453172 -1.4507595
## 7 -0.4136273 -0.48332864 -0.9732057 1.5926539 -0.9752315 -0.1453172 -0.2495930
## weight cups rating
## 1 -0.1967771 -2.1100340 1.8321876
## 2 -0.1967771 0.7690100 -0.6180571
## 3 -0.1967771 -2.1100340 1.1930986
## 4 -0.1967771 -1.3795303 3.6333849
## 6 -0.1967771 -0.3052601 -0.9365625
## 7 -0.1967771 0.7690100 -0.6756899
```

#After pre-processing and scaling the data, the total number of observations went from 77 to 74. Therefore, there were only 3 records with an “NA” value

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.”

#Single Linkage:

Create the dissimilarity matrix for the numeric values in the data set via Euclidean distance measurements

```
cereal_d_euclidean <- dist(cereal_preprocessed[ , c(4:16)], method = "euclidean")
```

```
# Perform hierarchical clustering via the single linkage method
```

```
ag_hc_single <-agnes(cereal_d_euclidean, method = "single")
```

```
# Plot the results of the different methods
```

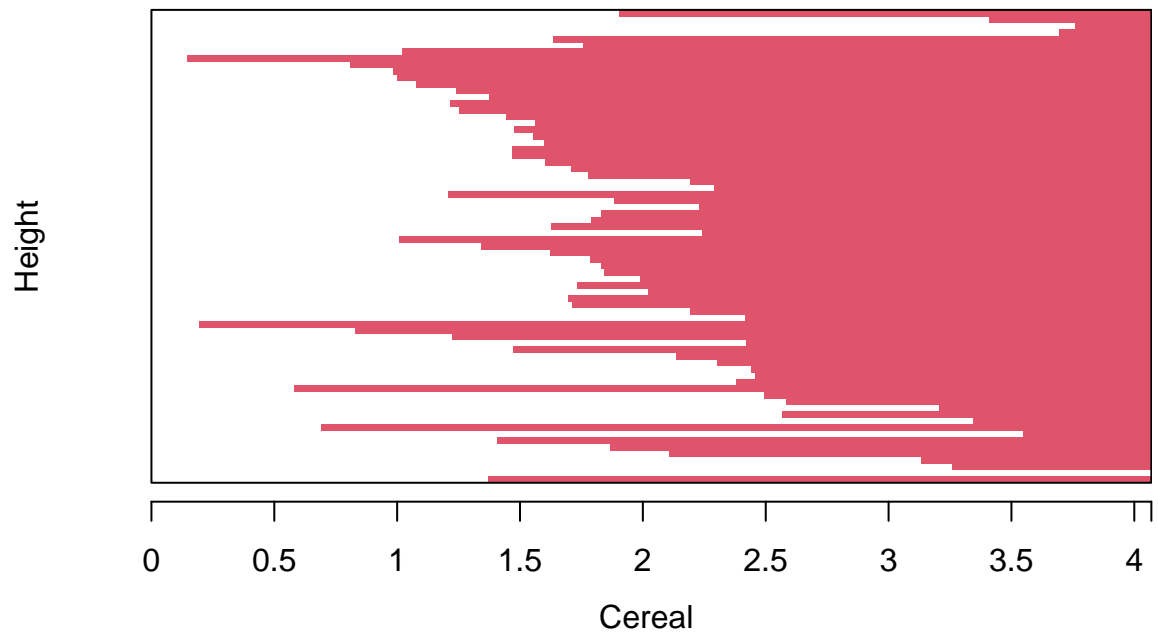
```
plot(ag_hc_single,
     main = "Customer Cereal Ratings - AGNES - Single Linkage Method",
     xlab = "Cereal",
     ylab = "Height",
     cex.axis = 1,
     cex = 0.55,
     hang = -1)
```

```
## Warning in plot.window(xlim, ylim, log = log, ...): "hang" is not a graphical
## parameter
```

```
## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...): "hang"
## is not a graphical parameter
```

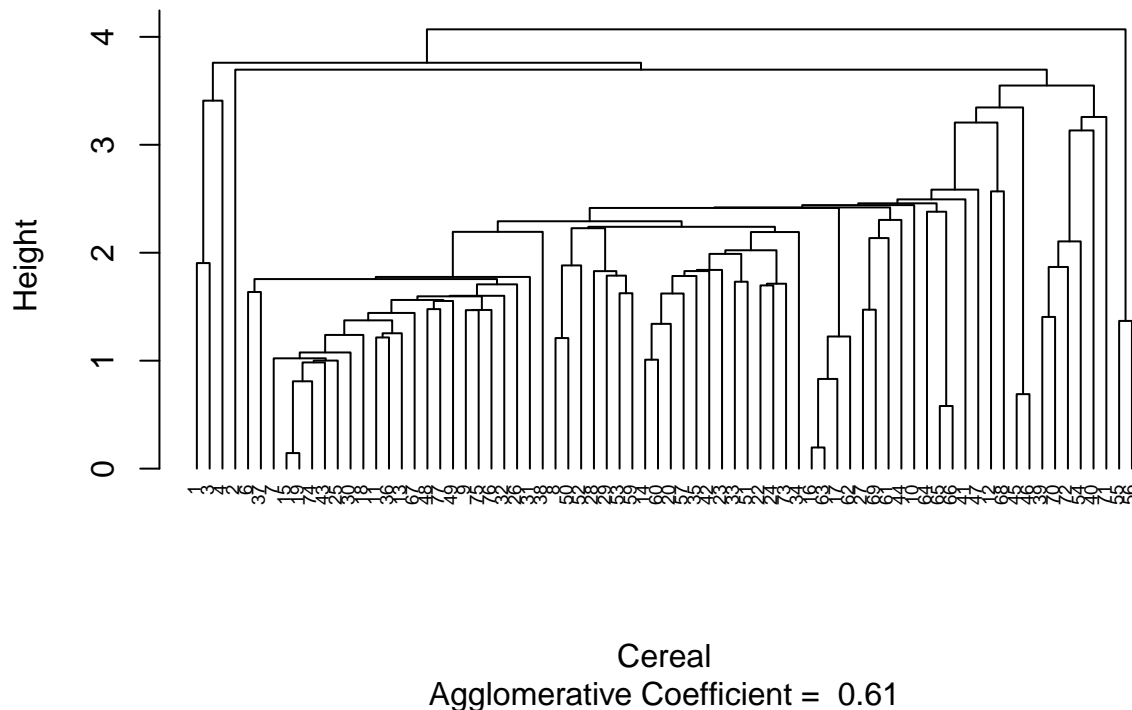
```
## Warning in axis(1, at = at.vals, labels = lab.vals, ...): "hang" is not a
## graphical parameter
```

Customer Cereal Ratings – AGNES – Single Linkage Method



Agglomerative Coefficient = 0.61

Customer Cereal Ratings – AGNES – Single Linkage Method



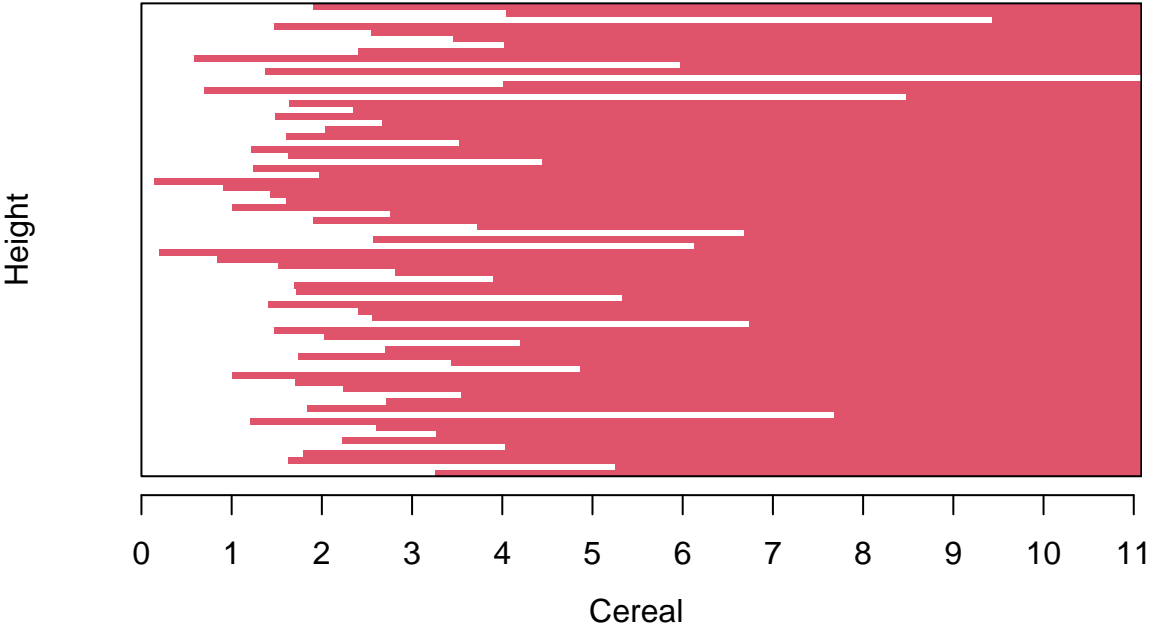
```
#Complete Linkage:
# Perform hierarchical clustering via the complete linkage method
ag_hc_complete <- agnes(cereal_d_euclidean, method = "complete")
# Plot the results of the different methods
plot(ag_hc_complete,
  main = "Customer Cereal Ratings - AGNES - Complete Linkage Method",
  xlab = "Cereal",
  ylab = "Height",
  cex.axis = 1,
  cex = 0.55,
  hang = -1)
```

