Assignment-5

vivek rao kathheragandla

2024-04-01

Importing required libraries

```
library("cluster")
```

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

```
library("caret")
```

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

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

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

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

```
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
```

#Import data set from local drive

```
CEREALS<-read.csv("C:\\Users\\KATHHERAGANDLA VIVEK\\OneDrive\\Documents\\FML\\\\Assignment_5
\\Cereals.csv")
```

#printing some records form the dataset

```
head(CEREALS)
```

```
##                         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
```

#printing the summary of the dataset

```
summary(CEREALS)
```

```
##      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
##
```

TASK-0 PRE-PROCESSING OF THE DATA

```
#initializing a dataframe to store the pre-processsed data
cereal_scaled_data <- CEREALS
#scaling the cereals data and preparing for the clustering algorithm
cereal_scaled_data[ , c(4:16)] <- scale(CEREALS[ , c(4:16)])
#removing unwanted null values
cleansed_data <- na.omit(cereal_scaled_data)
#printing the pre-processed data
head(cleansed_data)
```

```
##                        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
```

Following pre-processing, the data is sorted in an alphabetical order and, once the null values have been eliminated, has 74 records.

TASK 1 -

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.

```
#calculating the euclidean distance
Euclidean_dist<- dist(cleansed_data[ , c(4:16)], method = "euclidean")
#Euclidea_dist
```

APPLYING SINGLE LINKAGE

```
#using AGNES to compare and single linkage method for clustering
clustering_H_single <- agnes(Euclidean_dist, method = "single")
```

Plotting the hierarchical clustering using single linkage method in a graph with Height on X-axis and cereal on Y-axis.

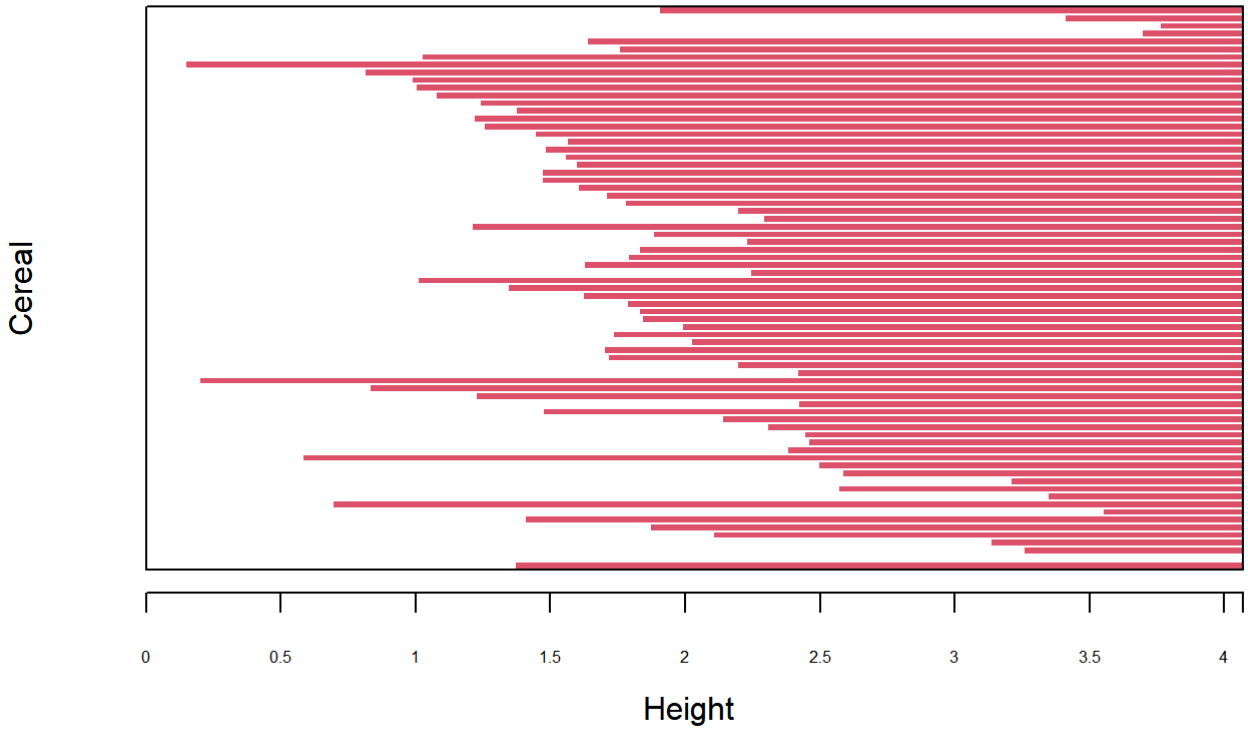
```
plot(clustering_H_single,
main = "Ratings, AGNES comparision (Single Linkage)",
 xlab = "Height",
 ylab = "Cereal",
 cex.axis = 0.5,
 cex = 0.5,
 hang = -1)
```

```
## 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
```

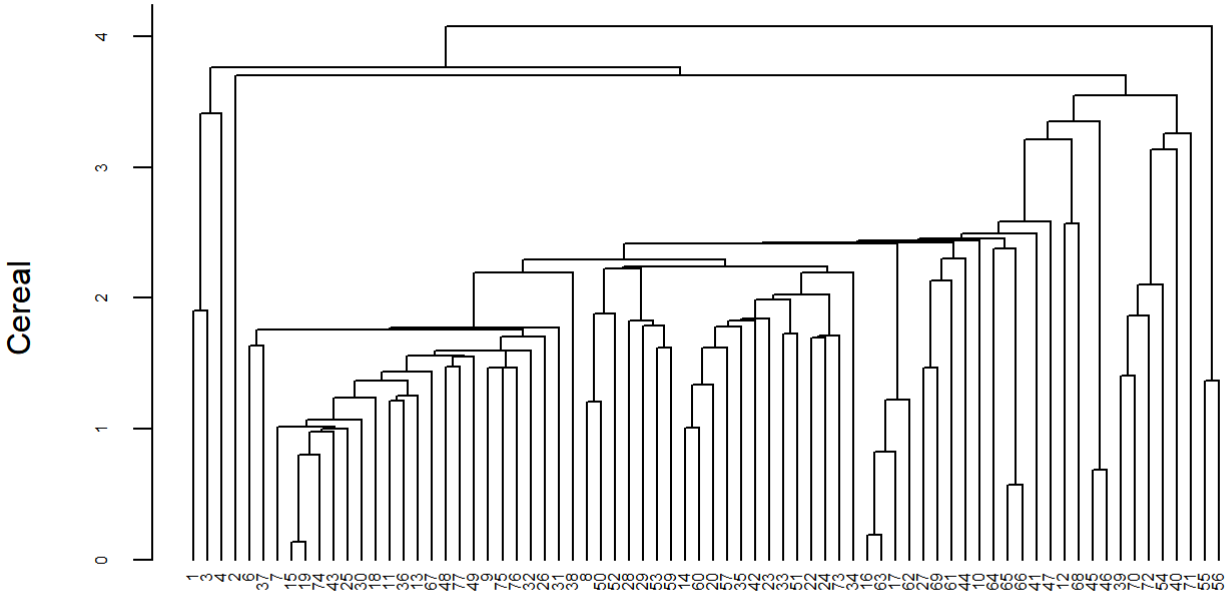
```
## Warning in axis(1, at = at.vals, labels = lab.vals, ...): "hang" is not a
## graphical parameter
```

Ratings, AGNES comparision (Single Linkage)



Agglomerative Coefficient = 0.61

Ratings, AGNES comparision (Single Linkage)



Agglomerative Coefficient = 0.61

```
## 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
```

APPLYING COMPLETE LINKAGE

```
#applying AGNES with complete linkage for clustering
clustering_H_complete <- agnes(Euclidean_dist, method = "complete")
```

plotting the clustering with complete linkage method

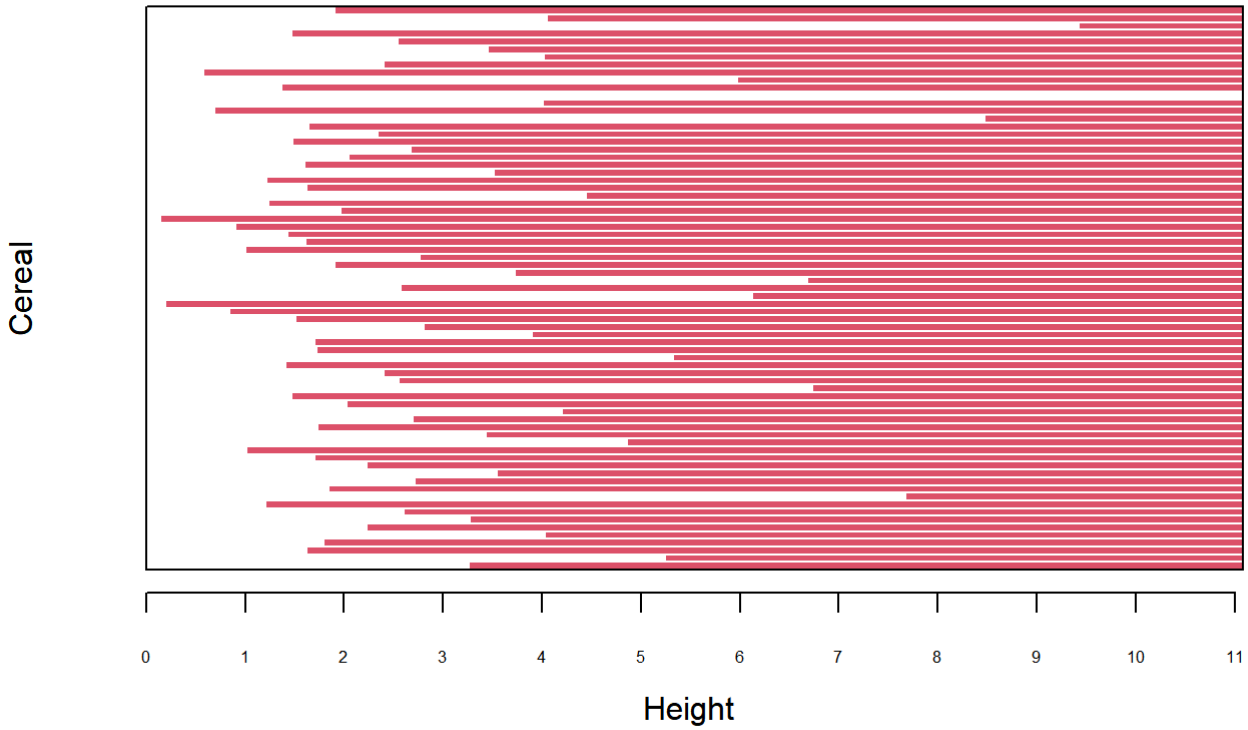
```
plot(clustering_H_complete,
 main = "Ratings AGNES comparision (Complete Linkage)",
 xlab = "Height",
 ylab = "Cereal",
 cex.axis = 0.5,
 cex = 0.50,
 hang = -1)
```

```
## 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
```

