Heirarchial Clustering

David Slusser

12/6/2020

R Markdown

This is an R Markdown file for the fifth homework assignment. This is looking at a consumer ratings for 77 breakfast cereals. Use hierarchical clusting model and comparing it to K-Means

```
# Load in the data packages that are needed for hierarchical clusting and agnes
library(readr) # To load in csv data files
library(cluster) # For agnes function and compare clustering
```

```
## Warning: package 'cluster' was built under R version 4.0.2
```

```
library(dplyr) # For standardized
```

```
## Warning: package 'dplyr' was built under R version 4.0.2
```

```
##
## 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
```

```
# First load in the cereal data set
# It is a csv file located in my Machine Learning course folder
Cereals <- read_csv("~/Desktop/School/Graduate/Machine Learning/HW/Data/Cereals.csv")
```

```
## Parsed with column specification:
## cols(
##   name = col_character(),
##   mfr = col_character(),
##   type = col_character(),
##   calories = col_double(),
##   protein = col_double(),
##   fat = col_double(),
##   sodium = col_double(),
##   fiber = col_double(),
##   carbo = col_double(),
##   sugars = col_double(),
##   potass = col_double(),
##   vitamins = col_double(),
##   shelf = col_double(),
##   weight = col_double(),
##   cups = col_double(),
##   rating = col_double()
## )
```

```
head(Cereals) # Examine the Cereals dataset to look at the variables included
```

```
## # A tibble: 6 x 16
##   name  mfr   type  calories protein   fat sodium fiber carbo sugars potass
##   <chr> <chr> <chr>    <dbl>   <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl>  <dbl>
## 1 100%… N     C           70       4     1    130  10     5        6    280
## 2 100%… Q     C          120       3     5     15   2     8        8    135
## 3 All-… K     C           70       4     1    260   9     7        5    320
## 4 All-… K     C           50       4     0    140  14     8        0    330
## 5 Almo… R     C          110       2     2    200   1    14        8     NA
## 6 Appl… G     C          110       2     2    180   1.5  10.5     10     70
## # … with 5 more variables: vitamins <dbl>, shelf <dbl>, weight <dbl>,
## #   cups <dbl>, rating <dbl>
```

```
Cereals <- na.omit(Cereals) # Remove the missing values
head(Cereals) # Look at the Cereals dataset to examine the missing values
```

```
## # A tibble: 6 x 16
##   name  mfr   type  calories protein   fat sodium fiber carbo sugars potass
##   <chr> <chr> <chr>    <dbl>   <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl>  <dbl>
## 1 100%… N     C           70       4     1    130  10     5        6    280
## 2 100%… Q     C          120       3     5     15   2     8        8    135
## 3 All-… K     C           70       4     1    260   9     7        5    320
## 4 All-… K     C           50       4     0    140  14     8        0    330
## 5 Appl… G     C          110       2     2    180   1.5  10.5     10     70
## 6 Appl… K     C          110       2     0    125   1    11       14     30
## # … with 5 more variables: vitamins <dbl>, shelf <dbl>, weight <dbl>,
## #   cups <dbl>, rating <dbl>
```

```
# We need to scale the data instead of using arbitraty units
# (for instance, calories and proteins are different scales so we want to be able to com
pare and thus
# need to standardize)

Cereals.norm <- Cereals %>%
    as_tibble() %>%
    mutate(across(where(is.numeric), scale))

# Get the Euclidean distance measurement
distance <- dist(Cereals.norm[4:16], method = "euclidean")

# Now we're going to to use the agnes function to compute the different linkage methods
# We want to use single, complete and average linkage along with Ward
# We use the cluster package for computing the linkage methods

hc_single <- agnes(Cereals.norm[4:16], method = "single")
hc_complete <- agnes(Cereals.norm[4:16], method = "complete")
hc_average <- agnes(Cereals.norm[4:16], method = "average")
hc_ward <- agnes(Cereals.norm[4:16], method = "ward")

# Now we want to see the results of the coefficients
# The best method has the lowest coefficient
print(hc_single) # 0.607
```

```
## Call:     agnes(x = Cereals.norm[4:16], method = "single")
## Agglomerative coefficient:  0.6067859
## Order of objects:
##  [1]  1  3  4  2  5 35  6 14 18 71 41 23 28 17 10 34 12 64 46 74 47  8 72 73 30
## [26] 24 29 36  7 48 50 26 27 51 56 13 57 19 55 33 40 21 31 49 20 22 70 32 15 60
## [51] 16 59  9 25 66 58 42 61 62 63 39 45 11 65 43 44 37 67 69 52 38 68 53 54
## Height (summary):
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1431  1.3777  1.7695  1.8668  2.2787  4.0361
##
## Available components:
## [1] "order"  "height" "ac"     "merge"  "diss"   "call"   "method" "data"
```

```
print(hc_complete) # 0.835
```

```
## Call:     agnes(x = Cereals.norm[4:16], method = "complete")
## Agglomerative coefficient:  0.8353712
## Order of objects:
##  [1]  1  3  4  2 25 66 58 42 61 62 63 53 54  5 35 46 74 24 30 47 10 34 12  6 17
## [26] 14 18 71 28 23 41 29 64 36  8 72 73  9 31 49 32 13 57 19 33 21 40 55 11 65
## [51] 15 60 16 59 39 20 22 70 37 67 69 52  7 48 45 26 50 43 44 27 51 56 38 68
## Height (summary):
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1431  1.6076  2.3389  2.9321  3.7169 10.9839
##
## Available components:
## [1] "order"  "height" "ac"     "merge"  "diss"   "call"   "method" "data"
```

