Assignment II - Cluster Analysis

Joseph Shu

3/13/2020

The data set food.xls gives the content of five different nutrients (calories, protein, fat, calcium and iron) in 27 food items.

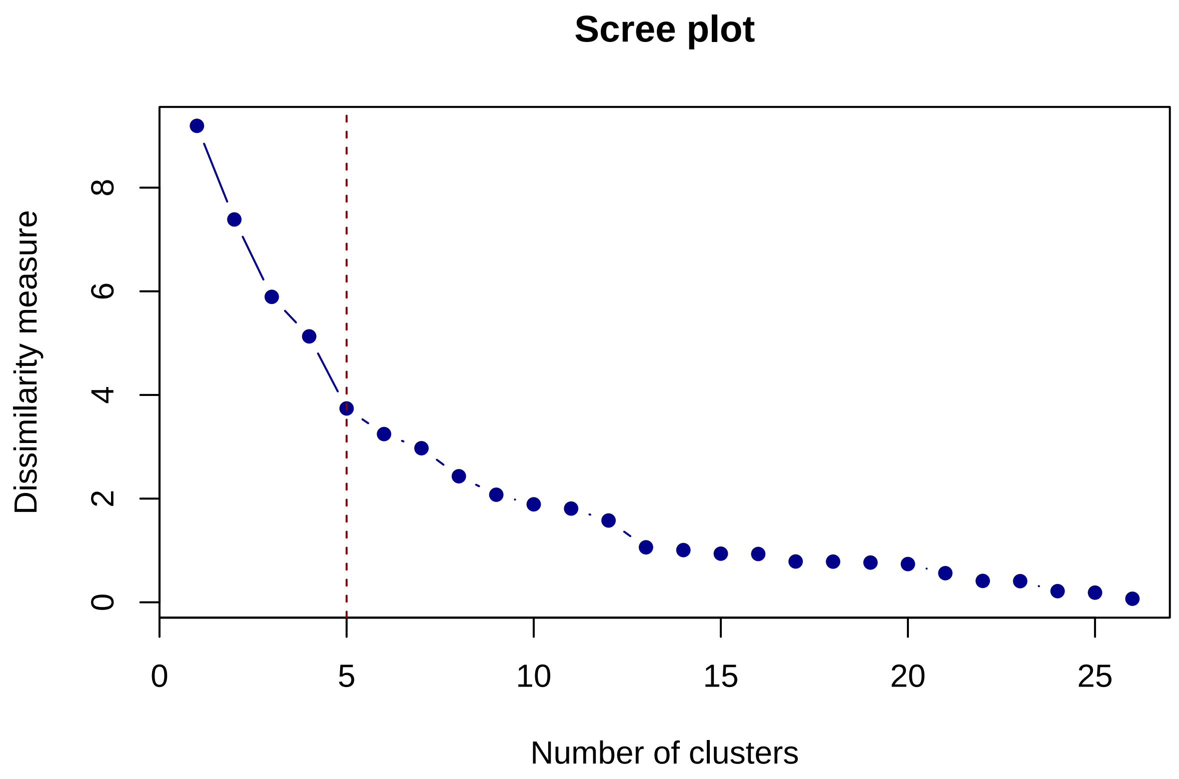
1. Perform an analysis of the data in order to group the food items into a few interpretable clusters. Assess the robustness of the cluster solution, and discuss your approach for it.
2. How many clusters is it optimal to retain? What do the clusters represent? Interpret their meaning.
3. Describe your general findings from the cluster analysis and interpret the software output.
4. Do you have any other thoughts about further analysis of this data? Since there's no single clustering method is best, I would start with some method of hierachical clustering to determine the probable number of clusters and their centroids.