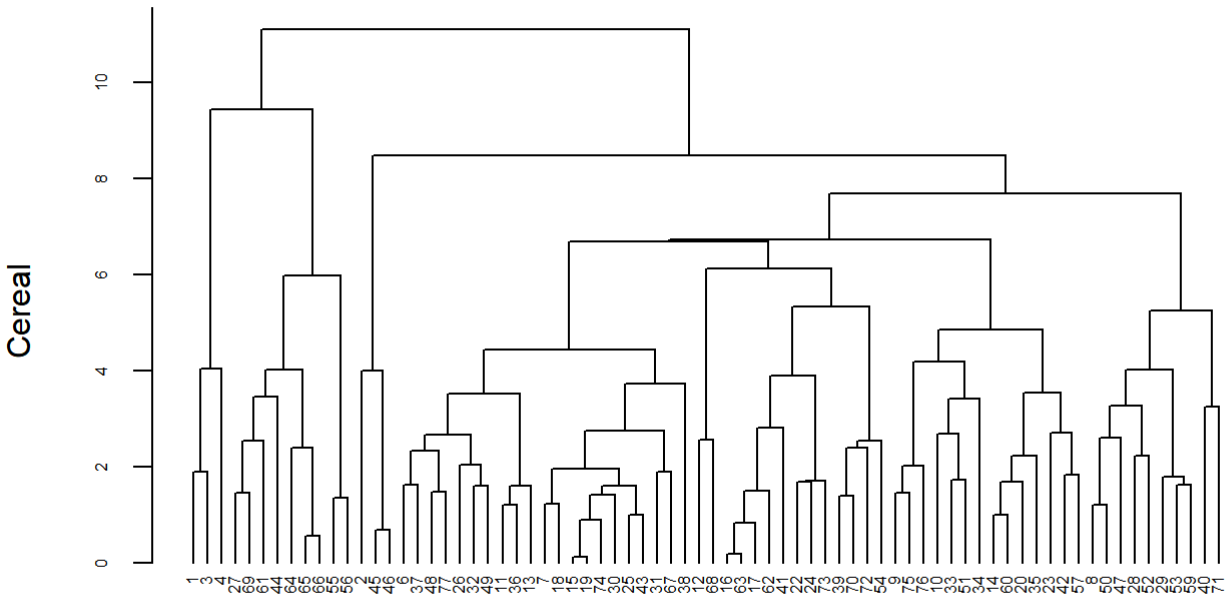
```
## Warning in axis(1, at = at.vals, labels = lab.vals, ...): "hang" is not a
## graphical parameter
```

Ratings AGNES comparision (Complete Linkage)



Agglomerative Coefficient = 0.84

Ratings AGNES comparision (Complete Linkage)



Height

Agglomerative Coefficient = 0.84

```
## 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
```

AVERAGE LINKAGE METHOD

```
#applying AGNES with Average linkage method
clustering_H_average <- agnes(Euclidean_dist, method = "average")
```

PLOTTING THE AGNES comparison with Average linkage method

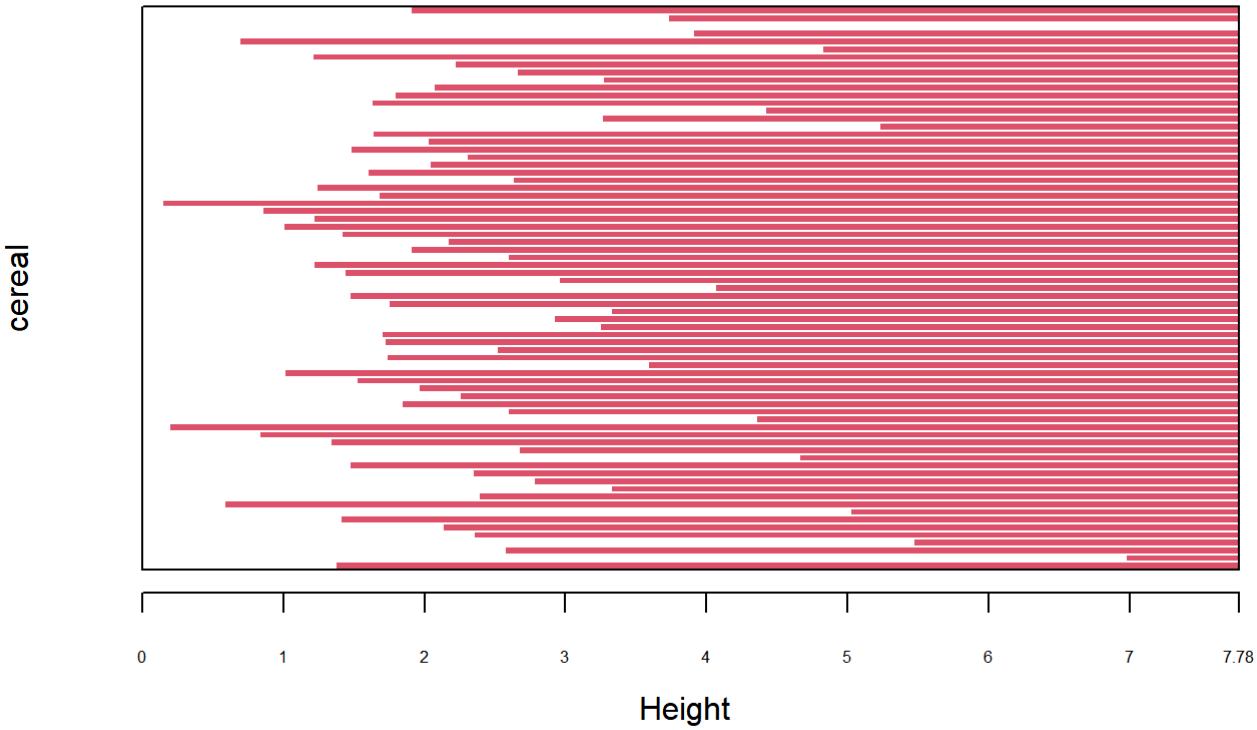
```
plot(clustering_H_average,
 main = "Ratings, AGNES comparision (Average Linkage)",
 xlab = "Height",
 ylab = "cereal",
 cex.axis = 0.5,
 cex = 0.50,
 hang = -1)
```

```
## 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
```

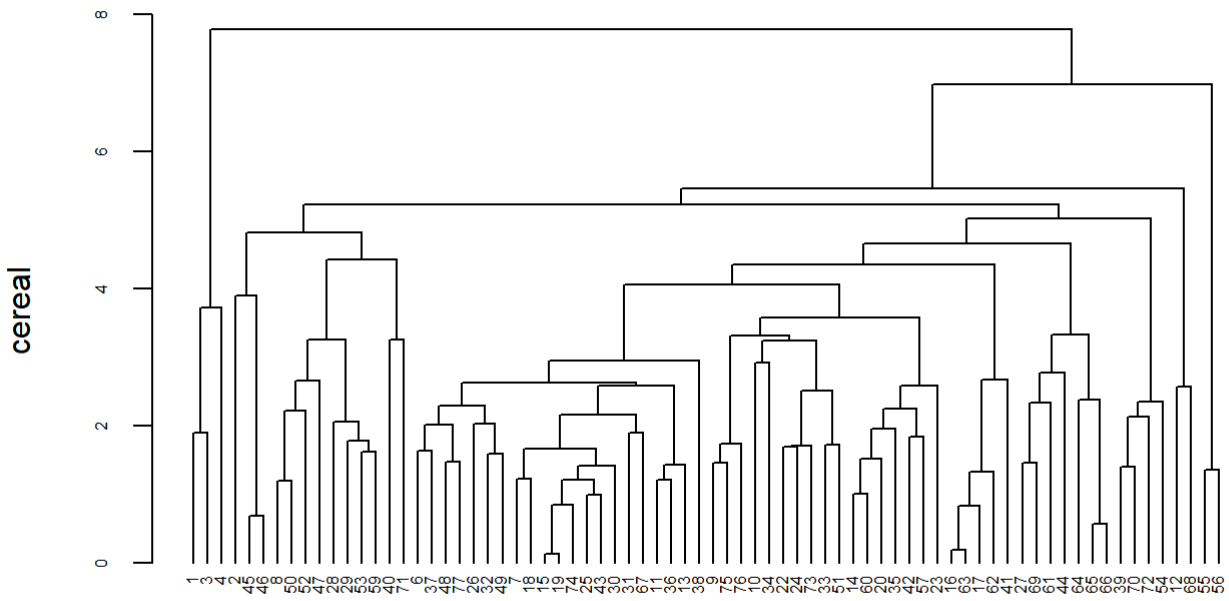
```
## Warning in axis(1, at = at.vals, labels = lab.vals, ...): "hang" is not a
## graphical parameter
```