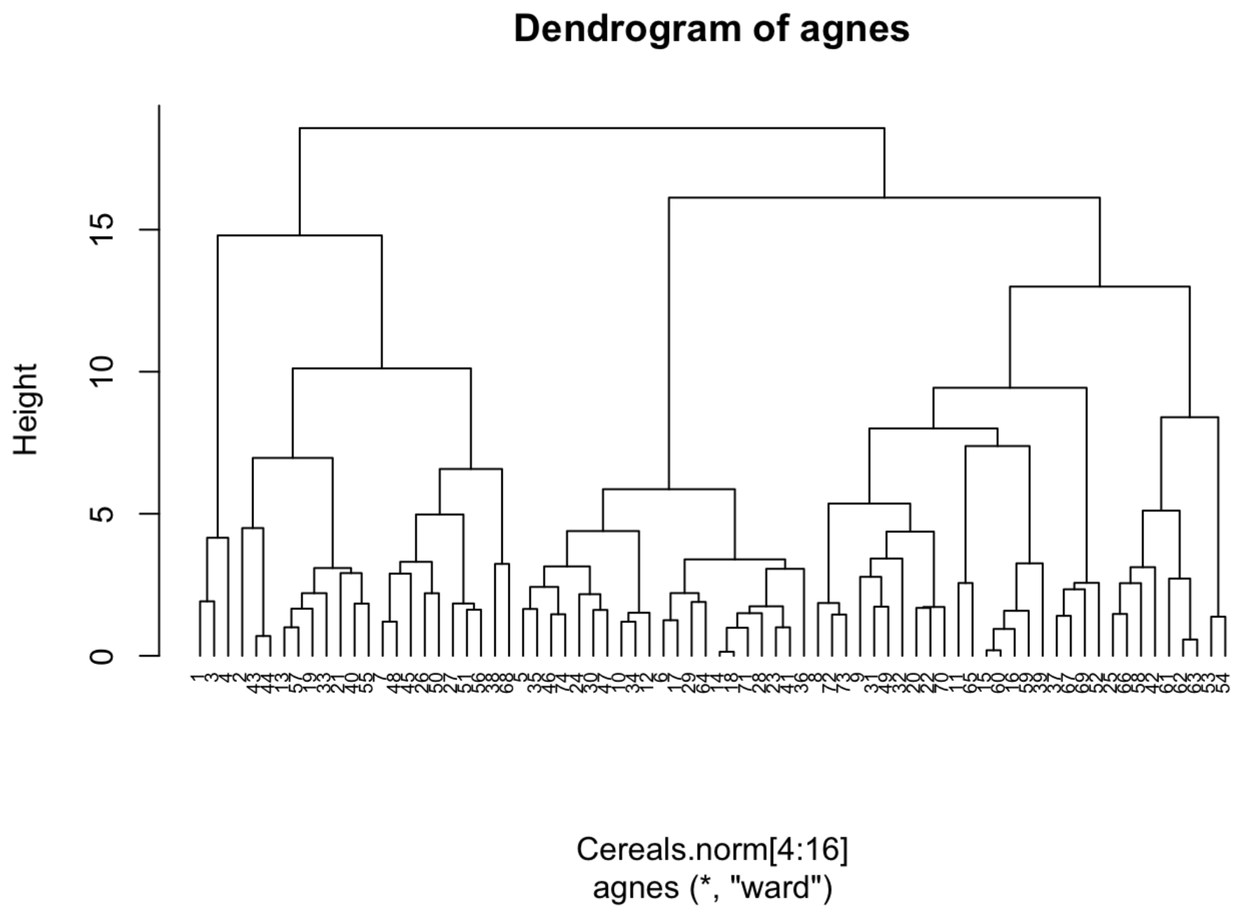
```
print(hc_average) # 0.777
```

```
## Call:     agnes(x = Cereals.norm[4:16], method = "average")
## Agglomerative coefficient:  0.7766075
## Order of objects:
##  [1]  1  3  4  2  5 35 46 74 24 30 47  6 17 14 18 71 23 41 28 29 64 10 34 12 36
## [26]  8 72 73  9 32 20 22 70 31 49 13 57 19 33 40 55 21 15 60 16 59 39 25 66 58
## [51] 42 61 62 63  7 48 50 45 26 27 51 56 43 44 37 67 69 52 38 68 11 65 53 54
## Height (summary):
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1431  1.4633  2.0666  2.4461  2.9445  7.7243
##
## Available components:
## [1] "order"  "height" "ac"     "merge"  "diss"   "call"   "method" "data"
```

```
print(hc_ward) # 0.904
```