Using the ward method

```r
library(readxl)
food1 <- read_xlsx("food.xlsx")
food1[,2:6] <- scale(food1[,2:6])
food1_ward <- hclust(dist(food1[, 2:6]), method="ward.D2")
```

As we don't want the clustering algorithm to depend on variable units, we standardize the data and choose the Ward mothod for aggregation.

```r
plot(rev(food1_ward$height), # rev is used to plot from low to high values on Y axis
     type = "b",           # to display both the points and lines
     ylab = "Dissimilarity measure",
     xlab = "Number of clusters",
     main = "Scree plot",
     col = "darkblue",
     pch = 16)             # specify the plot symbol: 16 = filled circle
abline(v = 5, lty = 2, col = "darkred") # draw a vertical line at v = 3
```



Based on the scree plot, similar as factor analysis, an elbow seems to appear between the third and forth clusters. We can create a dendrogram to see a clearer distance between the observations.

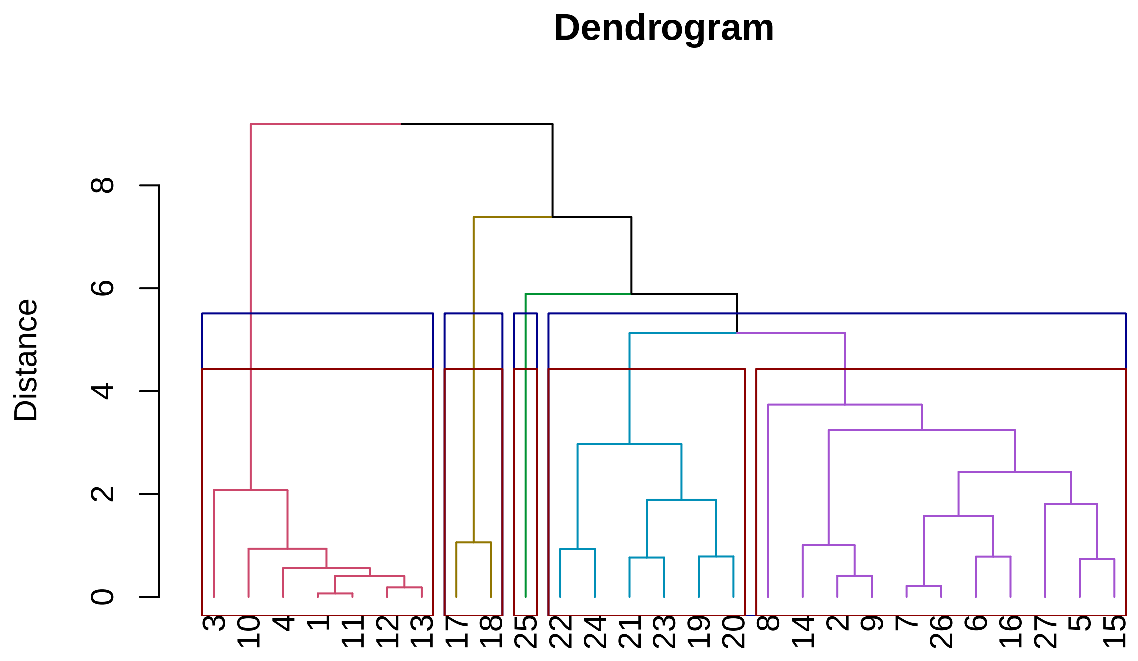
```
library(dendextend)
```

```
##
## ---------------------
## Welcome to dendextend version 1.13.4
## 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
## Or contact: <tal.galili@gmail.com>
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------

##
## Attaching package: 'dendextend'

## The following object is masked from 'package:stats':
##
##     cutree
```

```r
plot(set(as.dendrogram(food1_ward),
         "branches_k_color", # to highlight the cluster solution with a color
         k = 5),
     ylab = "Distance",
     main = "Dendrogram",
     cex = 0.2)              # Size of labels
rect.hclust(food1_ward, k = 4, border = "darkblue")  # draw red borders around 4 clusters
rect.hclust(food1_ward, k = 5, border = "darkred")  # draw red borders around 5 clusters
```



In the dendrogram, we can see that with either four or five clusters, item #25 would be categorized in the its own cluster. Item #17 and #18 seem to be far from the others and are allocated to their own cluster. We have to think about to whether proceed with three clusters, for parsimony's sake, or five clusters that might have better solution but hard to interpret. I'd proceed with 5 clusters because I want to see a more complex solution.

```r
memb <- cutree(food1_ward, k = 5)
table(memb)
```

```
## memb
##  1  2  3  4  5
##  7 11  2  6  1
```

We can find the size of each cluster.