```
## Warning in plot.window(xlim, ylim, log = log, ...): "hang" is not a graphical
## parameter
```

```
## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...): "hang"
## is not a graphical parameter
```

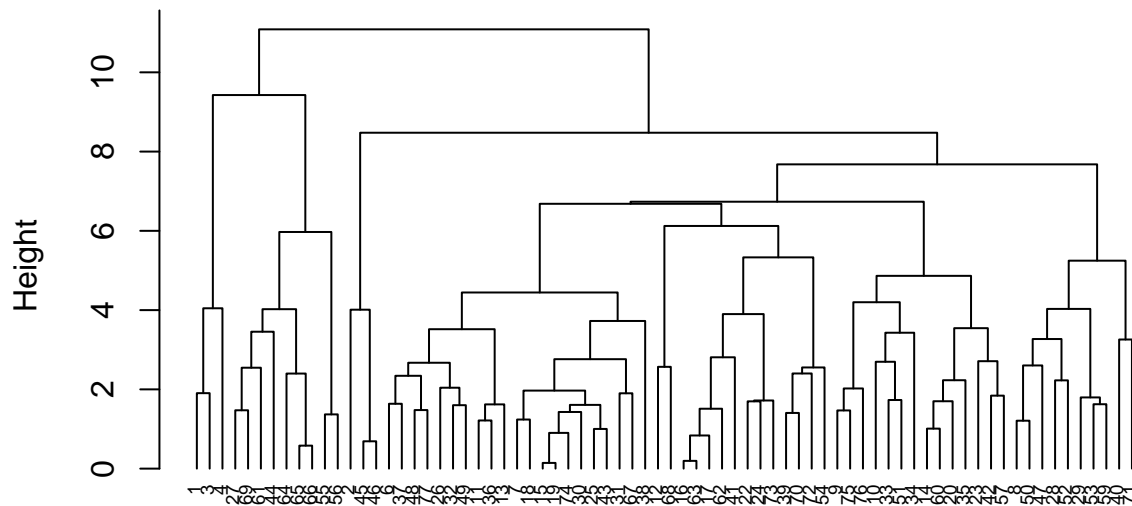
```
## Warning in axis(1, at = at.vals, labels = lab.vals, ...): "hang" is not a
## graphical parameter
```

Customer Cereal Ratings – AGNES – Complete Linkage Methc



Agglomerative Coefficient = 0.84

Customer Cereal Ratings – AGNES – Complete Linkage Method



Cereal
Agglomerative Coefficient = 0.84

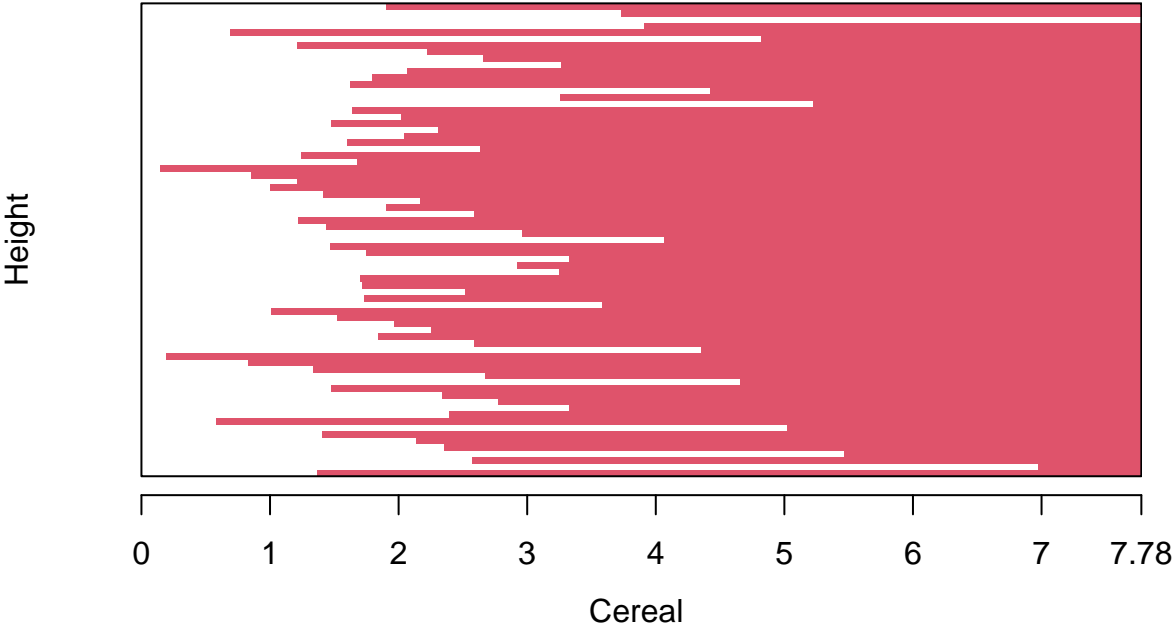
```
#Average Linkage:
# Perform hierarchical clustering via the average linkage method
ag_hc_average <- agnes(cereal_d_euclidean, method = "average")
# Plot the results of the different methods
plot(ag_hc_average,
     main = "Customer Cereal Ratings - AGNES - Average Linkage Method",
     xlab = "Cereal",
     ylab = "Height",
     cex.axis = 1,
     cex = 0.55,
     hang = -1)
```