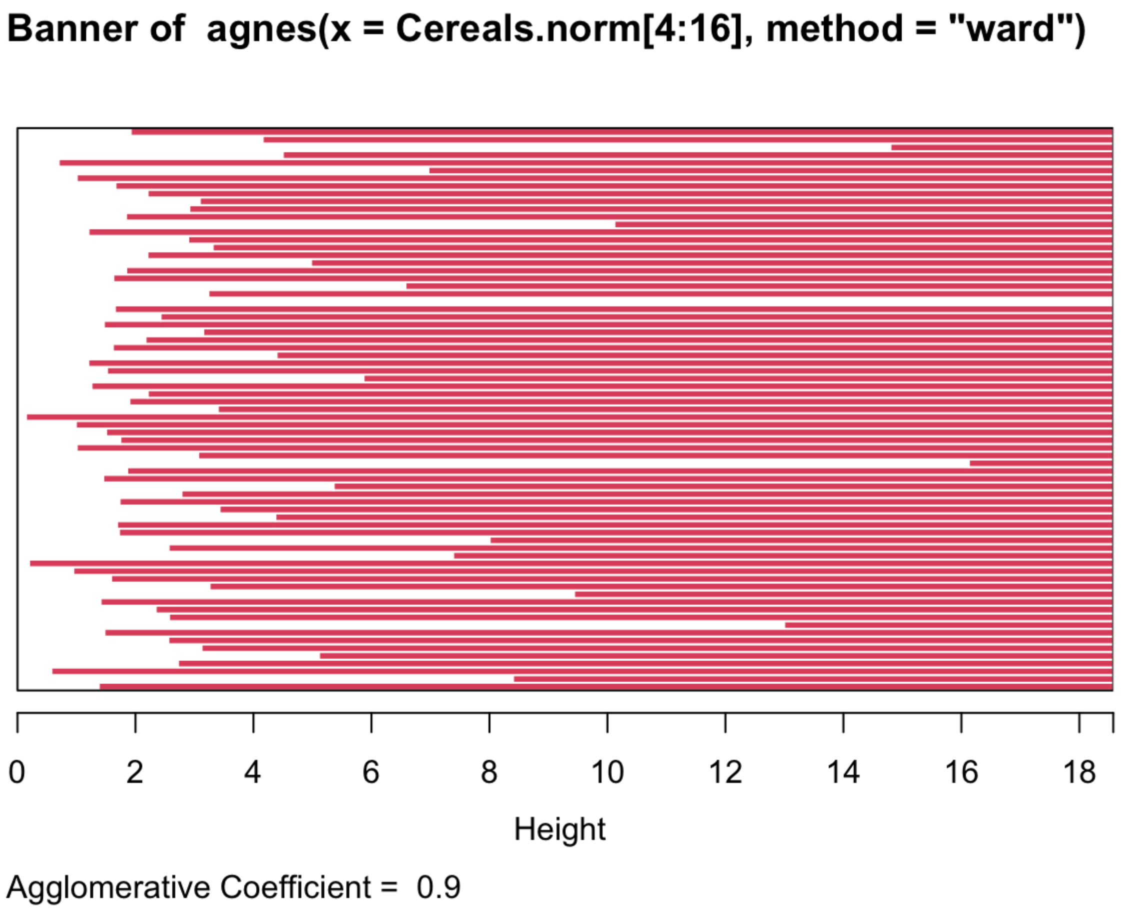
```
## Call:     agnes(x = Cereals.norm[4:16], method = "ward")
## Agglomerative coefficient:  0.9046042
## Order of objects:
##  [1]  1  3  4  2 43 44 13 57 19 33 21 40 55  7 48 45 26 50 27 51 56 38 68  5 35
## [26] 46 74 24 30 47 10 34 12  6 17 29 64 14 18 71 28 23 41 36  8 72 73  9 31 49
## [51] 32 20 22 70 11 65 15 60 16 59 39 37 67 69 52 25 66 58 42 61 62 63 53 54
## Height (summary):
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1431  1.5858  2.3422  3.6092  4.1559 18.5749
##
## Available components:
## [1] "order"  "height" "ac"     "merge"  "diss"   "call"   "method" "data"
```

```
# Let us visualize the ward linkage since it has the highest
pltree(hc_ward, cex = 0.6, hang = -1, main = "Dendrogram of agnes")
```