K-means cluster analysis

```r
cent <- NULL
for(k in 1:5){
  cent <- rbind(cent, colMeans(food1[,2:6][memb == k, , drop = FALSE]))
}
round(cent, 3)
```

```
##      Calories Protein    Fat Calcium   Iron
## [1,]    1.437  -0.101  1.480  -0.452  0.022
## [2,]   -0.307   0.770 -0.414  -0.313  0.075
## [3,]   -1.481  -2.352 -1.109   0.436  2.271
## [4,]   -0.575  -0.627 -0.531   0.265 -0.934
## [5,]   -0.271   0.706 -0.398   4.140  0.081
```

Before we start with K-means cluster analysis, we can use a partioning approach, by defining the initial cluster centroids to be submitted to the K-means anlysis. The output indicates the average content levels for each of the five nutrients in each cluster.

```r
set.seed(1)
food_kmeans <- kmeans(food1[, 2:6], centers = cent, iter.max = 10)
food_kmeans
```

```
## K-means clustering with 5 clusters of sizes 8, 8, 2, 8, 1
##
## Cluster means:
##     Calories     Protein        Fat    Calcium        Iron
## 1  1.3286287 -0.05880006  1.3674579 -0.4512501  0.03833458
## 2 -0.2893295  0.91140090 -0.4203134 -0.2862584  0.32061646
## 3 -1.4811842 -2.35200232 -1.1087718  0.4361807  2.27092763
## 4 -0.6351527 -0.35280035 -0.6201884  0.1110030 -0.93682101
## 5 -0.2708033  0.70560069 -0.3981050  4.1396825  0.08110456
##
## Clustering vector:
##  [1] 1 2 1 1 2 4 2 2 1 1 1 1 1 2 2 4 3 3 4 4 4 4 4 4 5 2 2
##
## Within cluster sum of squares by cluster:
## [1]  4.3364978 13.0434862  0.5626097 10.2031176  0.0000000
##  (between_SS / total_SS =  78.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Now we can run the k-means clustering method by using the results from the preceding hierachical clustering as inputs into K-means. Based on the printed output, it shows us an overview of the size of the clusters and vector(specify which items are allocated to which cluster) and the cluster centers.(the average of all points that belong to that cluster)

```r
str(food_kmeans)
```

```
## List of 9
##  $ cluster     : int [1:27] 1 2 1 1 2 4 2 2 1 1 ...
##  $ centers     : num [1:5, 1:5] 1.329 -0.289 -1.481 -0.635 -0.271 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:5] "1" "2" "3" "4" ...
##   .. ..$ : chr [1:5] "Calories" "Protein" "Fat" "Calcium" ...
##  $ totss       : num 130
##  $ withinss    : num [1:5] 4.336 13.043 0.563 10.203 0
##  $ tot.withinss: num 28.1
##  $ betweenss   : num 102
##  $ size        : int [1:5] 8 8 2 8 1
##  $ iter        : int 2
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

```r
food_kmeans$size
```

```
## [1] 8 8 2 8 1
```

Extract the size of each of the clusters for a better look. It appears that the segments are relatively balanced in size, with 3 clusters contained of 8 items. The other two clusters should be those items (#25 and #17,18) as mentioned above, that are unique to the rest. We can later see the distance of each cluster and within cluster items more clearly with visualization.

```r
food_kmeans$centers
```

```
##     Calories     Protein        Fat    Calcium        Iron
## 1  1.3286287 -0.05880006  1.3674579 -0.4512501  0.03833458
## 2 -0.2893295  0.91140090 -0.4203134 -0.2862584  0.32061646
## 3 -1.4811842 -2.35200232 -1.1087718  0.4361807  2.27092763
## 4 -0.6351527 -0.35280035 -0.6201884  0.1110030 -0.93682101
## 5 -0.2708033  0.70560069 -0.3981050  4.1396825  0.08110456
```