```
## Warning in plot.window(xlim, ylim, log = log, ...): "hang" is not a graphical
## parameter
```

```
## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...): "hang"
## is not a graphical parameter
```

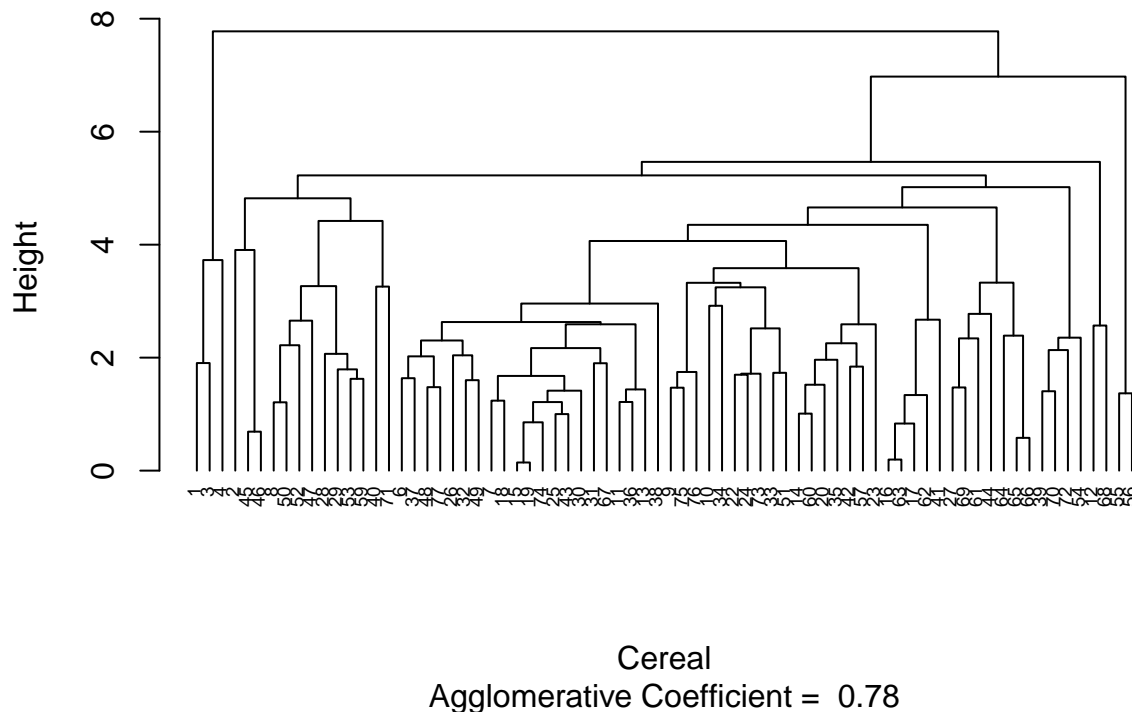
```
## Warning in axis(1, at = at.vals, labels = lab.vals, ...): "hang" is not a
## graphical parameter
```

Customer Cereal Ratings – AGNES – Average Linkage Method



Agglomerative Coefficient = 0.78

Customer Cereal Ratings – AGNES – Average Linkage Method



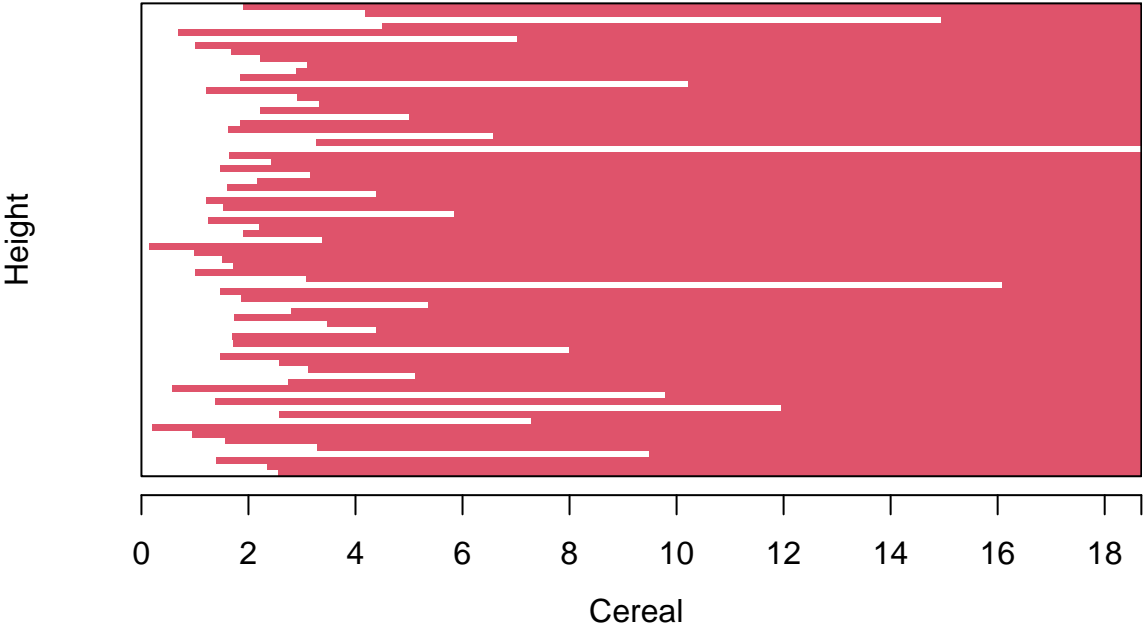
```
#Ward Method:
# Perform hierarchical clustering via the ward linkage method
ag_hc_ward <- agnes(cereal_d_euclidean, method = "ward")
# Plot the results of the different methods
plot(ag_hc_ward,
  main = "Customer Cereal Ratings - AGNES - Ward Linkage Method",
  xlab = "Cereal",
  ylab = "Height",
  cex.axis = 1,
  cex = 0.55,
  hang = -1)
```

```
## Warning in plot.window(xlim, ylim, log = log, ...): "hang" is not a graphical
## parameter
```

