DA5030.P3-2

Minxin Cheng

Problem 2

Step 1. Read in Data file

```
data <-read.csv("Wholesale customers data.csv",header = T)
```

Step 2. Get an overview of the dataset

```
summary(data)
```

```
##     Channel          Region          Fresh             Milk
##  Min.   :1.000   Min.   :1.000   Min.   :     3   Min.   :   55
##  1st Qu.:1.000   1st Qu.:2.000   1st Qu.:  3128   1st Qu.: 1533
##  Median :1.000   Median :3.000   Median :  8504   Median : 3627
##  Mean   :1.323   Mean   :2.543   Mean   : 12000   Mean   : 5796
##  3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.: 16934   3rd Qu.: 7190
##  Max.   :2.000   Max.   :3.000   Max.   :112151   Max.   :73498
##     Grocery          Frozen        Detergents_Paper    Delicassen
##  Min.   :    3   Min.   :   25.0   Min.   :    3.0   Min.   :    3.0
##  1st Qu.: 2153   1st Qu.:  742.2   1st Qu.:  256.8   1st Qu.:  408.2
##  Median : 4756   Median : 1526.0   Median :  816.5   Median :  965.5
##  Mean   : 7951   Mean   : 3071.9   Mean   : 2881.5   Mean   : 1524.9
##  3rd Qu.:10656   3rd Qu.: 3554.2   3rd Qu.: 3922.0   3rd Qu.: 1820.2
##  Max.   :92780   Max.   :60869.0   Max.   :40827.0   Max.   :47943.0
```

There is a big difference for the top customers in each category (e.g. Fresh goes from a min of 3 to a max of 112,151). Normalizing/scaling the data won't necessarily remove those outliers. Doing a log transformation might help. We could also remove those customers completely. From a business perspective, you don't really need a clustering algorithm to identify what your top customers are buying. You usually need clustering and segmentation for your middle 50%.

Therefore, we will try to remove the top 5 customers from each category. We'll use a custom function and create a new data set called data.rm.top

```
# create a function to remove the top 5 customers
top.n.custs <- function(data, cols, n = 5){
  # initialize a vector to hold customers being removed
  idx.to.remove <- integer(0)
  for(c in cols){
    # sort column in descending order, which returns the sorted index
    # instead the actual values sorted
    col.order <- order(data[ , c], decreasing = TRUE)
    # take the first n of the sorted column
    idx <- head(col.order, n)
    # combine and remove the row ids that need to be removed
    idx.to.remove <- union(idx.to.remove, idx)
  }
  return(idx.to.remove)
}
# perform the function
top.custs <- top.n.custs(data, cols = 3:8, n = 5)
# return the number of customers that were removed
length(top.custs)
```

```
## [1] 19
```

```
# check the removed customers
data[top.custs, ]
```

```
##     Channel Region  Fresh  Milk Grocery Frozen Detergents_Paper Delicassen
## 182       1      3 112151 29627   18148  16745             4948       8550
## 126       1      3  76237  3473    7102  16538              778        918
## 285       1      3  68951  4411   12609   8692              751       2406
## 40        1      3  56159   555     902  10002              212       2916
## 259       1      1  56083  4563    2124   6422              730       3321
## 87        2      3  22925 73498   32114    987            20070        903
## 48        2      3  44466 54259   55571   7782            24171       6465
## 86        2      3  16117 46197   92780   1026            40827       2944
## 184       1      3  36847 43950   20170  36534              239      47943
## 62        2      3  35942 38369   59598   3254            26701       2017
## 334       2      2   8565  4980   67298    131            38102       1215
## 66        2      3     85 20959   45828     36            24231       1423
## 326       1      2  32717 16784   13626  60869             1272       5609
## 94        1      3  11314  3090    2062  35009               71       2698
## 197       1      1  30624  7209    4897  18711              763       2876
## 104       1      3  56082  3504    8906  18028             1480       2498
## 24        2      3  26373 36423   22019   5154             4337      16523
## 72        1      3  18291  1266   21042   5373             4173      14472
## 88        1      3  43265  5025    8117   6312             1579      14351
```

```
# remove them from dataset
data.rm.top <- data[-c(top.custs), ]
```

Now, using data.rm.top, we can perform the cluster analysis. We'll still need to drop the Channel and Region variables. These are two ID fields and are not useful in clustering.

```
set.seed(76964057)
# remove first two columns and create 5 clusters
k <- kmeans(data.rm.top[ , -c(1, 2)], center = 5)
# check cluster centers
k$centers
```

```
##       Fresh      Milk   Grocery   Frozen Detergents_Paper Delicassen
## 1  5830.214 15295.048 23449.167 1936.452       10361.6429   1912.738
## 2 18649.606  3335.586  4497.848 3301.747        1046.5859   1450.566
## 3  5845.392  2337.319  2878.205 2766.596         660.2952    858.994
## 4  4238.892  7725.289 11011.747 1336.566        4733.3614   1400.530
## 5 35922.387  4851.806  5862.581 3730.677        1004.6129   1552.161
```

```
# count the number of data points in each cluster
table(k$cluster)
```

```
## 
##   1   2   3   4   5 
##  42  99 166  83  31
```

