STAT 4051

Ruwiada Al Harrasi

2023-09-22

```
library(cluster)
library(dplyr)
```

```
##
## 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
```

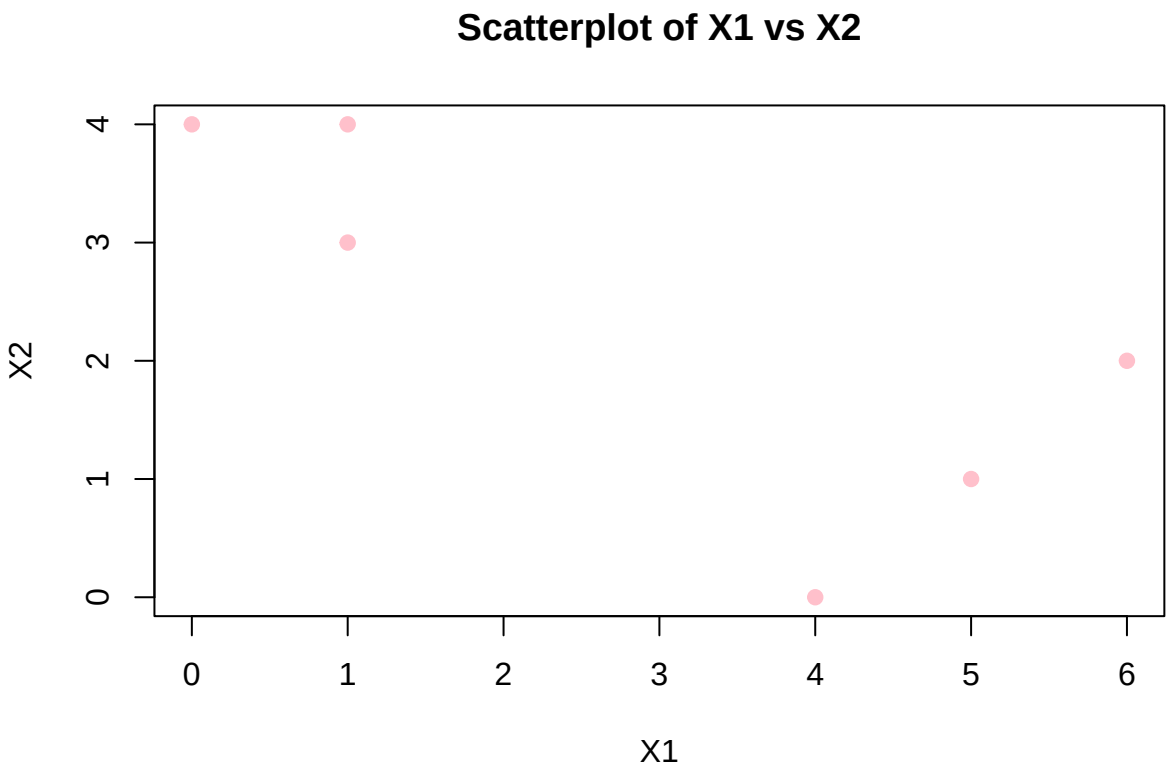
```
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

1) In this problem, you will perform K-means clustering manually, with K = 2, on a small example with n = 6 observations and p = 2 features. The observations are as follows. A. Plot the observations.

```
k=2
n=6
p=2
#Creating a dataframe
df= data.frame(
  X1=c(1,1,0,5,6,4),
  X2=c(4,3,4,1,2,0)
)
plot(df$X1, df$X2, main="Scatterplot of X1 vs X2", xlab = "X1", ylab= "X2", pch=19,col= "pink")
```



C. Compute the centroid for each cluster.

```
labels = c(1, 2, 2, 1, 1, 1)
df= cbind(df, labels)
centroid_1= colMeans(df[df[,3] == 1,, drop = FALSE])[1:2]
centroid_1
```

```
##   X1   X2
## 4.00 1.75
```

```
centroid_2 = colMeans(df[df[,3] == 2,, drop = FALSE])[1:2]
centroid_2
```

```
##  X1  X2
## 0.5 3.5
```

D.Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation.