Extract the cluster centers from the k-means analysis for better overview.

```r
change <- NULL
for (i in 1:5){
  change <- rbind(change, food_kmeans$centers[i,]-cent[i,])
}
round(change, 3)
```

```
##      Calories Protein    Fat Calcium   Iron
## [1,]   -0.109   0.042 -0.113   0.000  0.016
## [2,]    0.017   0.142 -0.006   0.027  0.246
## [3,]    0.000   0.000  0.000   0.000  0.000
## [4,]   -0.060   0.274 -0.089  -0.154 -0.003
## [5,]    0.000   0.000  0.000   0.000  0.000
```

We can calculate the differences of cluster centers between K-means and previous hierachical methond. I have run two methods using both three and five clusters, here shows only five for simplicity. Notably, with five clusters, the changes in cluster centers are smaller than with three clusters, as expected. In addition, in the 3rd and 5th clusters, there are no changes between two methods. Again, those two clusters contain the same items (#25, #17,18) that are so unique(far from others), that they won't change (allocated to other clusters) because of different methods.

```
dist(food_kmeans$centers)
```

```
##          1        2        3        4
## 2 2.619563                           
## 3 5.005751 4.107301                  
## 4 3.026660 1.869950 3.917484         
## 5 5.228594 4.437286 5.461708 4.309143
```

This table of pairwise distances indicates that, cluster 2 and 4 are the closest, while cluster 3 and 5 are furthest away from each other.(3 and 5 again,not surpised!)

```
food1 <- cbind(food1, cluster = food_kmeans$cluster)
```

We can store the cluster solution in our dataset for a clear view of each observation being assigned to which cluster.

```
# Overall average in the sample
round(colMeans(food1[, 2:6]), 3)
```

```
## Calories  Protein      Fat  Calcium     Iron 
##        0        0        0        0        0
```

The means value of each variable are 0 as expected due to standardization.

```
# Average for each cluster with one step
aggregate(food1[, 2:6],
          by = list(cluster = food1$cluster),
          FUN = mean)
```

```
##   cluster   Calories     Protein        Fat    Calcium        Iron
## 1       1  1.3286287 -0.05880006  1.3674579 -0.4512501  0.03833458
## 2       2 -0.2893295  0.91140090 -0.4203134 -0.2862584  0.32061646
## 3       3 -1.4811842 -2.35200232 -1.1087718  0.4361807  2.27092763
## 4       4 -0.6351527 -0.35280035 -0.6201884  0.1110030 -0.93682101
## 5       5 -0.2708033  0.70560069 -0.3981050  4.1396825  0.08110456
```