Now we can start interpreting the cluster results:

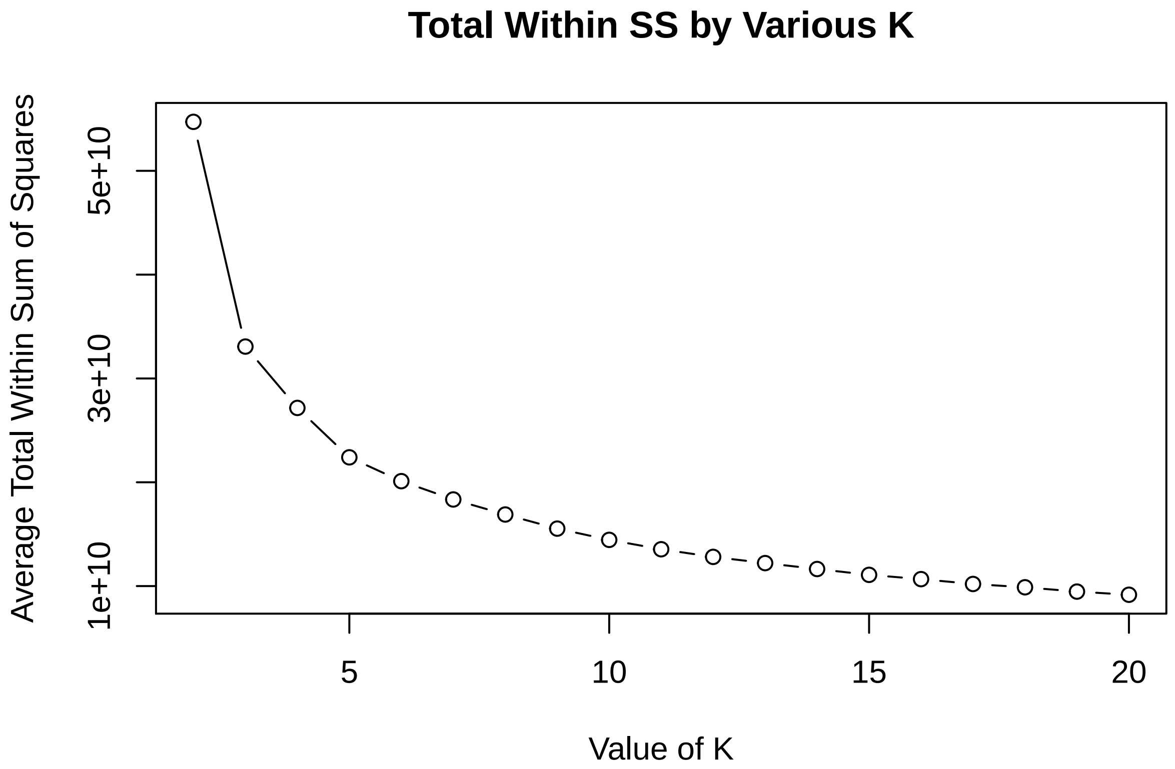
- Cluster 1 looks to be a heavy Grocery and above average Detergents_Paper but low Fresh foods.
- Cluster 3 is dominant in the Fresh category.
- Cluster 5 might be either the “junk drawer” catch-all cluster or it might represent the small customers. A measurement that is more relative would be the withinss and betweenss.

k*withinsswouldtellyouthesumofthesquareofthedistancefromeachdatapointtotheclustercenter.Lowerisbetter.Seeingahigh* tells you the sum of the squared distance between cluster centers. Ideally you want cluster centers far apart from each other. It’s important to try other values for K. You can then compare withinss and betweenss. This will help you select the best K. For example, with this data set, what if you ran K from 2 through 20 and plotted the total within sum of squares? You should find an “elbow” point. Wherever the graph bends and stops making gains in withinss you call that your K.

```
# try k from 2 to 20
rng <- 2:20
# run the kmeans algorithm 100 times
tries <- 100
# set up an empty vector to hold all of points
avg.totw.ss <- integer(length(rng))
for(v in rng){
  # set up an empty vector to hold the 100 tries
  v.totw.ss <- integer(tries)
  for(i in 1: tries){
    # run kmeans
    k.temp <- kmeans(data.rm.top, centers = v)
    # store the total withinss
    v.totw.ss[i] <- k.temp$tot.withinss
  }
  # average the 100 total withinss
  avg.totw.ss[v - 1] <- mean(v.totw.ss)
}
```

```
## Warning: did not converge in 10 iterations
```

```
# plot the figure to see the ideal number of clusters
plot(rng, avg.totw.ss, type = "b",
     main = "Total Within SS by Various K",
     ylab = "Average Total Within Sum of Squares",
     xlab = "Value of K")
```



This plot doesn't show a very strong elbow. Somewhere around K = 5 we start losing dramatic gains.