```
## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...): "hang"
## is not a graphical parameter
```

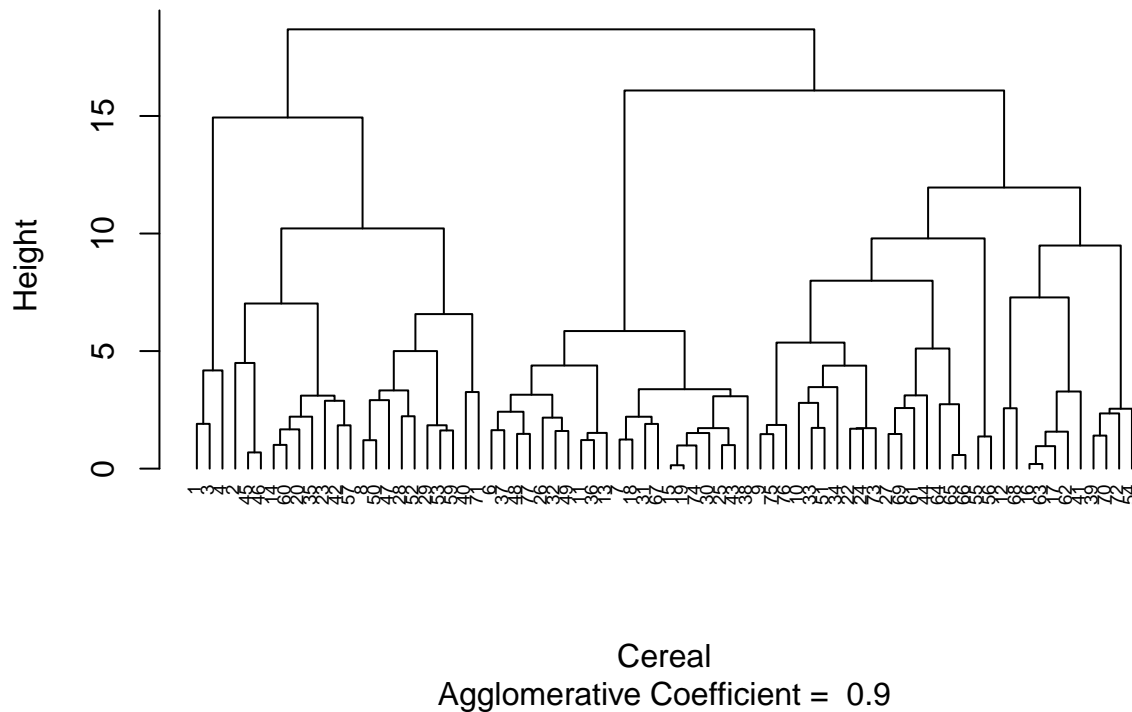
```
## Warning in axis(1, at = at.vals, labels = lab.vals, ...): "hang" is not a
## graphical parameter
```

Customer Cereal Ratings – AGNES – Ward Linkage Method



Agglomerative Coefficient = 0.9

Customer Cereal Ratings – AGNES – Ward Linkage Method



#The best clustering method would be based on the agglomerative coefficient that is returned from each

*#Single Linkage: 0.61 #Complete Linkage: 0.84 #Average Linkage: 0.78 #Ward Method: 0.90
#As a result, the Ward method will be chosen as the best clustering model in this problem.*

#Assignment Task B

#"How many clusters would you choose?"

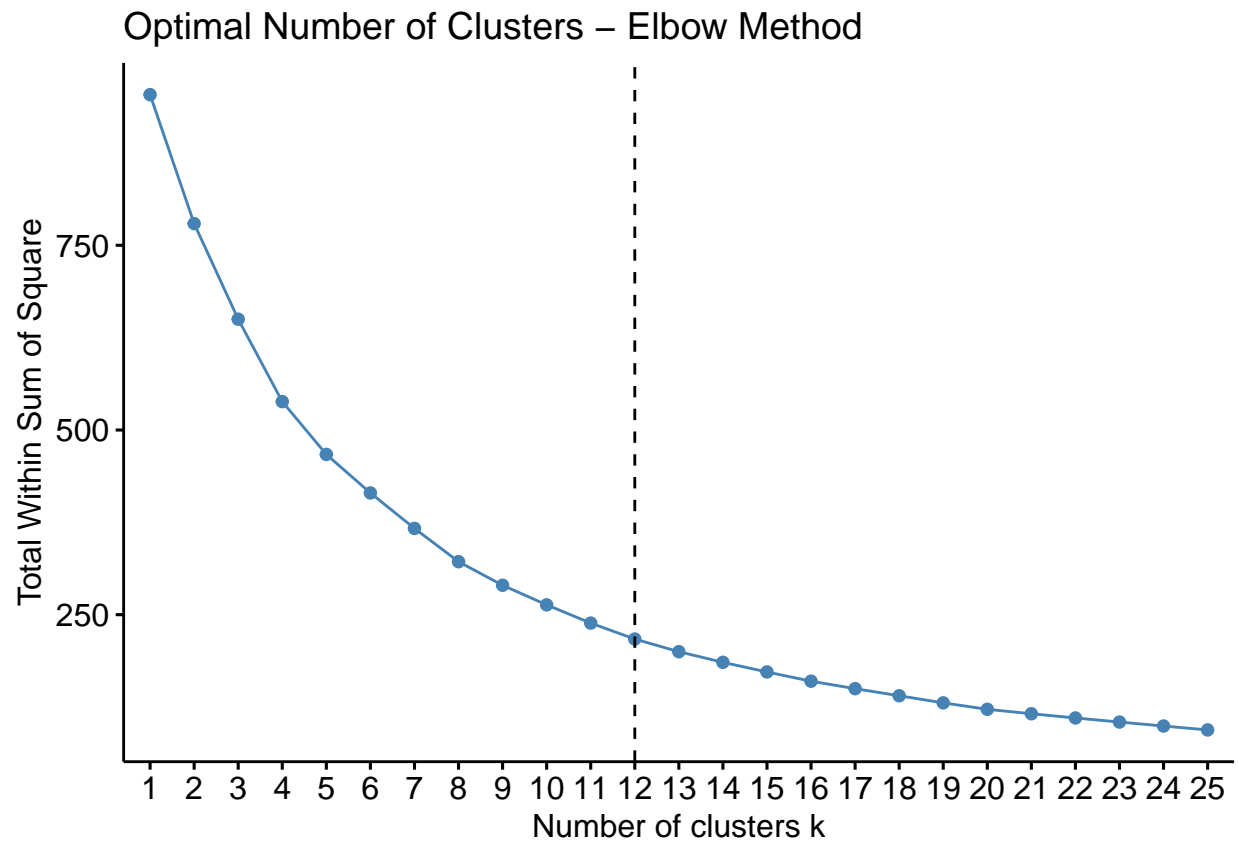
#To determine the appropriate number of clusters, we will use the elbow and silhouette methods.

#Elbow Method:

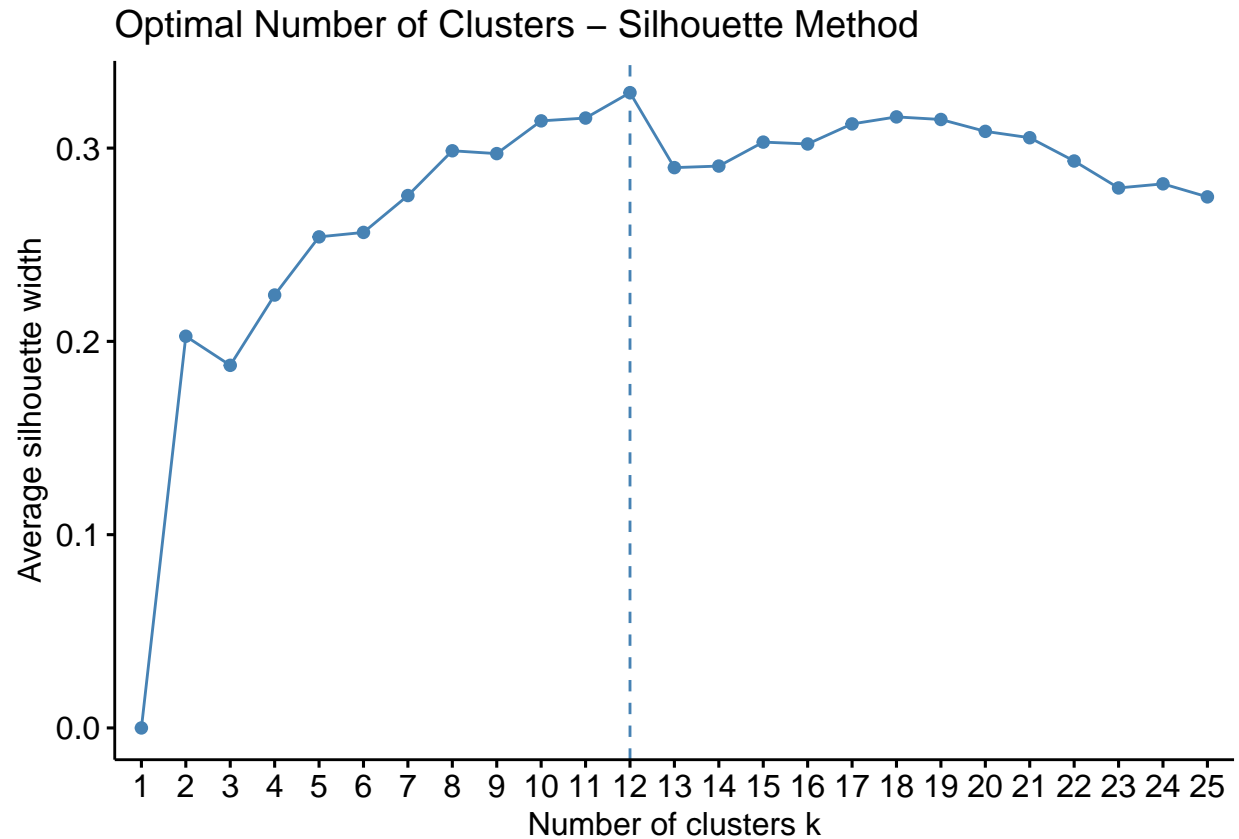
Determine the optimal number of clusters for the dataset via the Elbow method