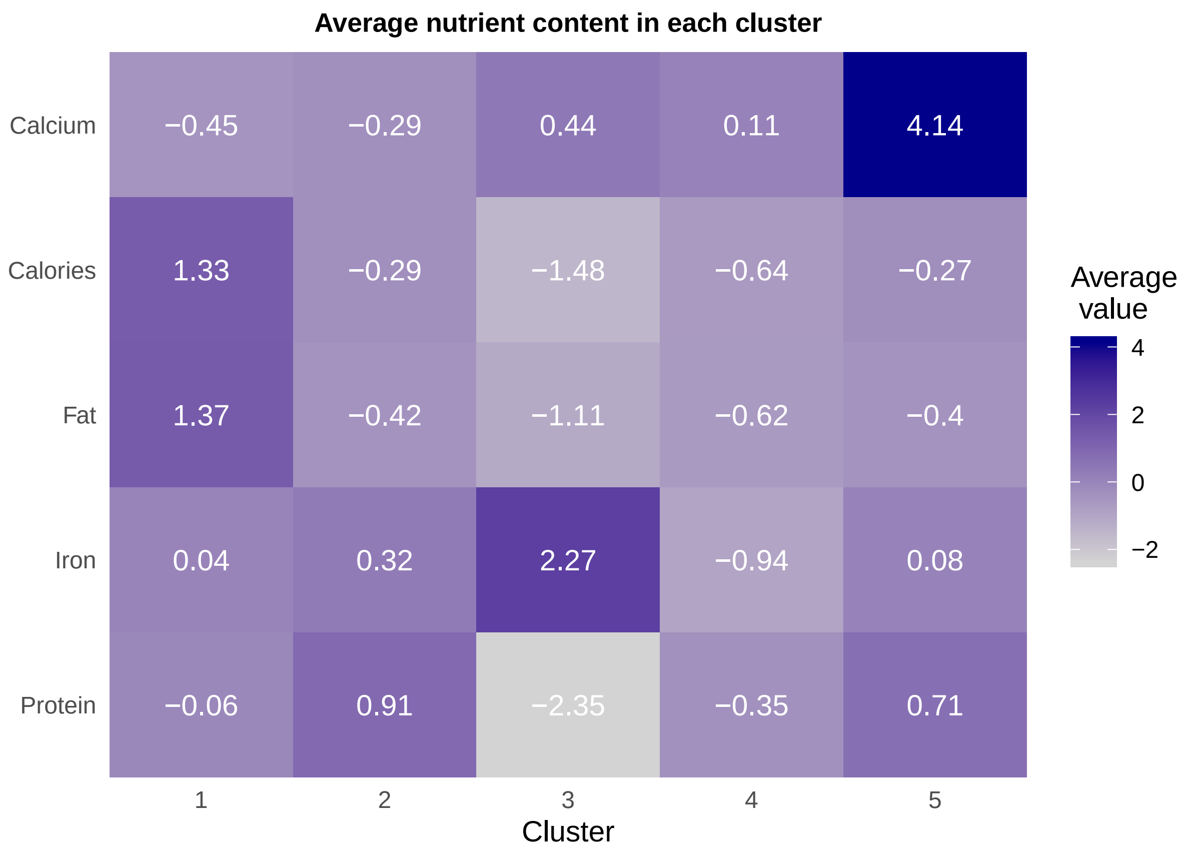
The average values in each cluster can be interpreted as the deviation from the average pattern in the data, we can create a heat map for a better intepretation.

```
library(tidyverse)
```

```
## -- Attaching packages ----------------------------------------------------- tidyverse 1.3.0 --
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
dt.cluster = aggregate(food1[, 2:6],
                       by = list(cluster = food1$cluster),
                       FUN = mean)
dt.cluster %>%
  gather(carmake, value, -cluster) %>% # to transfrom from wide to long format
  mutate(carmake = fct_rev(factor(carmake))) %>% # to reverse the order of car makes' names on the plot
  ggplot(aes(x = factor(cluster), y = carmake)) +
  geom_tile(aes(fill = round(value, digits = 2))) +
  geom_text(aes(label = round(value, digits = 2)), color="white") +
  scale_x_discrete(expand = c(0,0)) +
  scale_y_discrete(expand = c(0,0)) +
  scale_fill_gradient("Average\n value", low = "lightgrey", high = "darkblue") +
  theme_minimal() +
  labs(title = "Average nutrient content in each cluster",
       x = "Cluster",
       y = " ") +
  theme(legend.position="right",
        plot.title = element_text(size = 10, face = "bold", hjust = 0.5),
        axis.ticks = element_blank())
```