```
# Function to calculate Euclidean distance
euclidean_distance <- function(x1, y1, x2, y2) {
  sqrt((x1 - x2)^2 + (y1 - y2)^2)
}

# Assign each observation to the closest centroid
```

```r
for (i in 1:nrow(df)) {
  x <- df[i, "X1"]
  y <- df[i, "X2"]

  # Calculate distances to both centroids
  distance_to_centroid_1 <- euclidean_distance(x, y, centroid_1[1], centroid_1[2])
  distance_to_centroid_2 <- euclidean_distance(x, y, centroid_2[1], centroid_2[2])

  # Assign the observation to the cluster with the closest centroid
  if (distance_to_centroid_1 < distance_to_centroid_2) {
    df[i, "Cluster"] <- 1
  } else {
    df[i, "Cluster"] <- 2
  }
}

df$Cluster
```

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

E. Repeat (c) and (d) until the answers obtained stop changing.

```r
labels = c(2, 2, 2, 1, 1, 1)
df[,3] = labels
newcentroid_1= colMeans(df[df[,3] == 1,, drop = FALSE])[1:2]
newcentroid_1
```

```
## X1 X2
##  5  1
```

```r
newcentroid_2 = colMeans(df[df[,3] == 2,, drop = FALSE])[1:2]
newcentroid_2
```

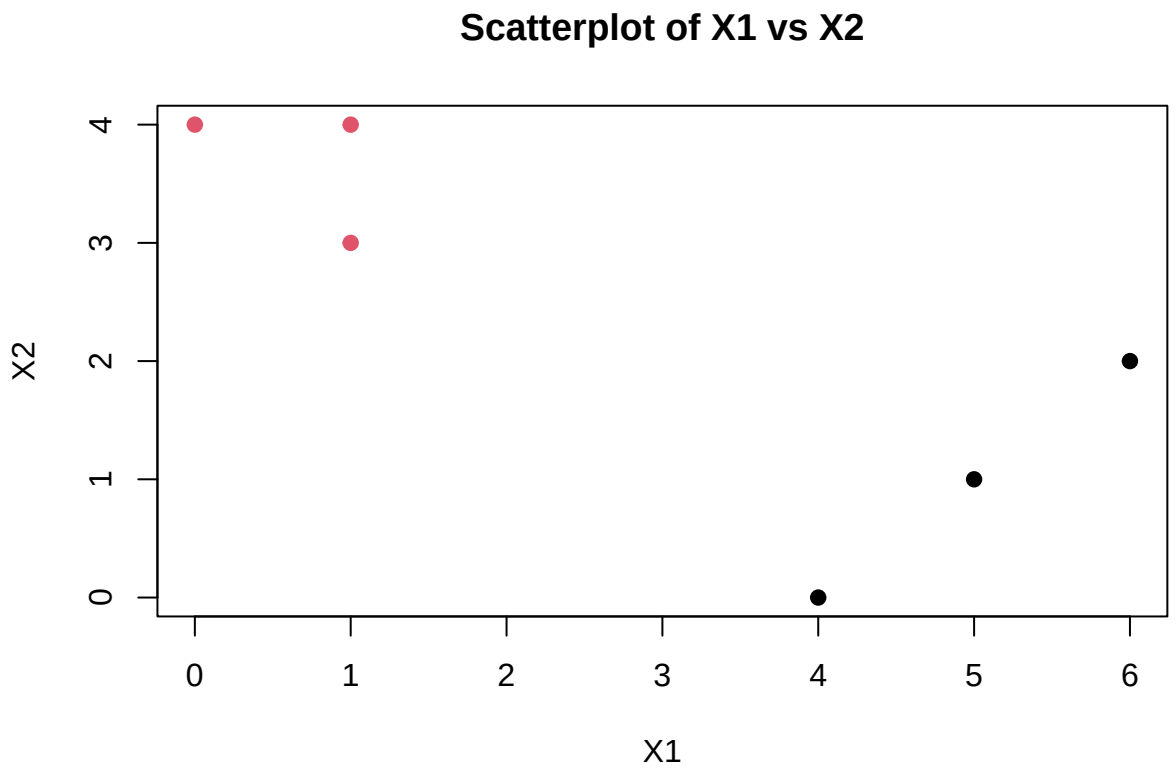
```
##        X1        X2
## 0.6666667 3.6666667
```