```
fviz_nbclust(cereal_preprocessed[, c(4:16)], hcut, method = "wss", k.max = 25) +
```

```
labs(title = "Optimal Number of Clusters - Elbow Method") +  
geom_vline(xintercept = 12, linetype = 2)
```



```
#Silhouette Method:  
# Determine the optimal number of clusters for the dataset via the silhouette method  
fviz_nbclust(cereal_preprocessed[, c(4:16)],  
             hcut,  
             method = "silhouette",  
             k.max = 25) +  
labs(title = "Optimal Number of Clusters - Silhouette Method")
```



#Based on the agreement of the silhouette and elbow method, the appropriate number of clusters would be 12
#Below we will outline the 12 clusters on the hierarchical tree
Plot of the Ward hierarchical tree with the 12 clusters outlined for reference

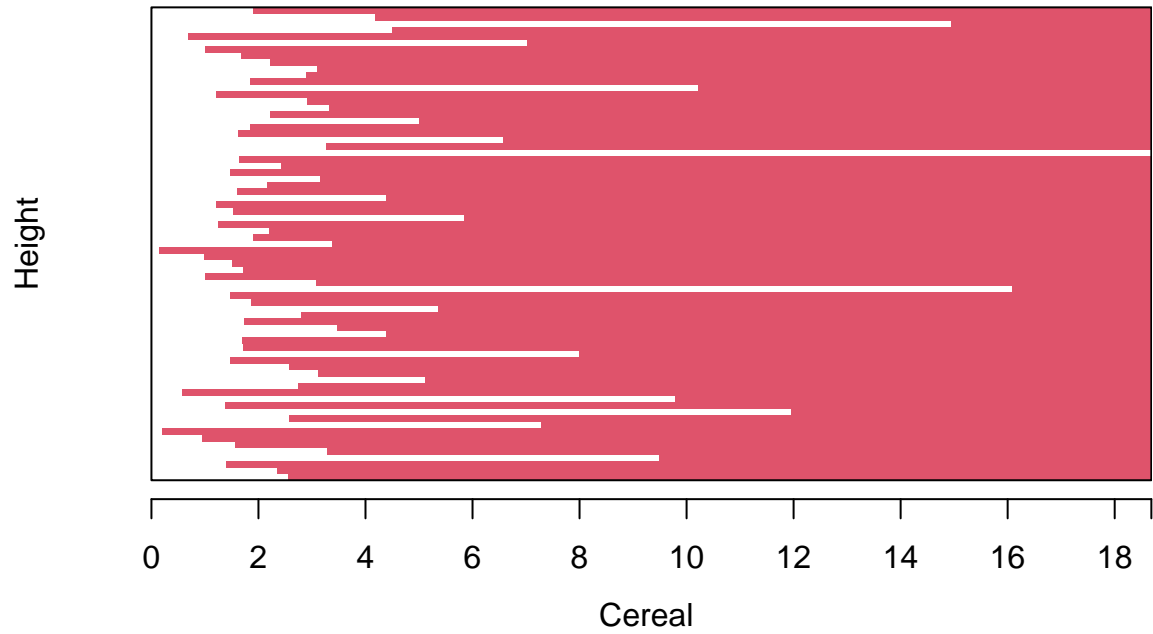
```
plot(ag_hc_ward,
     main = "AGNES - Ward Linkage Method - 12 Clusters Outlined",
     xlab = "Cereal",
     ylab = "Height",
     cex.axis = 1,
     cex = 0.55,
     hang = -1)
```

```
## Warning in plot.window(xlim, ylim, log = log, ...): "hang" is not a graphical
## parameter
```