Ratings, AGNES comparision (Average Linkage)



Agglomerative Coefficient = 0.78

Ratings, AGNES comparision (Average Linkage)



Agglomerative Coefficient = 0.78

```
## 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
```

PERFORMING WARD LINKAGE METHOD

```
clustering_H_WARD <- agnes(Euclidean_dist, method = "ward")
```

PLOTTING THE RESULTS AFTER USING WARD LINKAGE METHOD

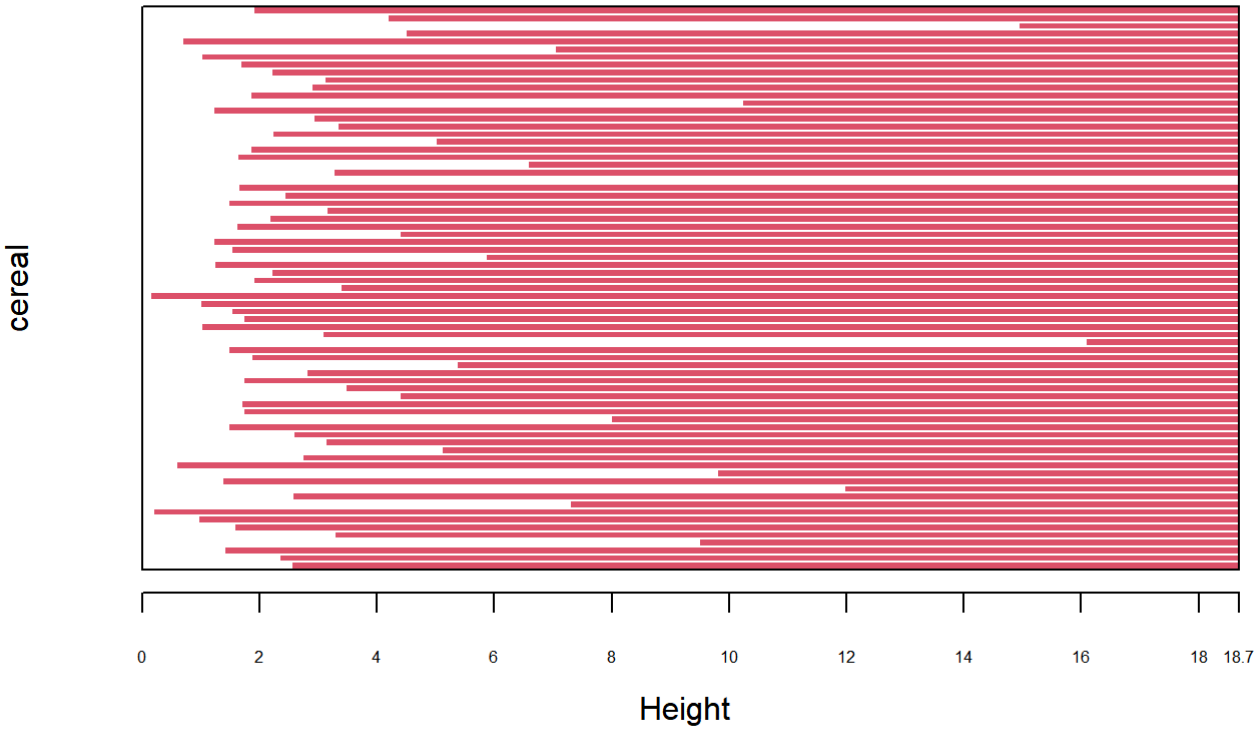
```
plot(clustering_H_WARD,
 main = "Ratings, AGNES comparision (Ward Linkage)",
 xlab = "Height",
 ylab = "cereal",
 cex.axis = 0.5,
 cex = 0.50,
 hang = -1)
```

```
## 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
```

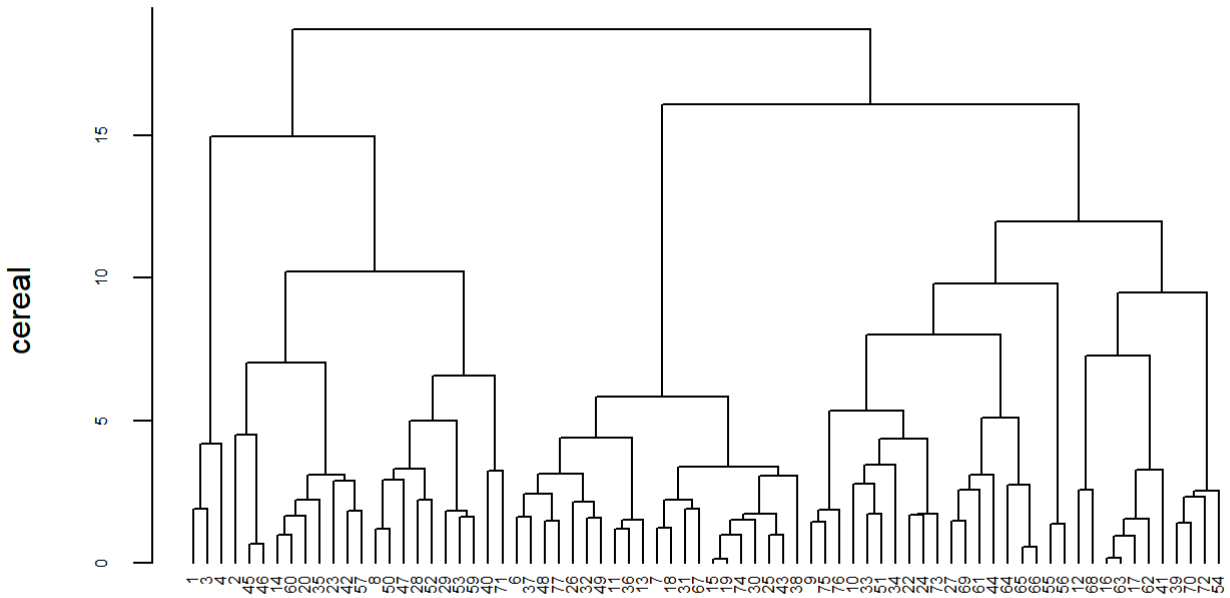
```
## Warning in axis(1, at = at.vals, labels = lab.vals, ...): "hang" is not a
## graphical parameter
```

Ratings, AGNES comparision (Ward Linkage)



Agglomerative Coefficient = 0.9

Ratings, AGNES comparision (Ward Linkage)



Agglomerative Coefficient = 0.9

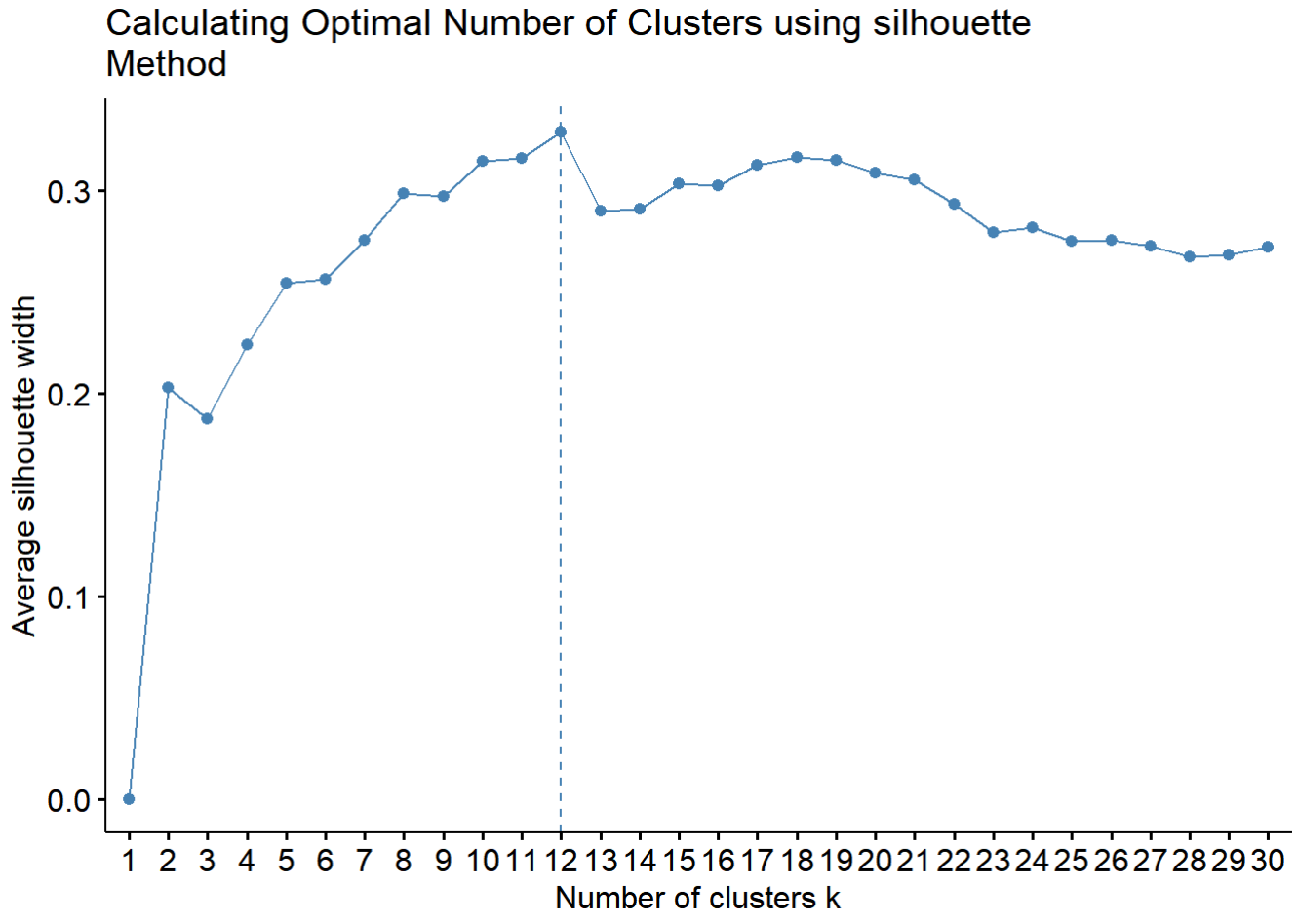
```
## Warning in plot.window(xlim, ylim, log = log, ...): "hang" is not a graphical
## parameter

## 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
```

After applying all the linkage methods with AGNES we found the observations as Single Linkage method: 0.61 Complete Linkage method: 0.84 Average Linkage method: 0.78 Ward Method method: 0.90 Here we can conclude that, the ward method with highest Agglomerative coefficient will be better suited for this clustering.