```r
# Assign each observation to the closest centroid
for (i in 1:nrow(df)) {
  x <- df[i, "X1"]
  y <- df[i, "X2"]

  # Calculate distances to both centroids
  newdistance_to_centroid_1 <- euclidean_distance(x, y, newcentroid_1[1], newcentroid_1[2])
  newdistance_to_centroid_2 <- euclidean_distance(x, y, newcentroid_2[1], newcentroid_2[2])

  # Assign the observation to the cluster with the closest centroid
  if (newdistance_to_centroid_1 < newdistance_to_centroid_2) {
    df[i, "Cluster"] <- 1
  } else {
    df[i, "Cluster"] <- 2
  }
}
print(df$Cluster)
```

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

F. In your plot from (a), color the observations according to the cluster labels obtained.

```
plot(df$X1, df$X2, main="Scatterplot of X1 vs X2", xlab = "X1", ylab= "X2",pch = 19, col=df$Cluster)
```

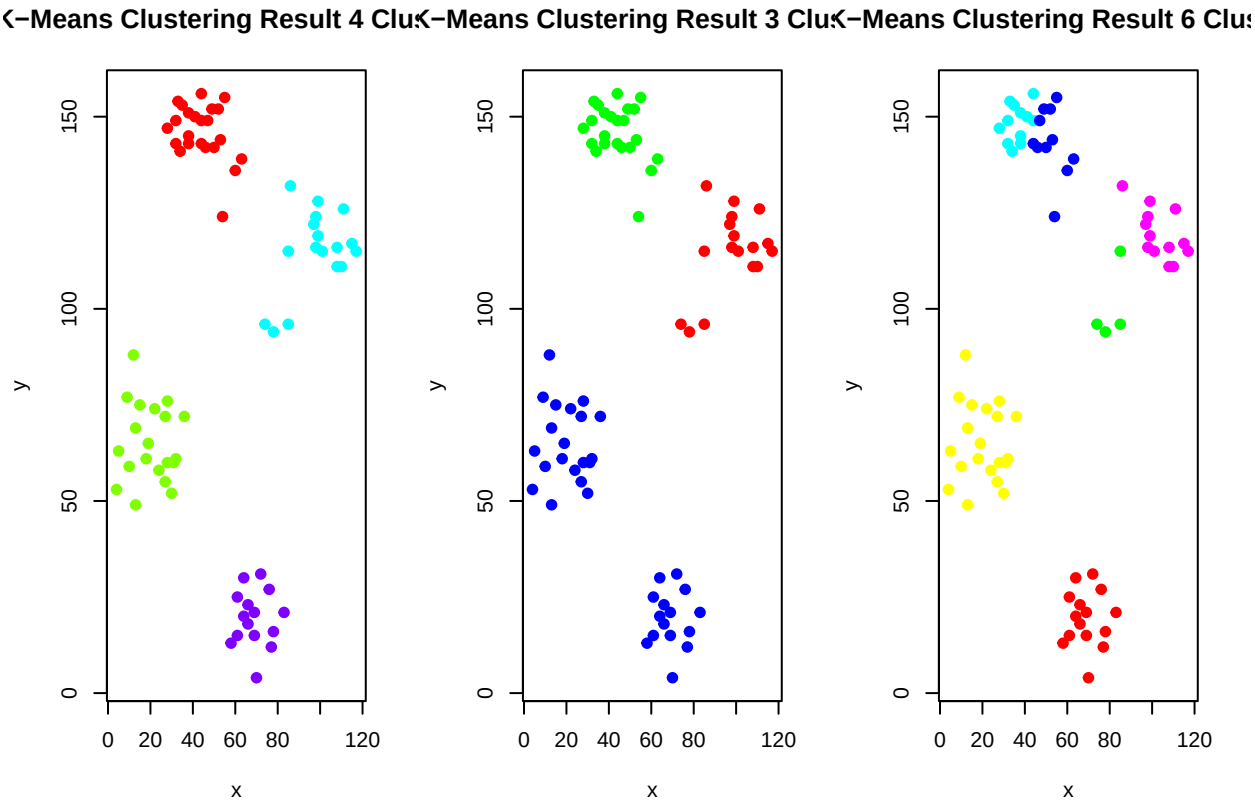


2. (10 pts) Use the data set ruspini from the R package cluster. Use random start number > 100 for all results.

(a) Run the Kmeans algorithm with 4 clusters, and produce the clustering result plot as in the slides.
(b) Run the Kmeans algorithm with 3 clusters, and produce the clustering result plot. (c) Run the Kmeans algorithm with 6 clusters, and produce the clustering result plot.