```
## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...): "hang"
## is not a graphical parameter
```

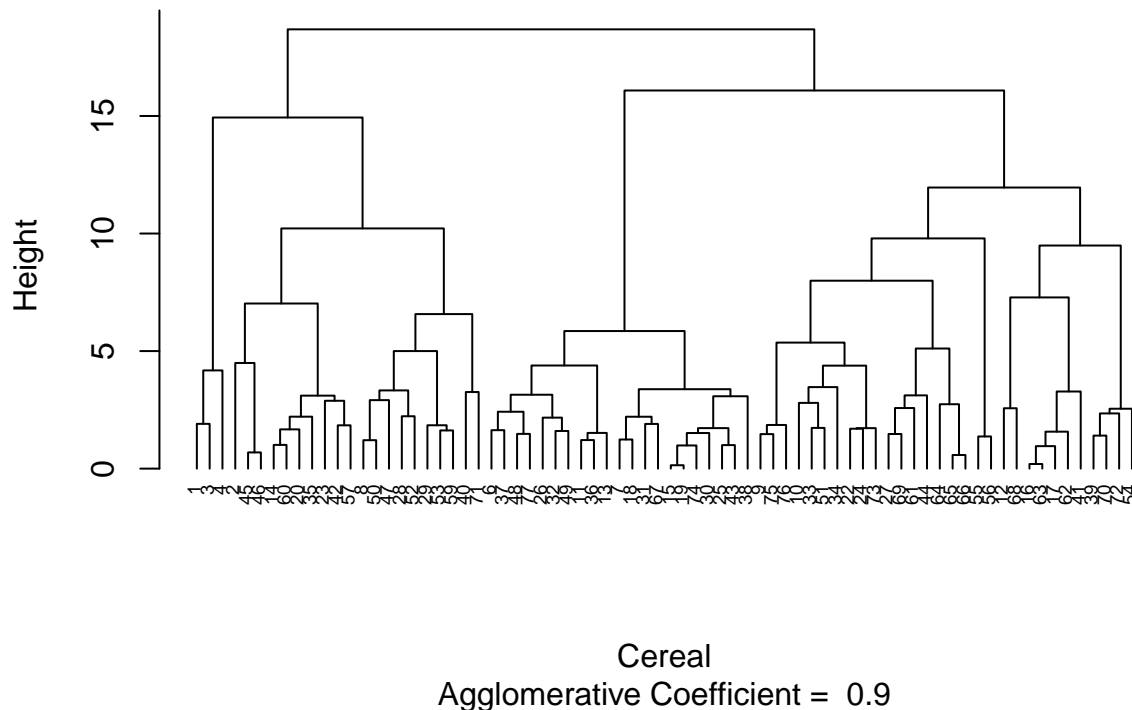
```
## Warning in axis(1, at = at.vals, labels = lab.vals, ...): "hang" is not a
## graphical parameter
```

AGNES – Ward Linkage Method – 12 Clusters Outlined



Agglomerative Coefficient = 0.9

AGNES – Ward Linkage Method – 12 Clusters Outlined



#Assignment Task C

#"Comment on the structure of the clusters and on their stability. Hint: To check stability, partition #1. Cluster partition A #2. Use the cluster centroids from A to assign each record in partition B (each #3. Assess how consistent the cluster assignments are compared to the assignments based on all the data"

#All Data Assigned Clusters:

#The assigned clusters for all data sets will be in "cereal_preprocessed_1":

Cut the tree into 12 clusters for analysis
`ward_clusters_12 <- cutree(ag_hc_ward, k = 12)`

Add the assigned cluster to the preprocessed data set
`cereal_preprocessed_1 <- cbind(cluster = ward_clusters_12,
cereal_preprocessed)`

#Partition Data:

#To check stability of clusters, the data set will be split into a 70/30 partition. The 70% will be use

Set the seed for randomized functions
`set.seed(982579)`

```

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

#Re-Run Clustering with Partitioned Data:

#For the purposes of this task, we will assume the same K value (12) and ward clustering method to determine the number of clusters

# Create the dissimilarity matrix for the numeric values in the partitioned data set via Euclidean distance
cereal_d_euclidean_A <- dist(cereal_preprocessed_PartitionA[ , c(4:16)],
method = "euclidean")

# Perform hierarchical clustering via the ward linkage method on partitioned data
ag_hc_ward_A <- agnes(cereal_d_euclidean_A, method = "ward")

# Plot the results of the different methods
plot(ag_hc_ward_A,
      main = "Customer Cereal Ratings - Ward Linkage Method - Partition A",
      xlab = "Cereal",
      ylab = "Height",
      cex.axis = 1,
      cex = 0.55,
      hang = -1)

```

```

## Warning in plot.window(xlim, ylim, log = log, ...): "hang" is not a graphical
## parameter

```

```

## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...): "hang"
## is not a graphical parameter