TASK-2

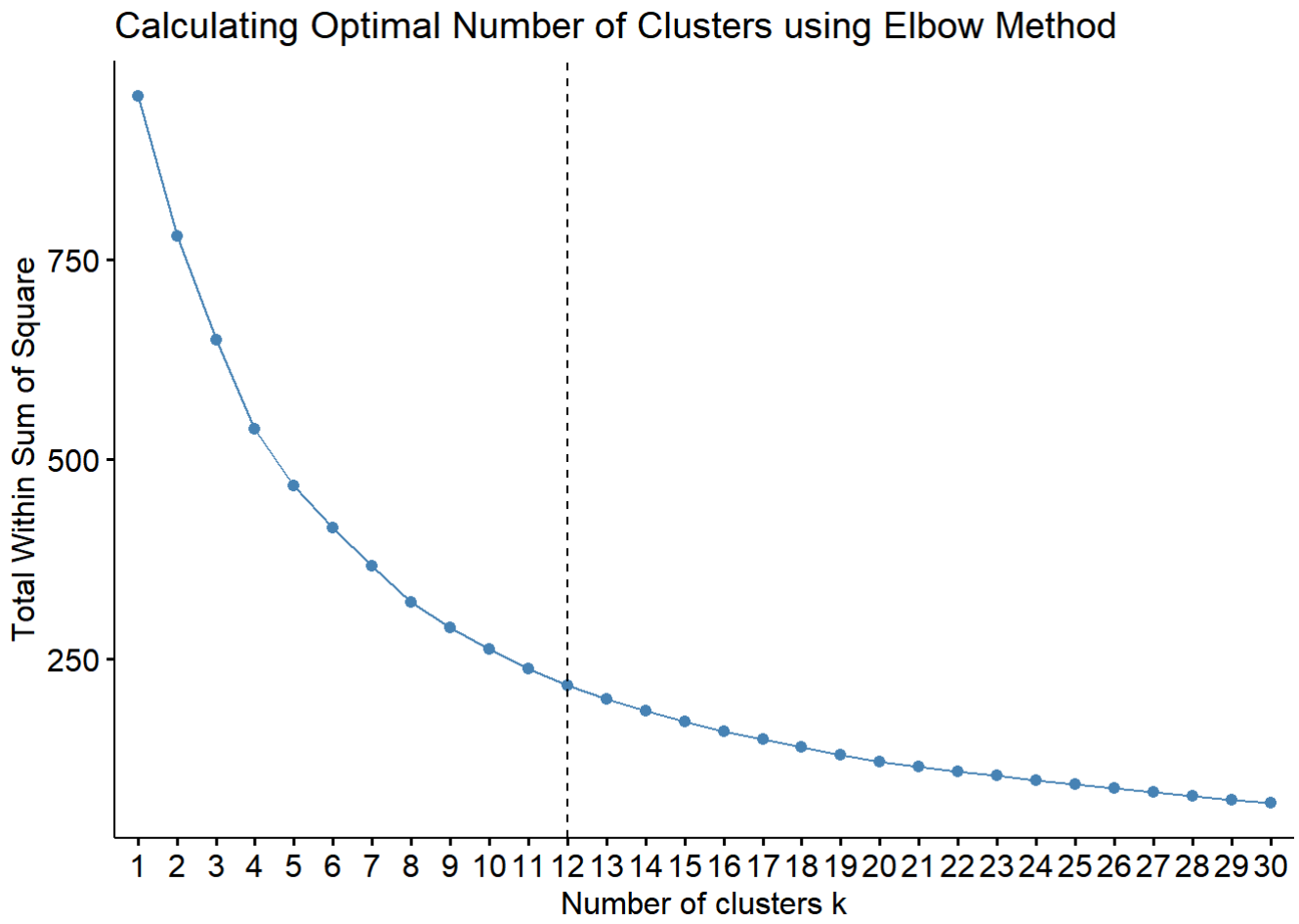
How many clusters would you choose? Figuring out the optimal of clusters we can make with this data. Applying silhoutte method to figure out the number of optimal clusters.

```
# Calculate the optimal number of clusters for the dataset using the within-cluster sum of sq
uares (WCSS) method
# using the fviz_nbclust() function from the factoextra package
fviz_nbclust(cleansed_data[ , c(4:16)], hcut, method = "silhouette", k.max =
30) +
 labs(title = "Calculating Optimal Number of Clusters using silhouette
Method")
```



Applying Elbow method to figure out the number of optimal clusters.

```
# Determine the optimal number of clusters for the dataset using the Elbow method
fviz_nbclust(cleansed_data[ , c(4:16)], hcut, method = "wss", k.max = 30) +
 labs(title = "Calculating Optimal Number of Clusters using Elbow Method") +
 geom_vline(xintercept = 12, linetype = 2)
```



Based on both the methods we can conclude that 12 can be the optimal number of clusters for this problem. Therefore, Clustering the data into 12 using Ward linkage method. Plotting the results

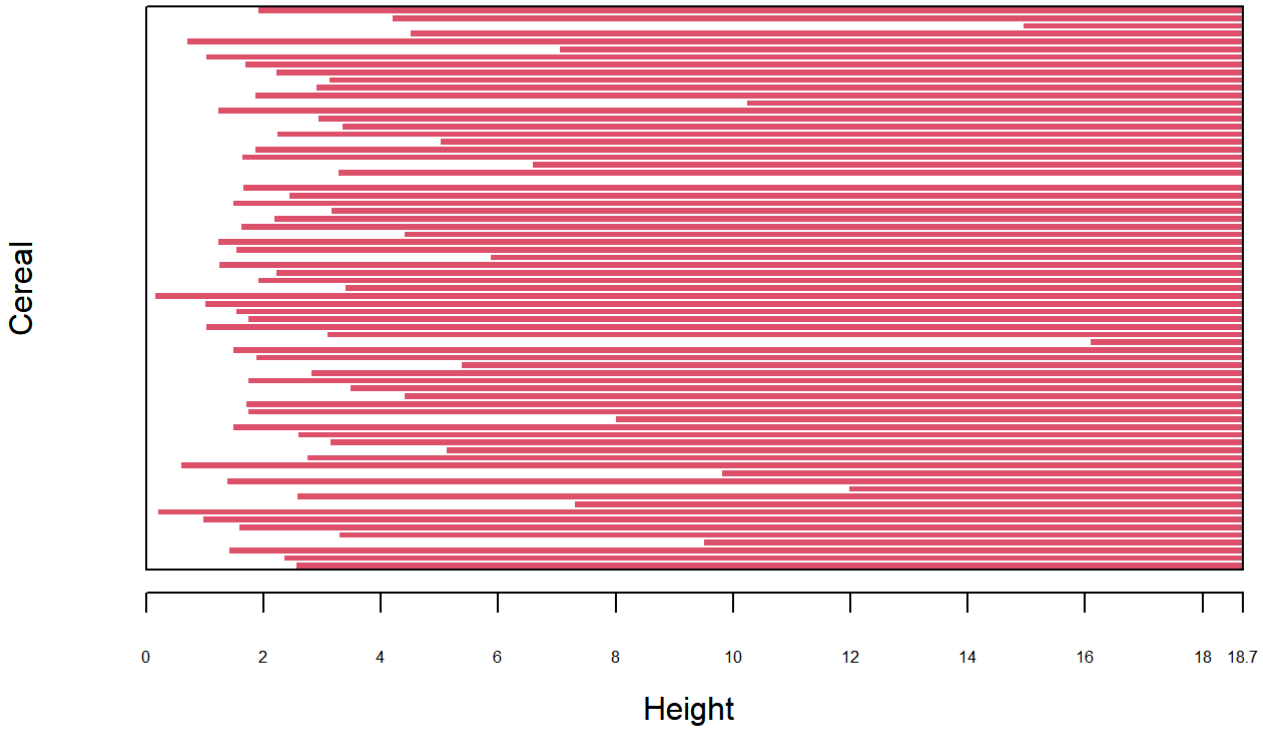
```
plot(clustering_H_WARD,
 main = "Clustering(12) AGNES using Ward linkage",
 xlab = "Height",
 ylab = "Cereal",
 cex.axis = 0.5,
 cex = 0.50,
 hang = -1)
```

```
## 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
```

