Exercise 1 Trashaj Alberto

alberto.trashaj

September 2023

1 Point 1

In the first point the request was to run a K-Means algorithm with $k = 3$ and $k = 9$ with both scaled and unscaled data: the dataset taken into account is the Olive Oil data which can be found in the virtuale page of the course.

We import the dataset, see the structure and ask for a summary.

```
oliveoil <- read.csv("~path/oliveoil.dat", sep="") #substitute the path with the
str(oliveoil)
summary(oliveoil)
```

From the structure we can see that there are two variables that should be converted as *factor*:

```
data <- oliveoil #change the name just for comfort
data$macro.area <- factor(data$macro.area)
data$region <- factor(data$region)
str(data)
```

Now we are ready to perform the clustering

```
library(cluster)

set.seed(123)

results_unscaled <- list()
results_scaled <- list()
scaled <- data %>%
    select(-macro.area, -region)

df.unscaled <- scaled
df.scaled <- scale(scaled)

k_values <- c(3,9)

wss_unscaled <- numeric(length(k_values))
```

```
    wss_scaled <- numeric(length(k_values))

    for (k in k_values){
        k_means_unscaled <- kmeans(df.unscaled, centers = k)
        results_unscaled[[as.character(k)]] <- k_means_unscaled$cluster

        kmeans_scaled <- kmeans(df.scaled, centers = k)
        results_scaled[[as.character(k)]] <- kmeans_scaled$cluster

        cat("For K=", k, ":\n")


        cat("Unscaled Data WSS:", k_means_unscaled$tot.withinss, "\n")
        cat("Scaled Data WSS:", kmeans_scaled$tot.withinss, "\n")


         cat("Unscaled Data (Macro Areas):\n")
         print(table(results_unscaled[[as.character(k)]], data$macro.area))
         cat("Scaled Data (Macro Areas):\n")
         print(table(results_scaled[[as.character(k)]], data$macro.area))
         cat("Unscaled Data (Regions):\n")
         print(table(results_unscaled[[as.character(k)]], data$region))
         cat("Scaled Data (Regions):\n")
         print(table(results_scaled[[as.character(k)]], data$region))
}
```

From the result we can clearly see that the tot.withinss is much lower when we perform clustering with the scaled data: this means that the cluster are closer to the centroids, suggesting more compact clusters.

2 Point 2

Let $D = \boldsymbol{x_1}, ..., \boldsymbol{x_n}$ and $\boldsymbol{x_i} = (x_{i1}, ..., x_{ip})$ for $i \in \mathbf{N}$ the centroids can be denoted as $m_1^{Km}, ..., m_K^{\hat{K}m}$ and the cluster assignements can be denoted as $c^{Km}(1), ..., c^{Km}(n)$ for given K.

Let's multiply the numerical values of the observations contained in a dataset by a constant $q \neq 0$: we denote this new dataset as D^*. All the points contained in the original dataset D can be denoted then as $x_i^* = q * x_i$.

Recall that the objective function that the k-Means algorithm try to minimize is

$$S(\mathcal{C}, \boldsymbol{m_1}, ...\boldsymbol{m_K}) = \sum_{i=1}^{n} ||\boldsymbol{x_i} - \boldsymbol{m_{c(i)}}||^2$$

Now, if we multiply all the variables, the distances between points and the centroids are also scaled by q:

$$S^*(\mathcal{C}^*, \boldsymbol{m_1^*}, ...\boldsymbol{m_K^*}) = \sum_{i=1}^{n} q * ||\boldsymbol{x_i^*} - \boldsymbol{m_{c(i)}}||^2$$

Minimizing the first or the second function will yeld to the same result: in fact, although the shape of the function will change, the location of the minimum remains the same. Therefore we can say that the clusters are the same: important note, since we want to find the minimum the constant q needs to be positive definite ($q > 0$).

Regarding the new centroids, they will differ because they are computed as the mean of the points in the cluster: therefore, the mean will be also multiplied by q.