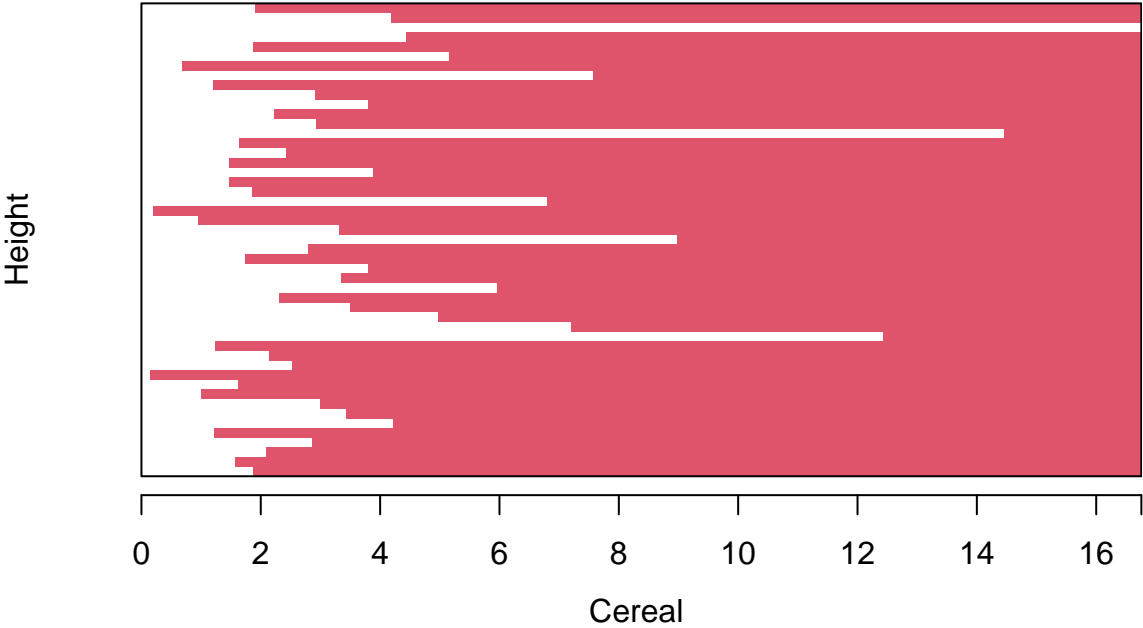
```

```

## Warning in axis(1, at = at.vals, labels = lab.vals, ...): "hang" is not a
## graphical parameter

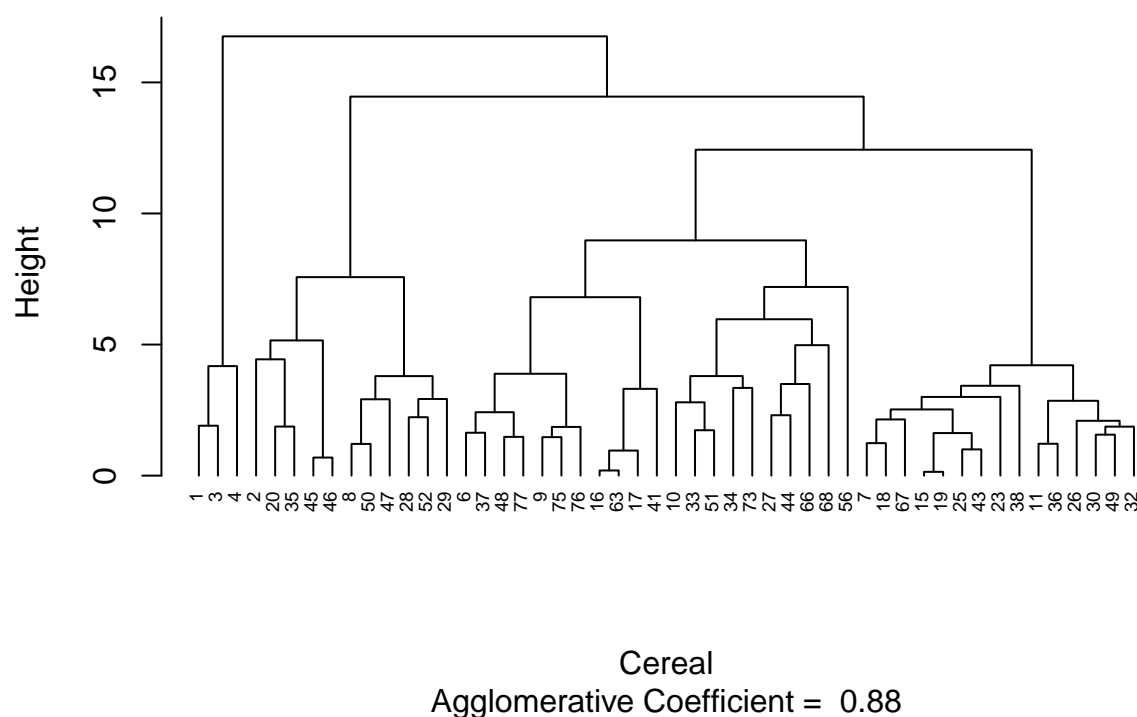
```

Customer Cereal Ratings – Ward Linkage Method – Partition /



Agglomerative Coefficient = 0.88

Customer Cereal Ratings – Ward Linkage Method – Partition A



```
# Cut the tree into 12 clusters for analysis
ward_clusters_12_A <- cutree(ag_hc_ward_A, k = 12)

# Add the assigned cluster to the preprocessed data set
cereal_preprocessed_A <- cbind(cluster = ward_clusters_12_A,
cereal_preprocessed_PartitionA)

#The centroids for each of the clusters will need to be calculated, so we can find the closest centroid

# Find the centroids for the re-ran Ward hierarchical clustering
ward_Centroids_A <- aggregate(cereal_preprocessed_A[ , 5:17],
list(cereal_preprocessed_A$cluster), mean)
ward_Centroids_A <- data.frame(Cluster = ward_Centroids_A[ , 1], Centroid =
rowMeans(ward_Centroids_A[ , -c(1:4)]))
ward_Centroids_A <- ward_Centroids_A$Centroid

# Calculate Centers of Partition B data set
cereal_preprocessed_PartitionB_centers <-
data.frame(cereal_preprocessed_PartitionB[, 1:3], Center =
rowMeans(cereal_preprocessed_PartitionB[ , 4:16]))

# Calculate the distance between the centers of partition A and the values of partition B
B_to_A_centers <- dist(ward_Centroids_A,
cereal_preprocessed_PartitionB_centers$Center, method = "euclidean")

# Assign the clusters based on the minimum distance to cluster centers
```

```

cereal_preprocessed_B <- cbind(cluster =
c(4,8,7,3,5,6,7,11,11,10,8,5,10,1,10,1,4,12,12,7,7,1,4,9),
cereal_preprocessed_PartitionB)

# Combine partitions A and B for comparison to original clusters
cereal_preprocessed_2 <- rbind(cereal_preprocessed_A, cereal_preprocessed_B)
cereal_preprocessed_1 <-
cereal_preprocessed_1[order(cereal_preprocessed_1$name), ]
cereal_preprocessed_2 <-
cereal_preprocessed_2[order(cereal_preprocessed_2$name), ]

#Now that the data has been assigned by both methods (full data and partitioned data), we can compare the
sum(cereal_preprocessed_1$cluster == cereal_preprocessed_2$cluster)

```

```
## [1] 14
```

```

#From this result, it can be stated that the clusters are not very stable. With 70% of the data available
# Visualize the cluster assignments to see any difference between the two
# Plot of original hierarchical clustering algorithm

ggplot(data = cereal_preprocessed_1, aes(cereal_preprocessed_1$cluster)) +
  geom_bar(fill = "yellow3") +
  labs(title="Count of Cluster Assignments - All Original Data") +
  labs(x="Cluster Assignment", y="Count") +
  guides(fill=FALSE) +
  scale_x_continuous(breaks=c(1:12)) +
  scale_y_continuous(breaks=c(5,10,15,20), limits = c(0,25))