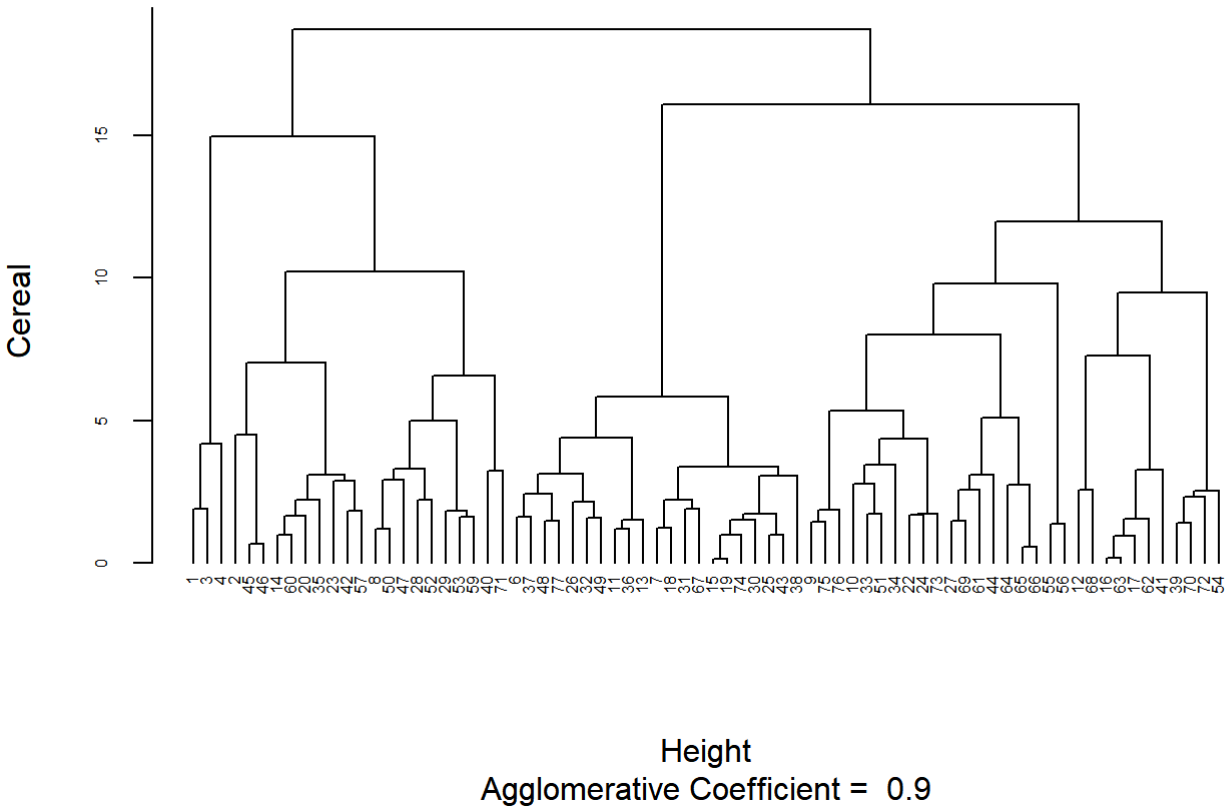
```
## Warning in axis(1, at = at.vals, labels = lab.vals, ...): "hang" is not a
## graphical parameter
```

Clustering(12) AGNES using Ward linkage



Agglomerative Coefficient = 0.9

Clustering(12) AGNES using Ward linkage



Agglomerative Coefficient = 0.9

```
## 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
```

Hence, the cereal data may be optimally grouped into 12 clusters using AGNE's ward linkage approach, with an agglomerative coefficent of about 0.9 for hierarchical clustering.

TASK - 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. To do this: ● Cluster partition A ● Use the cluster centroids from A to assign each record in partition B (each record is assigned to the cluster with the closest centroid). ● Assess how consistent the cluster assignments are compared to the assignments based on all the data.

```
# dividing the tree into 12 clusters
Clusters_ward_12 <- cutree(clustering_H_WARD, k = 12)
#Adding the cluster to the cleansed dataset
cereal_preprocessed_1 <- cbind(cluster = Clusters_ward_12, cleansed_data)
```

Splitting the data and making 70% and 30% partitions to find out the stability of the clusters.

```
# Set the seed for randomized functions
set.seed(111319)
# Split the data into 70% partition A and 30% partition B
cerealIndex <- createDataPartition(cleansed_data$protein, p=0.3, list = F)
cereal_preprocessed_PartitionB <- cleansed_data[cerealIndex, ]
cereal_preprocessed_PartitionA <- cleansed_data[-cerealIndex,]
```

Re-doing the clustering with the partitioned data

```
# calculating the euclidean distances for the partitioned data
Euclidean_dist_A <- dist(cereal_preprocessed_PartitionA[ ,c(4:16)], method ="euclidean")
#Euclidean_dist_A
```