3 Point 3

In the third point is requested to perform a cluster analysis on the Boston dataset, after visualizing the data and check if there are discrete variables that can cause problems with the clustering procedure.
But first, let's import and analyze the structure of the dataset:

```
Boston <- read.csv("~path/Boston.dat", sep="")
str(Boston)
summary(Boston)
cor(Boston)
attach(Boston)
```

To check whether there are some discrete variables we will use this rule: if there are less than 30 different values per variable, we will say that the variable is discrete.

```
library(dplyr)
names <- colnames(Boston)
out <- list()
for (i in names){
    number <- n_distinct(Boston[i])
    if (number < 30)
     out <- append(out, i)
}
out
```

This for loop suggest us to get rid of the following variables "zn", "rad" and "chas".

```
reduced_df <- Boston %>%
    select(-zn, -chas, -rad)
reduced_df
```

Let's try to visualize the dataset:

```
pairs_boston <- pairs(Boston)
```

```
pairs_reduced_boston <- pairs(reduced_df)
heatmap_boston <- heatmap(as.matrix(Boston))
heatmap_reduced_boston <- heatmap(as.matrix(reduced_df))
```

Now we perform a k-Means clustering:

```
set.seed(1234)
k <- c(3,5,7,9,11,13,15)

reduced_df <- scale(reduced_df)

for (i in k){
    k_means <- kmeans(reduced_df, centers = i)
    print(k_means)
}
```

Moreover, we perform a hierarchical clustering and we compare the result with the k-Means output.

```
dist_matrix <- dist(reduced_df)

hc <- hclust(dist_matrix, method = "complete")

plot(hc)
```

Let's make a contingency table to see whether the two methods assigned the same observations to the same cluster

```
clusters_hc <- cutree(hc, k = 15)  # Hierarchical clustering
clusters_kmeans <- k_means$cluster  # K-means clustering

table(clusters_hc, clusters_kmeans)
```

From the output we can see that the elements in the diagonal are approximately all 0: the two methods gives different results.

4 Point 4

For this part we are going to use a dataset taken from https://www.kaggle.com/datasets/mariopasquato/star-cluster-simulations/: this dataset contains the positions and velocities of simulated stars (particles) in a direct N-body simulation of a star cluster. In the cluster there are initially 64000 stars distributed in position-velocity space according to a King model. The matrix produced from the dataset will be large.

The second dataset is the one provided in virtuale called "Bundestag".

In this algorithm, the main difference with the "original" Lloyd algorithm, is how to choose the the initial points/centers in the clustering process. In the 'kmpp' function, centers are more likely to be chosen from points that are closer to the existing centers; in the original Lloyd algorithm, instead, the points are taken uniformly from the dataset.

```
library(pracma)
kmpp <- function(X, k) {
 n <- nrow(X)
 C <- numeric(k)
 C[1] <- sample(1:n, 1)
for (i in 2:k) {
  dm <- distmat(X, X[C, ])
  pr <- apply(dm, 1, min);
  pr[C] <- 0
  C[i] <- sample(1:n, 1, prob = pr)
  }
kmeans(X, X[C, ])
}

c_1800 <- read.csv("~/Desktop/Universita /Unsupervised/Stars/c_1800.csv")
c_1800_matrix <- as.matrix(c_1800)

bundestag <- read.csv("~/Desktop/Universita /Unsupervised/bundestag.dat", s
bundestag.matrix <- as.matrix(bundestag)


k <- 5

kmpp_stars <- kmpp(c_1800_matrix, k)
kmeans_star_first <- kmeans(c_1800_matrix, k, nstart=1)
kmeans_star_hundred <- kmeans(c_1800_matrix, k, nstart = 100)

cat("kmpp_stars withinss:", kmpp_stars$tot.withinss, "\n")
cat("kmeans_star_first withinss:", kmeans_star_first$tot.withinss, "\n")
cat("kmeans_star_hundred withinss:", kmeans_star_hundred$tot.withinss, "\n")



bundestag_num <- bundestag %>%
 select(-state, -ewb)

bundestag.matrix <- as.matrix(bundestag_num)

bundestag_kmpp <- kmpp(bundestag.matrix, k)
bundestag_kmeans_first <- kmeans(bundestag.matrix, k , nstart=1)
bundestag_kmeans_hundred <- kmeans(bundestag.matrix, k , nstart=100)



cat("bundestag_kmpp:", bundestag_kmpp$tot.withinss, "\n")
cat("bundestag_kmeans_first:", bundestag_kmeans_first$tot.withinss, "\n")
```

```
cat("bundestag_kmeans_hundred:", bundestag_kmeans_hundred$tot.withinss, "\n"
```