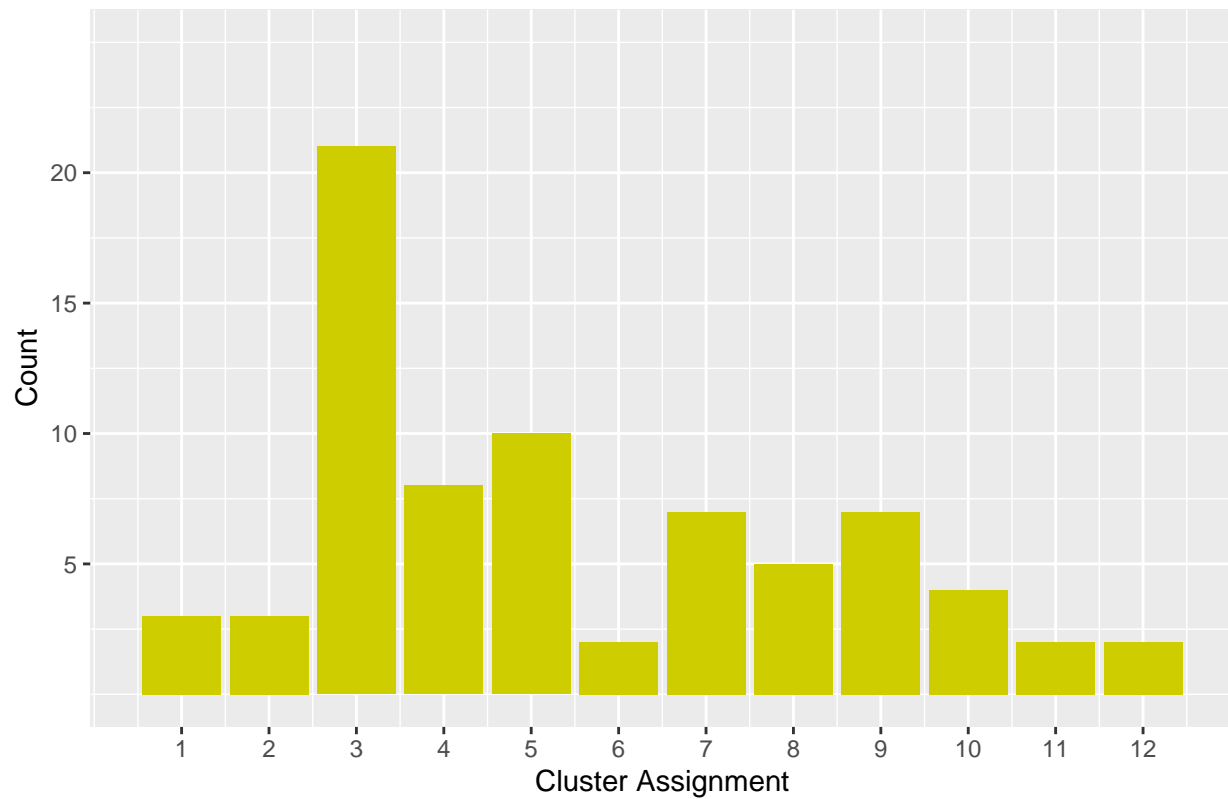
```

```

## Warning: The '<scale>' argument of 'guides()' cannot be 'FALSE'. Use "none" instead as
## of ggplot2 3.3.4.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.

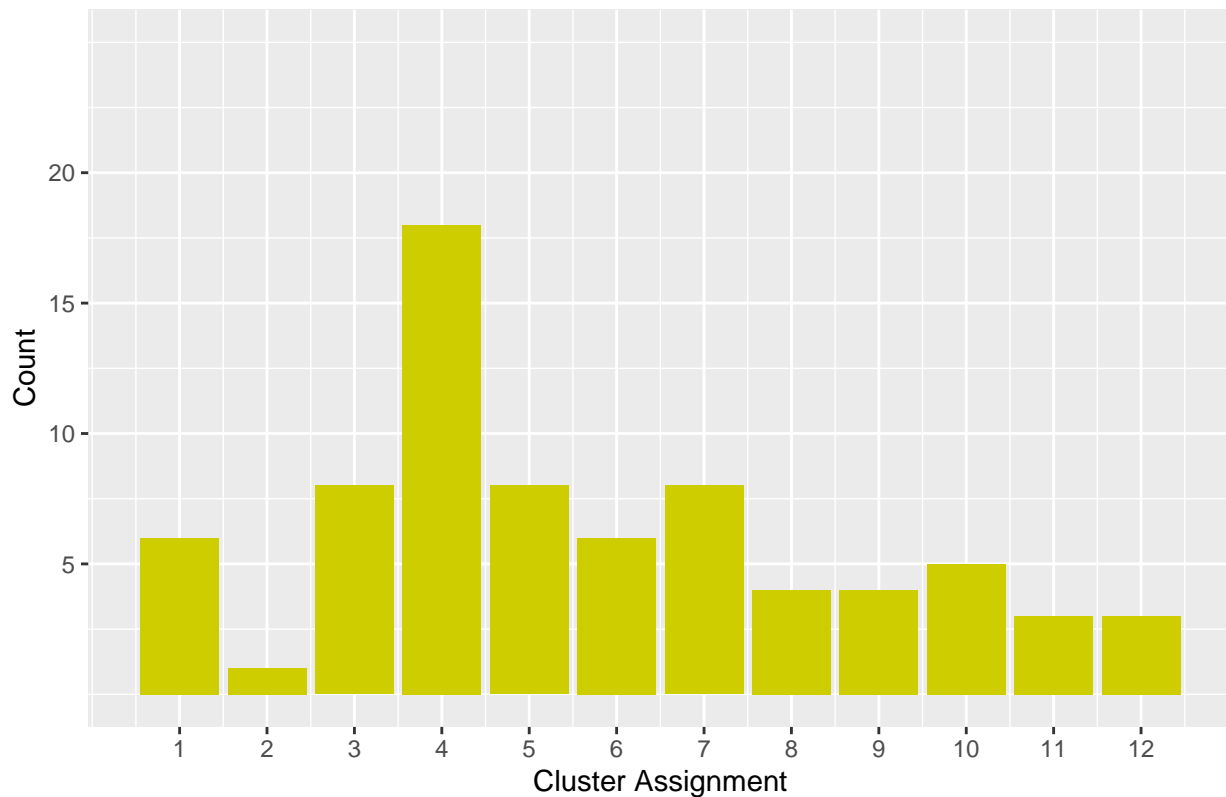
```

Count of Cluster Assignments – All Original Data



```
# Plot of algorithm that was partitioned prior to assigning the remaining data
ggplot(data = cereal_preprocessed_2, aes(cereal_preprocessed_2$cluster)) +
  geom_bar(fill = "yellow3") +
  labs(title="Count of Cluster Assignments - Partitioned Data") +
  labs(x="Cluster Assignment", y="Count") +
  guides(fill=FALSE) +
  scale_x_continuous(breaks=c(1:12)) +
  scale_y_continuous(breaks=c(5,10,15,20), limits = c(0,25))
```

Count of Cluster Assignments – Partitioned Data



#When we use partitioned data, we can see that Cluster 3 shrinks dramatically. As a result, several of the other clusters grew in size. When the data is partitioned, the clusters appear to be more uniformly spread over the 12 clusters, according to the graphic.

Assignment Task D

#“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?”

#In this case, normalizing the data would be improper. This is because the specific cereal sample under investigation determines how nutrition data from cereals should be scaled or normalized. Because of this, the data set that was collected might only contain cereals that are very high in sugar but low in iron, fiber, and other nutrients. It is impossible to estimate the nutritional value of cereal for a child after the data inside the sample set is scaled or normalized. An insensitive observer may conclude that a cereal with an iron score of 0.999 gives a child almost all of the iron they require, but it might actually be the best of the worst in the sample set, offering very little to no iron.

#Therefore, converting the data into a ratio to a child’s daily recommended intake of calories, fiber, carbs, and other nutrients would be a preferable way to preprocess the data. By doing this, analysts would be able to evaluate clusters more intelligently and prevent a small number of significant variables from overriding distance computations. When analyzing the clusters, an analyst may use the cluster averages to determine how much of a student’s daily nutritional needs would come from XX cereal. This would allow the employees to choose “healthy” cereal clusters with greater knowledge.