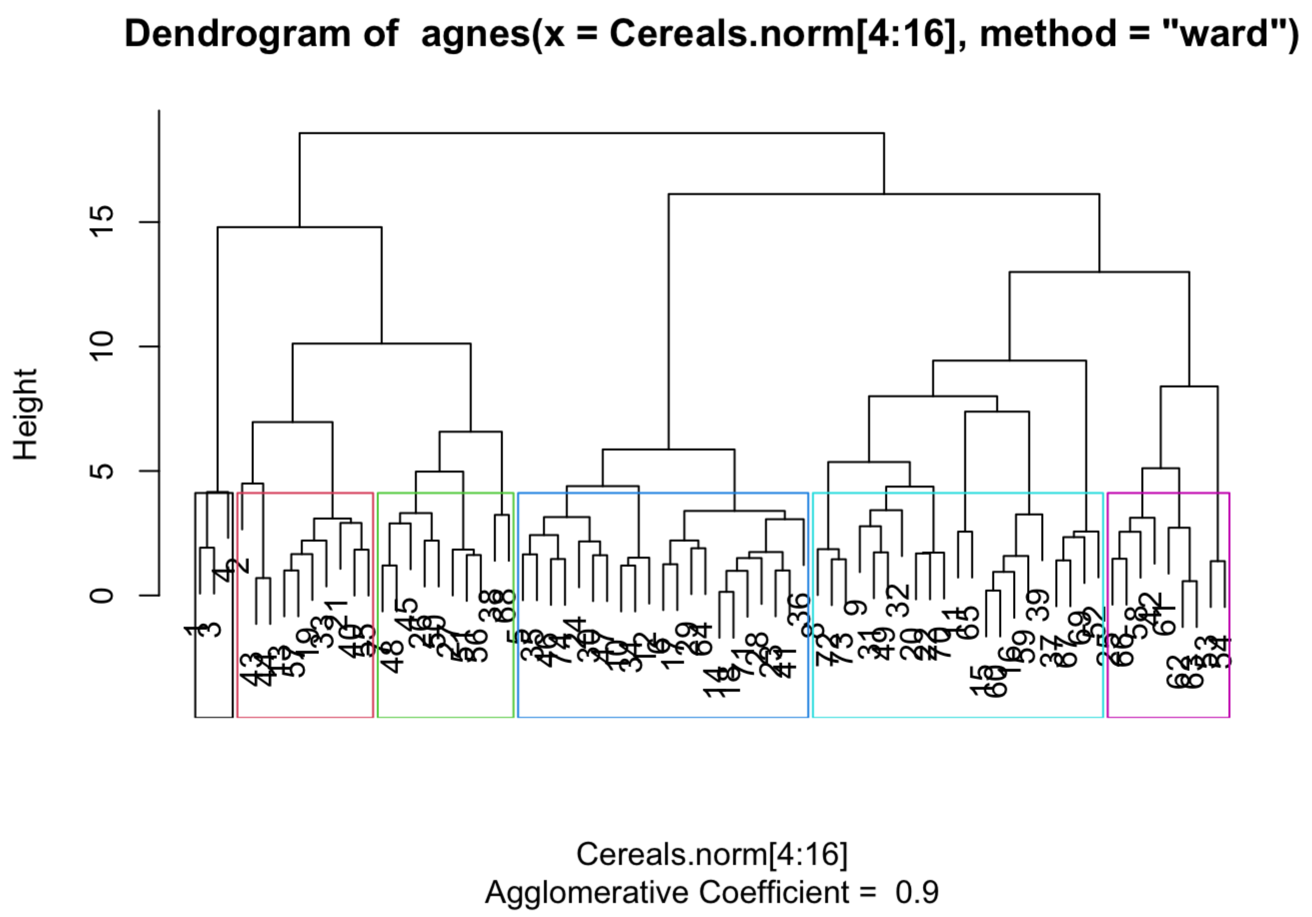


```
# We find that there would be 6-clusters

# Visualize the 6 clusters
plot(hc_ward)
```



```
rect.hclust(hc_ward, k = 6, border = 1:6)
```