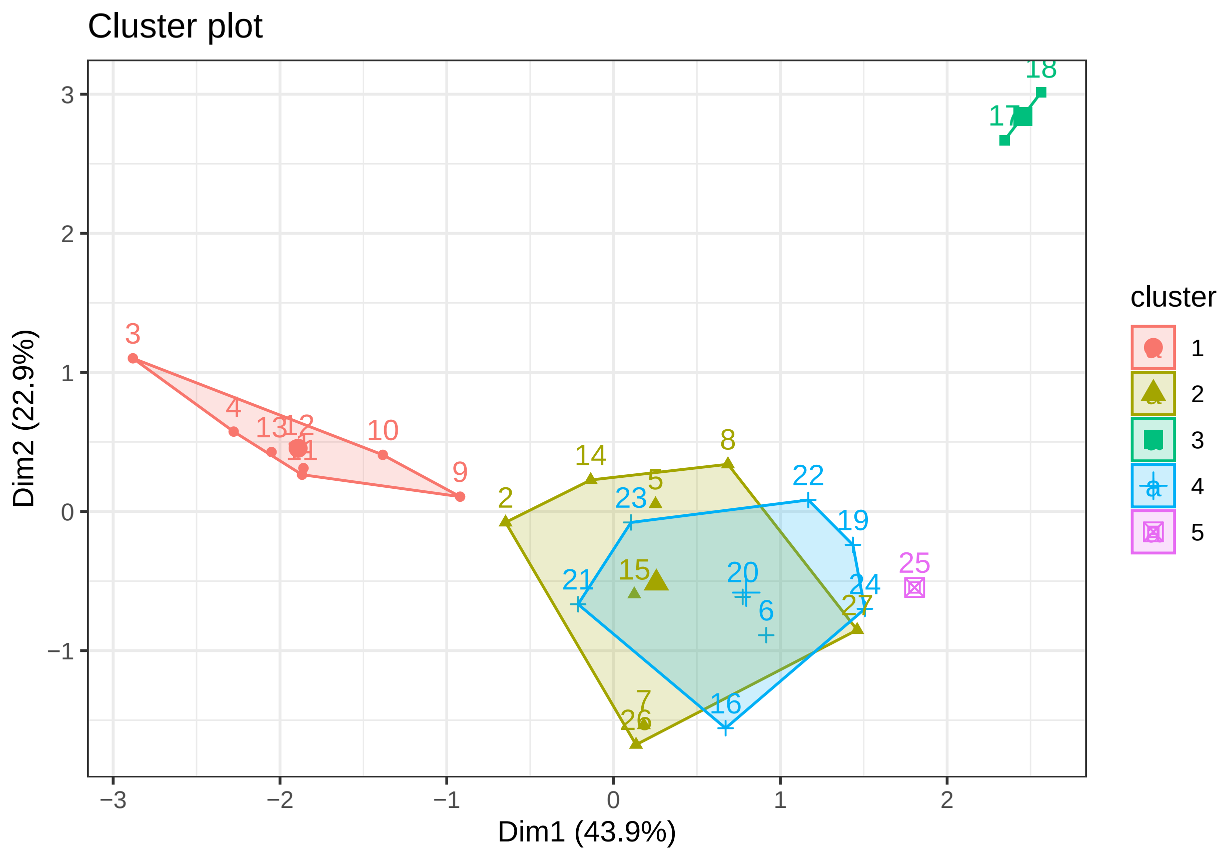


From the heat map we can observe that cluster 1 contains of relatively higher Fat and Calories items. We can then name it as "High Energy Foods". Cluster 2 consists of relatively high protein items than the rest of the clusters. We name it as "High Protein Foods". Cluster 3 shows an comparably extrem values of high iron and low protein. As mentioned many times above, they are the "unique" items that are far away from everthing, which stand for raw clams and canned clams. We can name it as "High Iron Foods". Cluster 4 seems to have opposite values than most of the foods. We can name it as "Low Nutrient Foods". The last cluster contains only one "unique" item, the canned sardines, which has a extremely high value in calcium than the rest of the foods. We can name it as "High Calcium Foods".

```
library(factoextra)
```

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

```
fviz_cluster(food_kmeans, data = food1[,2:6]) +
  theme_bw()
```



At the end, we can visualize the result for our cluster solution. I have run it both with three and five clusters, here we show only one solution for simplicity. Unsurprisingly, cluster 2,4,5 (yellow, blue, pink correspondingly) were allocated to the same cluster when we chose only three clusters, due to their proximity. On the other hand, items #17 and #18 are far away from other items as depicted. It may be easier to interpret with three clusters, however, we will also lose the abilitiy to identify item #25 with its "uniqueness", which weakens the robostness of the solution.

As a last note of the analysis, it may be interesting to validate the results, when we have a larger dataset to split. In addtion, to assess the overlapping domains of cluster 2 and cluster 4 for further analysis, such as optimal clustering and hard clustering solutions.