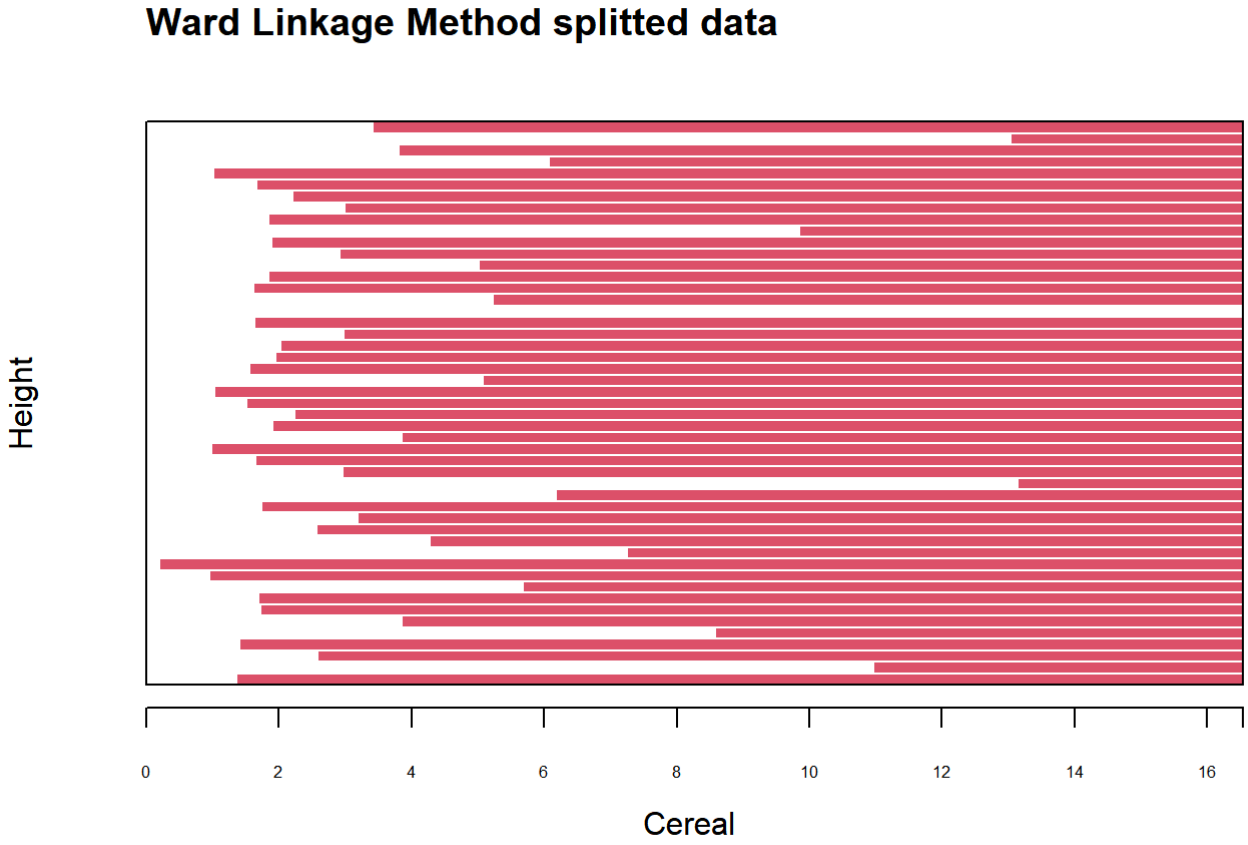
PERFORMING WARD METHOD IN CLUSTERING THE PARTITIONED DATA

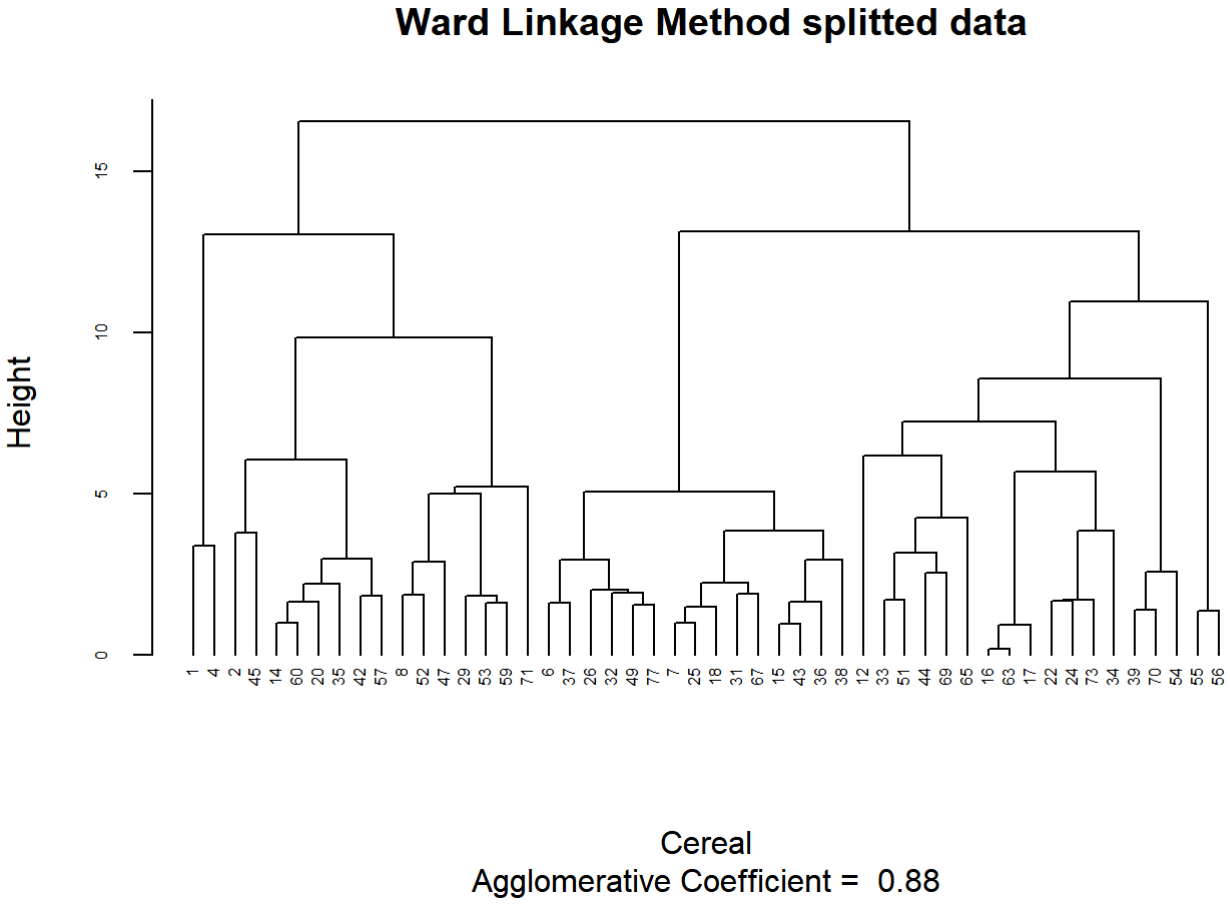
```
#performing AGNES uisng ward linkage method
Clustering_H_ward_A <- agnes(Euclidean_dist_A, method = "ward")
plot(Clustering_H_ward_A,
 main = " Ward Linkage Method splitted data",
 xlab = "Cereal",
 ylab = "Height",
 cex.axis = 0.5,
 cex = 0.50,
 hang = -1)
```

```
## 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
```


Ward Linkage Method splitted data
Height
0
2
4
6
8
10
12
14
16
Cereal
Agglomerative Coefficient = 0.88


Ward Linkage Method splitted data
Height
0
5
10
15
1 4 2 45 14 60 20 35 42 57 8 52 47 29 53 59 71 6 37 26 32 49 77 7 25 18 31 67 15 43 36 38 12 33 51 44 69 65 16 63 17 22 24 73 34 39 70 54 55 56
Cereal
Agglomerative Coefficient = 0.88

```
## 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
```

