Clustering Lab Activity Sheet

Learning outcomes:

After solving these exercises, you should be able to understand the following:

1. Read and preprocess the data.
2. Cluster the dataset using hclust and k-means algorithms
3. Choose the parameter k
4. Check the cluster stability

Dataset schema:

Number of cases: 77
Variable Names:

1. **Name:** Name of cereal
2. **mfr**: Manufacturer of cereal where A = American Home Food Products; G = General Mills; K = Kelloggs; N = Nabisco; P = Post; Q = Quaker Oats; R = Ralston Purina
3. **type**: cold or hot
4. **calories**: calories per serving
5. **protein**: grams of protein
6. **fat**: grams of fat
7. **sodium**: milligrams of sodium
8. **fiber**: grams of dietary fiber
9. **carbo**: grams of complex carbohydrates
10. **sugars**: grams of sugars
11. **potass**: milligrams of potassium
12. **vitamins**: vitamins and minerals - 0, 25, or 100, indicating the typical percentage of FDA recommended
13. **shelf**: display shelf (1, 2, or 3, counting from the floor)
14. **weight**: weight in ounces of one serving
15. **cups**: number of cups in one serving
16. **rating**: a rating of the cereals

Steps:

1. Load the following libraries and read "cereal_with_cat.csv" file into R data frame.

```
# Load the required libraries
library(dummies)

# Reading the Cereals data
cereals_Data = read.csv('cereal_with_cat.csv',header = T)
```

2. Understand the *structure* and summary of the data using *str* and summary R commands

```
str(cereals_Data)
summary(cereals_Data)
```

3. Remove the following unused attributes from the data frame.

```
#Drop the attributes which are not required
rownames(cereals_Data) = cereals_Data$name
cereals_Data$name = NULL
```

4. Using domain knowledge separate categorical and numeric attributes. Convert them into appropriate type.

- To numeric using as.numeric()
- To categorical using as.factor()

Hint: Try using "sapply" function.

```
#divide into categorical and numeric
attr = c("mfr","type","calories","protein","fat","sodium","fiber","carbo",
         "sugars","potass","vitamins","shelf","weight","cups","rating")
cat_Attr <- c("mfr", "type", "shelf")
num_Attr <- setdiff(attr, cat_Attr)
rm(attr)

#converting categorical variables to factors
cereals_Data$mfr = as.factor(cereals_Data$mfr)
cereals_Data$type = as.factor(cereals_Data$type)
cereals_Data$shelf = as.factor(cereals_Data$shelf)

#converting numeric variables to numeric
cereals_Data[,num_Attr] = data.frame(sapply(cereals_Data[,num_Attr], as.numeric))
```

5. Standardize the numeric data.

```
#standardizing the numeric data
data <- scale(cereals_Data[,num_Attr])
```

6. Create dummy variables and convert all categorical variables to numeric

```
#Convert all categorcial attributes to numeric
mfr <- dummy(cereals_Data$mfr)
type <- dummy(cereals_Data$type)
shelf <- dummy(cereals_Data$shelf)

data <- data.frame(cbind(data, mfr, type, shelf))

rm(mfr, type, shelf, cat_Attr, num_Attr)
```

7. Recombine all the attributes using cbind.

```
data <- data.frame(cbind(data, mfr, type, shelf))

rm(mfr, type, shelf, cat_Attr, num_Attr)

data1 = data  #For use in k-means clustering
```

8. Perform Ward.D2 Hierarchical Clustering.

```
# ---------Hierarchical Clustering-----------------#
# Calculate distance
d <- dist(data, method = "euclidean")

# Ward Hierarchical Clustering
fit <- hclust(d, method="ward.D2")

# display dendogram
plot(fit)

# draw dendogram with red borders around the 5 clusters
rect.hclust(fit, k=5, border="red")

# cut tree into 5 clusters
cluster_Num <- cutree(fit, k=5)
cluster_Num

data = data.frame(data, cluster_Num)

plot(data[c("fiber", "sugars")] , col = data$cluster_Num, pch = 16)
```

Note, the flexibility with respect to choosing the level of granularity and ease of using multiple distance measures. However, challenge is to find where to cut the tree and time it takes to construct.

9. Stability of the clusters

```
## Stability of the clusters
# We will randomly sample 65 datapoints and plot the clusters to visualise
par(mfrow = c(2, 2))

set.seed(123)
for (i in 1:4){
  # Randomly sample 65 data points
  sample_data = data[sample(1:nrow(data), 65),]
  d <- dist(sample_data, method = "euclidean")
  fit <- hclust(d, method="ward.D2")
  cluster_Num <- cutree(fit, k=5)
  #plot each sample to visualise the clusters
  plot(sample_data[c("fiber", "sugars")], col = cluster_Num, pch = 16)
}
#resetting to original
par(mfrow = c(1,1))
rm(sample_data)
```

10. Perform k-means clustering and understand the resultant components

```
# ---------k-means Clustering-----------------#
# Running kmeans
clus <- kmeans(data1, 5)
clus

clus$cluster

table(clus$cluster)
clus$size
clus$centers
clus$betweenss
clus$withinss
clus$tot.withinss

plot(data1[c("fiber", "sugars")], col = clus$cluster, pch = 16)
points(clus$centers[, c("fiber", "sugars")],
       col = 1:3, pch = 8, cex = 2) # plot cluster centers
rm(clus)
```

Note, the clusters work on numeric data, efficiently processes large datasets however the clustering is local optimum.

11. Identify the best k value for clustering

```
# Identifying right number of clusters
set.seed(2341)
tot.wss <- 0
for (i in 1:15) {
  tot.wss[i] <- kmeans(data,centers=i)$tot.withinss
}

plot(1:15, tot.wss,
     type="b",
     xlab="Number of Clusters",
     ylab="Total within groups sum of squares")
rm(i, tot.wss)

#Best k
clus <- kmeans(data1, 6)
plot(data1[c("fiber", "sugars")], col = clus$cluster, pch = 16)

# append cluster numbers
data1 <- data.frame(data1, "cluster_Num" = clus$cluster)
head(data1)

rm(clus)
```

12. Sample and test the stability of the clusters. Visualize the results:

```r
## Stability of the clusters
# We will randomly sample 65 datapoints and plot the kmeans clusters to visualise
par(mfrow = c(2, 2))

set.seed(123)
for (i in 1:4){
  # Randomly sample 65 data points
  sample_data = data1[sample(1:nrow(data1), 65),]
  clus <- kmeans(sample_data, 6)
  #plot each sample to visualise the clusters
  plot(sample_data[,c("fiber", "sugars")] , col = clus$cluster, pch = 16)
}
#resetting to original
par(mfrow = c(1,1))
```