Single linkage agglomerative coefficient: 0.607 Complete linkage agglomerative coefficient: 0.835 Average linkage agglomerative coefficient: 0.777 Ward agglomerative coefficient: 0.904

Since the ward agglomerative coefficient is the highest, this is the best linkage method

Hierarchical clustering produces consistent results, number of clusters picked from the dendrogram, and it takes hierarchical clustering longer to run

K-means is faster to run since computation time is linear, k-means starts with random number of clusters, and what number of clusters you want the data to be divided into (i.e. you set K)

Based on the dendrogram, there would be 6 clusters

```
# Look at stability of the clusters
# 6 centers from what we found in the dendrogram
# Use nstart = 25 for 25 initial centroids
model <- kmeans(Cereals.norm[4:16], centers = 6, nstart = 25)
100 * model$betweenss / model$totss # Calculate percentage of those that stay in their c
luster
```

```
## [1] 58.62927
```

```
# We get 0.5863, or 58.63%

# We now want to look at healthy cereal groups
# We don't want to standardize because of the importance of the measurements for health
 priorities
# For instance, average (i.e. z = 0) sugar might be unhealthy because it is in reference
to other sugars
# Use the unstandardized dataset
# Use k-means

# We want relevant health variables so only columns 4:12
cl <- kmeans(Cereals[4:12], centers = 6, nstart = 25)
Cereals <- data.frame(Cereals, cl$cluster)

# View the clusters
cl$centers
```

```
##   calories  protein       fat   sodium     fiber    carbo    sugars    potass
## 1 100.0000 3.333333 0.7777778 193.3333 7.0000000 11.00000  8.666667 248.88889
## 2 111.1538 2.192308 1.0000000 197.6923 1.4038462 15.75000  7.269231  70.96154
## 3 108.5714 1.571429 0.5714286 104.2857 0.5714286 11.57143 13.285714  30.00000
## 4  86.0000 2.500000 0.6000000   3.0000 2.1000000 14.60000  2.900000  95.00000
## 5 108.0000 2.400000 0.6000000 275.0000 0.5500000 19.35000  3.900000  51.00000
## 6 119.1667 3.250000 2.0833333 135.4167 2.5833333 13.41667  8.166667 127.91667
##   vitamins
## 1 33.33333
## 2 36.53846
## 3 25.00000
## 4 10.00000
## 5 32.50000
## 6 25.00000
```

We find that, using 6 clusters (what we found above) with 25 initial centroids that 58.63% of the cereals stay within the intial cluster of cereals.

The data shouldn't be standardized because units matter here. When it comes to health, it is important to know the intake of certain food choices - such as fiber and potassium - more than just the z-score. An average amount of sugar, for instance, can still be unhealthy given the comparison of other cereals. As such, knowing what the unit is matters from a health perspective. We find that cluster 1 is likely the most healthy, since it is low calorie, high in protien, high in fiver, less carboydrates, and high in potassium and higher in vitamins

The advantages of hierarchical clustering is the fact that there is no domain knowledge needed to set the clusters, you can just use the dendrogram to deduce the number. Hierarchical clustering also is easy to implement and has consistent results. This allows replication to be done, an important step for confirming research. The dendrogram is easy to understand the breakdown of the different subgroups within the main category (for example healhy cereal for the cereal category)