Dividing the tree into clusters for Analysis

```
Clusters_ward_12_A <- cutree(Clustering_H_ward_A, k = 12)
```

```
#clubbing the cluster to the pre-processed dataset
cereal_preprocessed_A <- cbind(cluster = Clusters_ward_12_A,
cereal_preprocessed_PartitionA)
cereal_preprocessed_A
```

```
##    cluster                           name mfr type   calories    protein
## 1        1                      100%_Bran   N    C -1.8929836  1.3286071
## 2        2                100%_Natural_Bran   Q    C  0.6732089  0.4151897
## 4        1        All-Bran_with_Extra_Fiber   K    C -2.9194605  1.3286071
## 6        3          Apple_Cinnamon_Cheerios   G    C  0.1599704 -0.4982277
## 7        3                       Apple_Jacks   K    C  0.1599704 -0.4982277
## 8        4                          Basic_4   G    C  1.1864474  0.4151897
## 12       5                         Cheerios   G    C  0.1599704  3.1554419
## 14       6                         Clusters   G    C  0.1599704  0.4151897
## 15       3                      Cocoa_Puffs   G    C  0.1599704 -1.4116451
## 16       7                        Corn_Chex   R    C  0.1599704 -0.4982277
## 17       7                      Corn_Flakes   K    C -0.3532681 -0.4982277
## 18       3                        Corn_Pops   K    C  0.1599704 -1.4116451
## 20       6              Cracklin'_Oat_Bran   K    C  0.1599704  0.4151897
## 22       8                          Crispix   K    C  0.1599704 -0.4982277
## 24       8                      Double_Chex   R    C -0.3532681 -0.4982277
## 25       3                      Froot_Loops   K    C  0.1599704 -0.4982277
## 26       3                    Frosted_Flakes   K    C  0.1599704 -1.4116451
## 29       4                     Fruitful_Bran   K    C  0.6732089  0.4151897
## 31       3                     Golden_Crisp   P    C -0.3532681 -0.4982277
## 32       3                    Golden_Grahams   G    C  0.1599704 -1.4116451
## 33       9                 Grape_Nuts_Flakes   P    C -0.3532681  0.4151897
## 34       8                       Grape-Nuts   P    C  0.1599704  0.4151897
## 35       6                Great_Grains_Pecan   P    C  0.6732089  0.4151897
## 36       3                  Honey_Graham_Ohs   Q    C  0.6732089 -1.4116451
## 37       3                Honey_Nut_Cheerios   G    C  0.1599704  0.4151897
## 38       3                         Honey-comb   P    C  0.1599704 -1.4116451
## 39      10      Just_Right_Crunchy__Nuggets   K    C  0.1599704 -0.4982277
## 42       6                              Life   Q    C -0.3532681  1.3286071
## 43       3                      Lucky_Charms   G    C  0.1599704 -0.4982277
## 44       9                             Maypo   A    H -0.3532681  1.3286071
## 45       2 Muesli_Raisins,_Dates,_&_Almonds   R    C  2.2129244  1.3286071
## 47       4             Mueslix_Crispy_Blend   K    C  2.7261629  0.4151897
## 49       3                 Nut&Honey_Crunch   K    C  0.6732089 -0.4982277
## 51       9                Nutri-grain_Wheat   K    C -0.8665066  0.4151897
## 52       4             Oatmeal_Raisin_Crisp   G    C  1.1864474  0.4151897
## 53       4           Post_Nat._Raisin_Bran   P    C  0.6732089  0.4151897
## 54      10                        Product_19   K    C -0.3532681  0.4151897
## 55      11                       Puffed_Rice   Q    C -2.9194605 -1.4116451
## 56      11                      Puffed_Wheat   Q    C -2.9194605 -0.4982277
## 57       6               Quaker_Oat_Squares   Q    C -0.3532681  1.3286071
## 59       4                       Raisin_Bran   K    C  0.6732089  0.4151897
## 60       6                  Raisin_Nut_Bran   G    C -0.3532681  0.4151897
## 63       7                    Rice_Krispies   K    C  0.1599704 -0.4982277
## 65       9           Shredded_Wheat_'n'Bran   N    C -0.8665066  0.4151897
## 67       3                           Smacks   K    C  0.1599704 -0.4982277
## 69       9          Strawberry_Fruit_Wheats   N    C -0.8665066 -0.4982277
## 70      10                Total_Corn_Flakes   G    C  0.1599704 -0.4982277
## 71      12                Total_Raisin_Bran   G    C  1.6996859  0.4151897
## 73       8                          Triples   G    C  0.1599704 -0.4982277
## 77       3              Wheaties_Honey_Gold   G    C  0.1599704 -0.4982277
##             fat     sodium       fiber       carbo       sugars      potass
## 1  -0.01290349 -0.3539844  3.29284661 -2.50878291 -0.234390576  2.57536849
## 2   3.96137277 -1.7257708 -0.06375361 -1.74099432  0.222370547  0.51602052
## 4  -1.00647256 -0.2346986  4.97114672 -1.74099432 -1.604673946  3.28548848
```

```
## 6   0.98066557  0.2424445 -0.27354112 -1.10117049  0.679131670 -0.40713546
## 7  -1.00647256 -0.4136273 -0.48332864 -0.97320572  1.592653916 -0.97523145
## 8   0.98066557  0.6003018 -0.06375361  0.81830100  0.222370547  0.01893653
## 12  0.98066557  1.5545879 -0.06375361  0.56237147 -1.376293384  0.08994853
## 14  0.98066557 -0.2346986 -0.06375361 -0.46134666 -0.006010015  0.08994853
## 15 -0.01290349  0.2424445 -0.90290366 -0.71727619  1.364273355 -0.62017146
## 16 -1.00647256  1.4353022 -0.90290366  1.84201913 -0.919532261 -1.04624345
## 17 -1.00647256  1.5545879 -0.48332864  1.58608960 -1.147912823 -0.90421945
## 18 -1.00647256 -0.8311275 -0.48332864 -0.46134666  1.135892793 -1.11725545
## 20  1.97423464 -0.2346986  0.77539645 -1.22913525 -0.006010015  0.87108052
## 22 -1.00647256  0.7195875 -0.48332864  1.58608960 -0.919532261 -0.97523145
## 24 -1.00647256  0.3617302 -0.48332864  0.81830100 -0.462771138 -0.26511146
## 25 -0.01290349 -0.4136273 -0.48332864 -0.97320572  1.364273355 -0.97523145
## 26 -1.00647256  0.4810160 -0.48332864 -0.20541712  0.907512232 -1.04624345
## 29 -1.00647256  0.9581591  1.19497147 -0.20541712  1.135892793  1.29715251
## 31 -1.00647256 -1.3679135 -0.90290366 -0.97320572  1.821034478 -0.83320745
## 32 -0.01290349  1.4353022 -0.90290366  0.05051241  0.450751108 -0.76219545
## 33 -0.01290349 -0.2346986  0.35582142  0.05051241 -0.462771138 -0.19409946
## 34 -1.00647256  0.1231587  0.35582142  0.56237147 -0.919532261 -0.12308746
## 35  1.97423464 -1.0100561  0.35582142 -0.46134666 -0.691151699  0.01893653
## 36  0.98066557  0.7195875 -0.48332864 -0.71727619  0.907512232 -0.76219545
## 37 -0.01290349  1.0774449 -0.27354112 -0.84524095  0.679131670 -0.12308746
## 38 -1.00647256  0.2424445 -0.90290366 -0.20541712  0.907512232 -0.90421945
## 39 -0.01290349  0.1231587 -0.48332864  0.56237147 -0.234390576 -0.54915946
## 42  0.98066557 -0.1154129 -0.06375361 -0.71727619 -0.234390576 -0.05207547
## 43 -0.01290349  0.2424445 -0.90290366 -0.71727619  1.135892793 -0.62017146
## 44 -0.01290349 -1.9046994 -0.90290366  0.30644194 -0.919532261 -0.05207547
## 45  1.97423464 -0.7714846  0.35582142  0.30644194  0.907512232  1.01310452
## 47  0.98066557 -0.1154129  0.35582142  0.56237147  1.364273355  0.87108052
## 49 -0.01290349  0.3617302 -0.90290366  0.05051241  0.450751108 -0.83320745
## 51 -1.00647256  0.1231587  0.35582142  0.81830100 -1.147912823 -0.12308746
## 52  0.98066557  0.1231587 -0.27354112 -0.33338189  0.679131670  0.30298453
## 53 -0.01290349  0.4810160  1.61454650 -0.97320572  1.592653916  2.29132050
## 54 -1.00647256  1.9124453 -0.48332864  1.33016007 -0.919532261 -0.76219545
## 55 -1.00647256 -1.9046994 -0.90290366 -0.46134666 -1.604673946 -1.18826745
## 56 -1.00647256 -1.9046994 -0.48332864 -1.22913525 -1.604673946 -0.69118346
## 57 -0.01290349 -0.2943415 -0.06375361 -0.20541712 -0.234390576  0.16096053
## 59 -0.01290349  0.6003018  1.19497147 -0.20541712  1.135892793  2.00727250
## 60  0.98066557 -0.2346986  0.14603391 -1.10117049  0.222370547  0.58703252
## 63 -1.00647256  1.5545879 -0.90290366  1.84201913 -0.919532261 -0.90421945
## 65 -1.00647256 -1.9046994  0.77539645  1.07423054 -1.604673946  0.58703252
## 67 -0.01290349 -1.0696990 -0.48332864 -1.48506478  1.821034478 -0.83320745
## 69 -1.00647256 -1.7257708  0.35582142  0.05051241 -0.462771138 -0.12308746
## 70 -0.01290349  0.4810160 -0.90290366  1.58608960 -0.919532261 -0.90421945
## 71 -0.01290349  0.3617302  0.77539645  0.05051241  1.592653916  1.86524850
## 73 -0.01290349  1.0774449 -0.90290366  1.58608960 -0.919532261 -0.54915946
## 77 -0.01290349  0.4810160 -0.48332864  0.30644194  0.222370547 -0.54915946
##      vitamins      shelf     weight       cups      rating
## 1  -0.1453172  0.9515734 -0.1967771 -2.1100340  1.83218758
## 2  -1.2642598  0.9515734 -0.1967771  0.7690100 -0.61805706
## 4  -0.1453172  0.9515734 -0.1967771 -1.3795303  3.63338491
## 6  -0.1453172 -1.4507595 -0.1967771 -0.3052601 -0.93656251
## 7  -0.1453172 -0.2495930 -0.1967771  0.7690100 -0.67568989
## 8  -0.1453172  0.9515734  1.9962520 -0.3052601 -0.40058570
## 12 -0.1453172 -1.4507595 -0.1967771  1.8432802  0.57657347
## 14 -0.1453172  0.9515734 -0.1967771 -1.3795303 -0.16127646
```

```
## 15 -0.1453172 -0.2495930 -0.1967771  0.7690100 -1.41872637
## 16 -0.1453172 -1.4507595 -0.1967771  0.7690100 -0.08689833
## 17 -0.1453172 -1.4507595 -0.1967771  0.7690100  0.22763247
## 18 -0.1453172 -0.2495930 -0.1967771  0.7690100 -0.48998167
## 20 -0.1453172  0.9515734 -0.1967771 -1.3795303 -0.15781928
## 22 -0.1453172  0.9515734 -0.1967771  0.7690100  0.30112138
## 24 -0.1453172  0.9515734 -0.1967771 -0.3052601  0.11853896
## 25 -0.1453172 -0.2495930 -0.1967771  0.7690100 -0.74449406
## 26 -0.1453172 -1.4507595 -0.1967771 -0.3052601 -0.79942345
## 29 -0.1453172  0.9515734  1.9962520 -0.6490266 -0.11747555
## 31 -0.1453172 -1.4507595 -0.1967771  0.2533603 -0.52773607
## 32 -0.1453172 -0.2495930 -0.1967771 -0.3052601 -1.34272615
## 33 -0.1453172  0.9515734 -0.1967771  0.2533603  0.66996501
## 34 -0.1453172  0.9515734 -0.1967771 -2.4538004  0.76209027
## 35 -0.1453172  0.9515734 -0.1967771 -2.1100340  0.22395859
## 36 -0.1453172 -0.2495930 -0.1967771  0.7690100 -1.48031505
## 37 -0.1453172 -1.4507595 -0.1967771 -0.3052601 -0.82531855
## 38 -0.1453172 -1.4507595 -0.1967771  2.1870466 -0.99117283
## 39  3.2115106  0.9515734 -0.1967771  0.7690100 -0.43723896
## 42 -0.1453172 -0.2495930 -0.1967771 -0.6490266  0.18952903
## 43 -0.1453172 -0.2495930 -0.1967771  0.7690100 -1.13411138
## 44 -0.1453172 -0.2495930 -0.1967771  0.7690100  0.86744227
## 45 -0.1453172  0.9515734 -0.1967771  0.7690100 -0.39358784
## 47 -0.1453172  0.9515734  3.1259942 -0.6490266 -0.87934079
## 49 -0.1453172 -0.2495930 -0.1967771 -0.6490266 -0.90703767
## 51 -0.1453172  0.9515734 -0.1967771  0.7690100  1.20857002
## 52 -0.1453172  0.9515734  1.4646086 -1.3795303 -0.86955299
## 53 -0.1453172  0.9515734  1.9962520 -0.6490266 -0.34349055
## 54  3.2115106  0.9515734 -0.1967771  0.7690100 -0.08273233
## 55 -1.2642598  0.9515734 -3.5195485  0.7690100  1.28782197
## 56 -1.2642598  0.9515734 -3.5195485  0.7690100  1.44796198
## 57 -0.1453172  0.9515734 -0.1967771 -1.3795303  0.48736586
## 59 -0.1453172 -0.2495930  1.9962520 -0.3052601 -0.24250288
## 60 -0.1453172  0.9515734 -0.1967771 -1.3795303 -0.21088091
## 63 -0.1453172 -1.4507595 -0.1967771  0.7690100 -0.14988985
## 65 -1.2642598 -1.4507595 -0.1967771 -0.6490266  2.26429773
## 67 -0.1453172 -0.2495930 -0.1967771 -0.3052601 -0.81408243
## 69 -0.1453172 -0.2495930 -0.1967771  0.7690100  1.18871964
## 70  3.2115106  0.9515734 -0.1967771  0.7690100 -0.27236281
## 71  3.2115106  0.9515734  3.1259942  0.7690100 -1.00182464
## 73 -0.1453172  0.9515734 -0.1967771 -0.3052601 -0.25339630
## 77 -0.1453172 -1.4507595 -0.1967771 -0.3052601 -0.46116700
```

Calculating the centroids for figuring out the closest point for all the elemnts in B

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

Calculating the distance between both centers of A and B

```
# Calculate the distance between the centers of partition A and the values of partition B
B_to_A_centers <- dist(centeroids_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 comparision 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 distributed using both methods (full data and partitioned data), we can compare the number of matching assignments to ascertain the stability of the clusters.

```
sum(cereal_preprocessed_1$cluster == cereal_preprocessed_2$cluster)
```

```
## [1] 33
```

```
## [1] 33
```

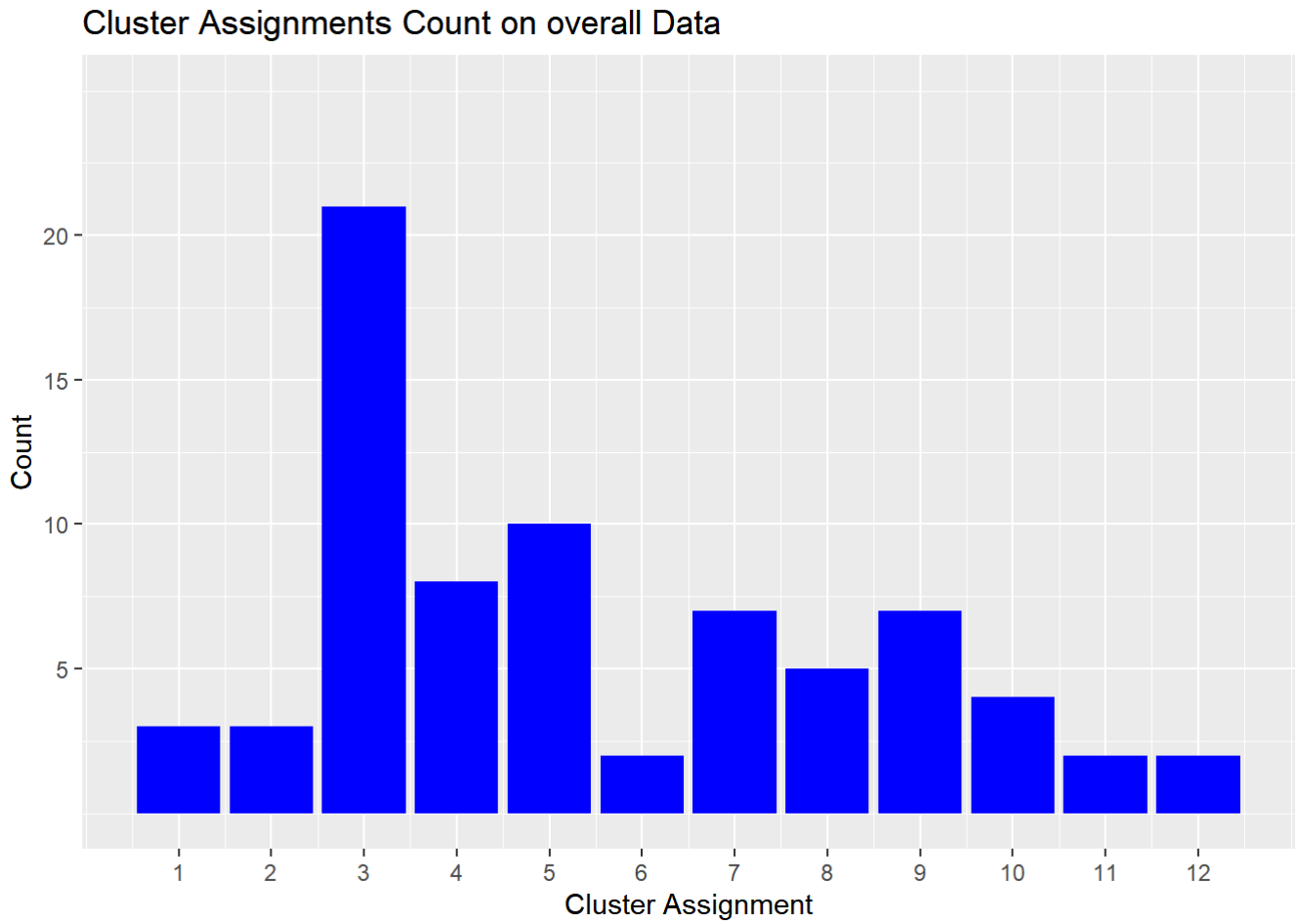
This investigation indicates that the clusters are not very stable. When just 70% of the data were available, the assignments for only 33 of the 74 observations were identical. Consequently, the assignment's repeatability is at 44.59%.

PLOTTING THE HIERARCHIAL CLUSTERING ALGORITHM

```
# 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 = "BLUE") +
 labs(title="Cluster Assignments Count on overall 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.
```

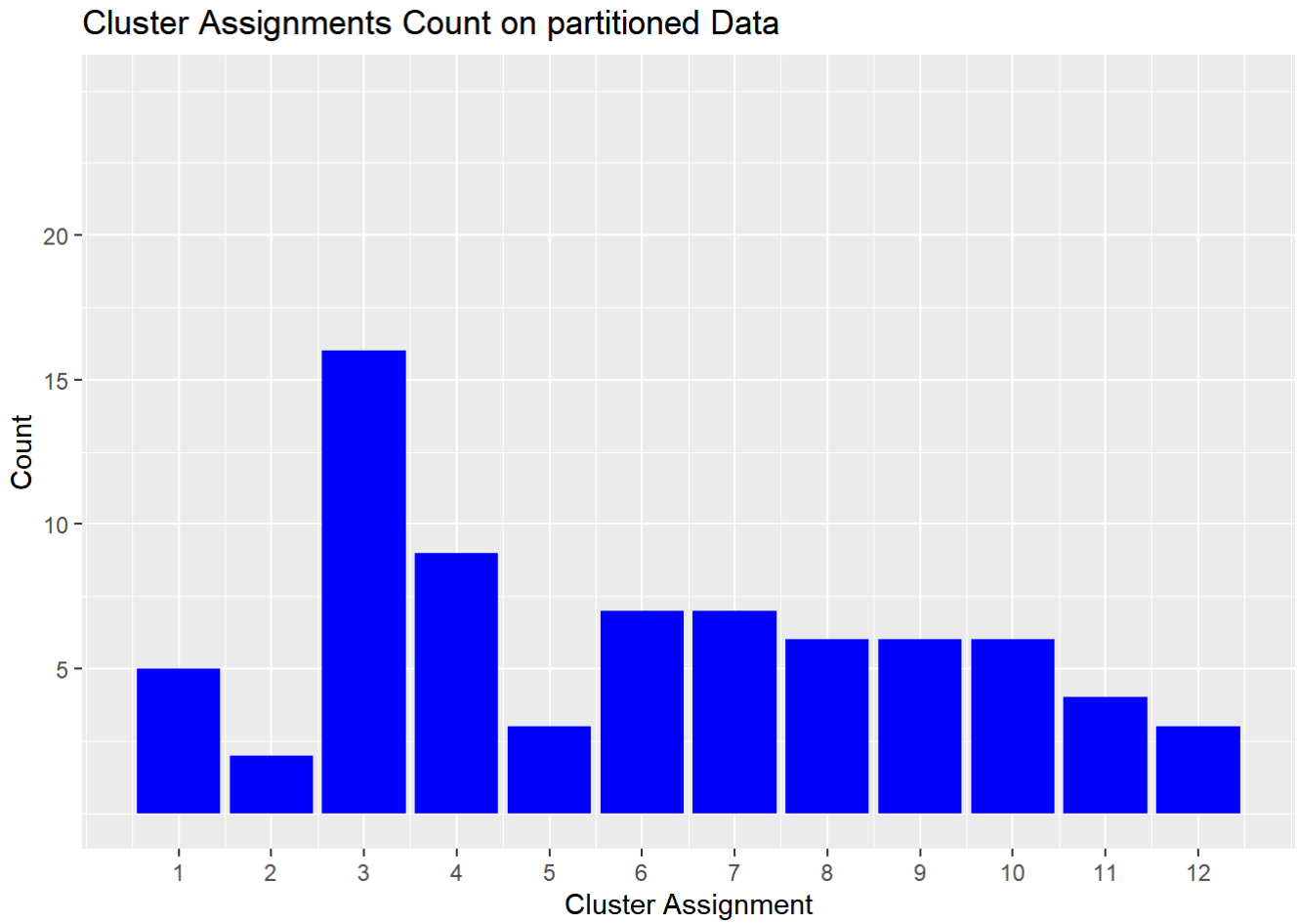
```
## Warning: Use of `cereal_preprocessed_1$cluster` is discouraged.
## ℹ Use `cluster` instead.
```




```
## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as
## of ggplot2 3.3.4.
```

```
# Plot the graph for the partitioned data
ggplot(data = cereal_preprocessed_2, aes(cereal_preprocessed_2$cluster)) +
 geom_bar(fill = "BLUE") +
 labs(title="Cluster Assignments Count on 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))
```

```
## Warning: Use of `cereal_preprocessed_2$cluster` is discouraged.
## ℹ Use `cluster` instead.
```



Visually, it is evident that once the data was divided, Cluster 3 shrank significantly. As a result, the size of some of the other clusters increased. The graphic suggests that when the data is partitioned, the clusters are distributed more equally among the 12 clusters.

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

In this case, normalising the data wouldn’t be appropriate. It would not be appropriate since the nutritional information for cereal must be scaled or normalised based on the cereal sample that is being assessed. Because of this, the dataset that was gathered could only include cereals with a very high sugar content and minimal amounts of fibre, iron, or other nutrients. It is difficult to forecast how much nutrition a child will receive from cereal after it has been scaled or standardised throughout the sample set. A cereal with an iron content of 0.999 can lead an uninformed observer to believe that it provides almost all of the nutritive iron a kid needs, but it might also be the worst cereal in the sample set, having very little nutritious value.

Therefore, a more appropriate method to preprocess the data would be to make it a ratio to the daily needed quantities of calories, fibre, carbohydrates, etc. for a youngster. This would allow analysts to make better conclusions about the clusters during evaluation by preventing a small number of important elements from overriding the distance calculations. When analysing the clusters, an analyst may determine how much of a student’s required daily intake of nutrients would come from XX cereal by looking at the cluster average. After that, the employees would have the information to select from a range of “healthy” cereal clusters.

thus, As the measurements for various nutrients may be on different scales, it is advised to normalise the data before doing a cluster analysis.Normalisation will guarantee that the weight of every nutrient in the analysis is the same. One method for identifying a cluster of “healthy cereals” would be to select a cluster of cereals with low ratings for nutrients like sugar and salt and high ratings for nutrients like fibre, protein, and vitamins. When

choosing cereals for the cafeteria, store displays and customer reviews could also be helpful factors to take into account. After the data has been normalised, you may use the chosen technique (Agnes with complete linkage, for example) to do hierarchical clustering. After then, you may look at the clusters that are formed to find a collection of grains that match By dividing the data into two parts and evaluating how well the clusters created based on one part apply to the other, you may further assess the stability of the clusters.