```
# Load the ruspini dataset
data1 <- ruspini
# Run k-means clustering for different numbers of clusters
kmeansa <- kmeans(data1, centers = 4, nstart = 100)
kmeansb <- kmeans(data1, centers = 3, nstart = 100)
kmeansc <- kmeans(data1, centers = 6, nstart = 100)

# Generate a sequence of rainbow colors
rainbow_colors_a <- rainbow(4)
rainbow_colors_b <- rainbow(3)
rainbow_colors_c <- rainbow(6)
```

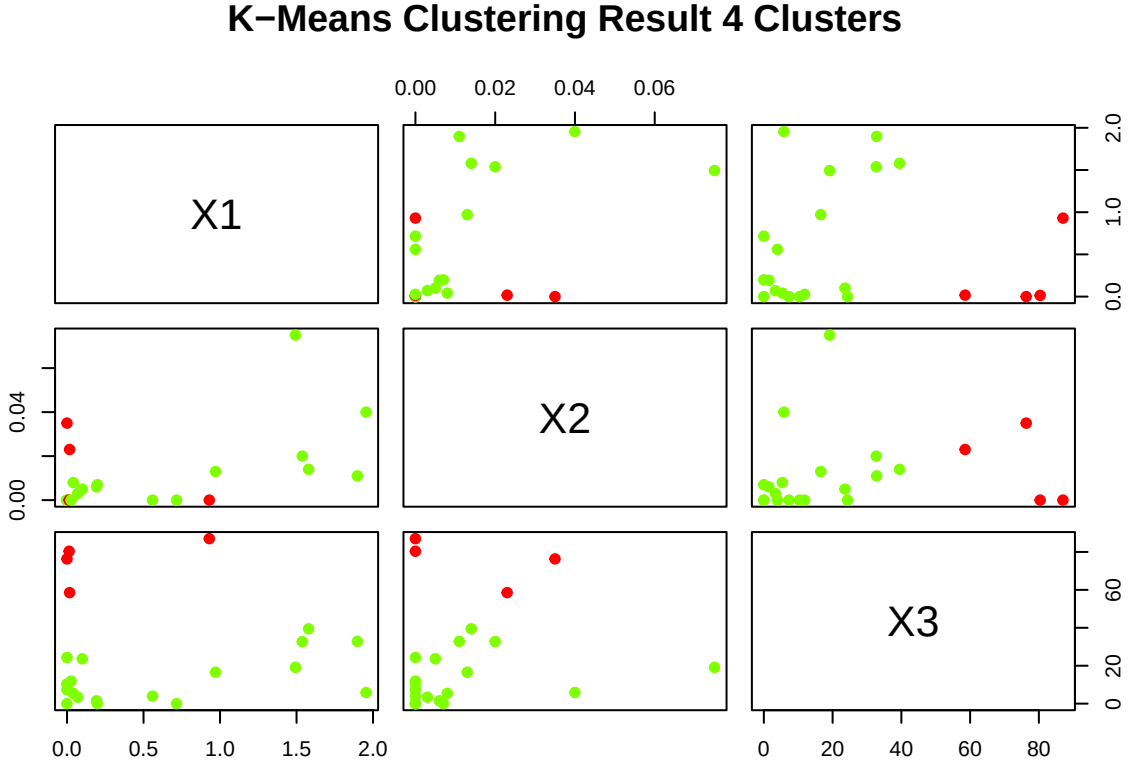
```
# Create plots for each clustering result with rainbow colors
par(mfrow=c(1, 3))  # Set up a 1x3 grid of plots
plot(data1, col = rainbow_colors_a[kmeansa$cluster], pch=19,
      main = "K-Means Clustering Result 4 Clusters")
plot(data1, col = rainbow_colors_b[kmeansb$cluster], pch=19,
      main = "K-Means Clustering Result 3 Clusters")
plot(data1, col = rainbow_colors_c[kmeansc$cluster], pch=19,
     main = "K-Means Clustering Result 6 Clusters")
```



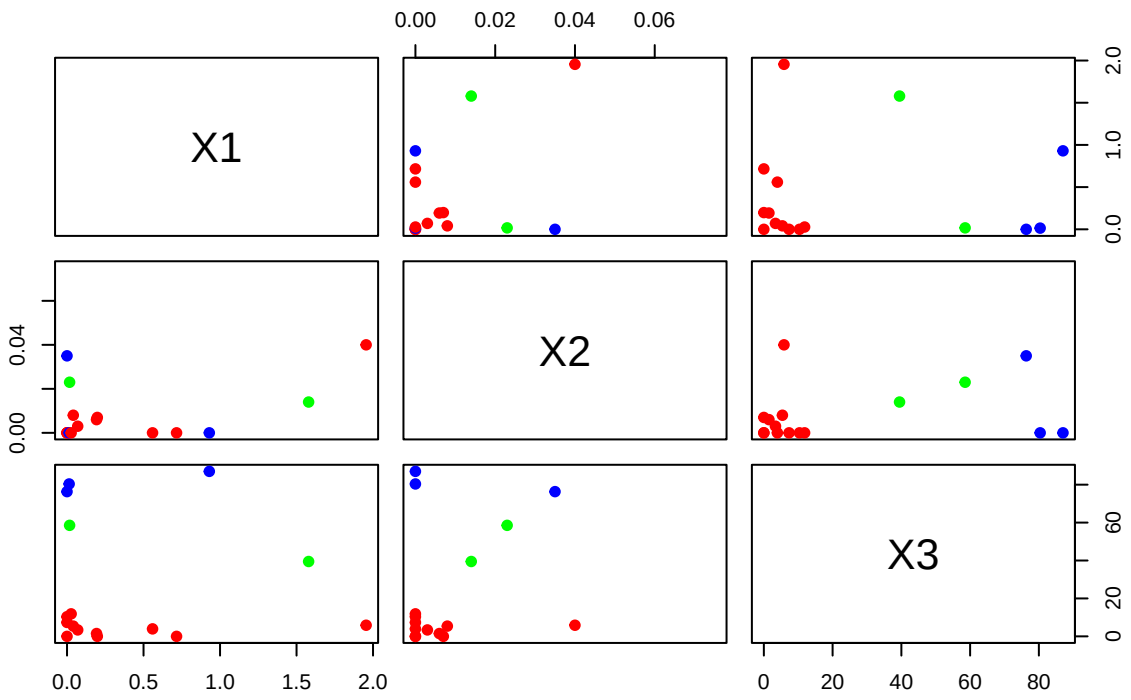
3. (10 pts) The CancerData-Small.csv is a csv data file available in the data folder of the Canvas site. The data set is about some basic medical measures on 22 patients with two types of cancers. This will be a small data set we use multiple times to experiment with different algorithms. The first three variables X1, X2, X3 are the medical measures, and the Cancer is the cancer type. Use the three medical measures as your data and cluster the patients into 2, 4, 6 clusters using the Kmeans algorithm. Report the cross-table of your clustering labels and the cancer types. Do any of your clustering results seem to find a matching pattern with the cancer type? (Hint: if you have two different label vectors y1 and y2 in R, you can directly use table(y1, y2) to see their cross-table counts.)

```
data= read.csv("~/Downloads/CancerData-Small.csv", header=T)
X1mean <- mean(data$X1, na.rm = TRUE)
data$X1[is.na(data$X1)]= X1mean
# Extract the three medical measures (X1, X2, X3)
X <- data[, c("X1", "X2", "X3")]
```

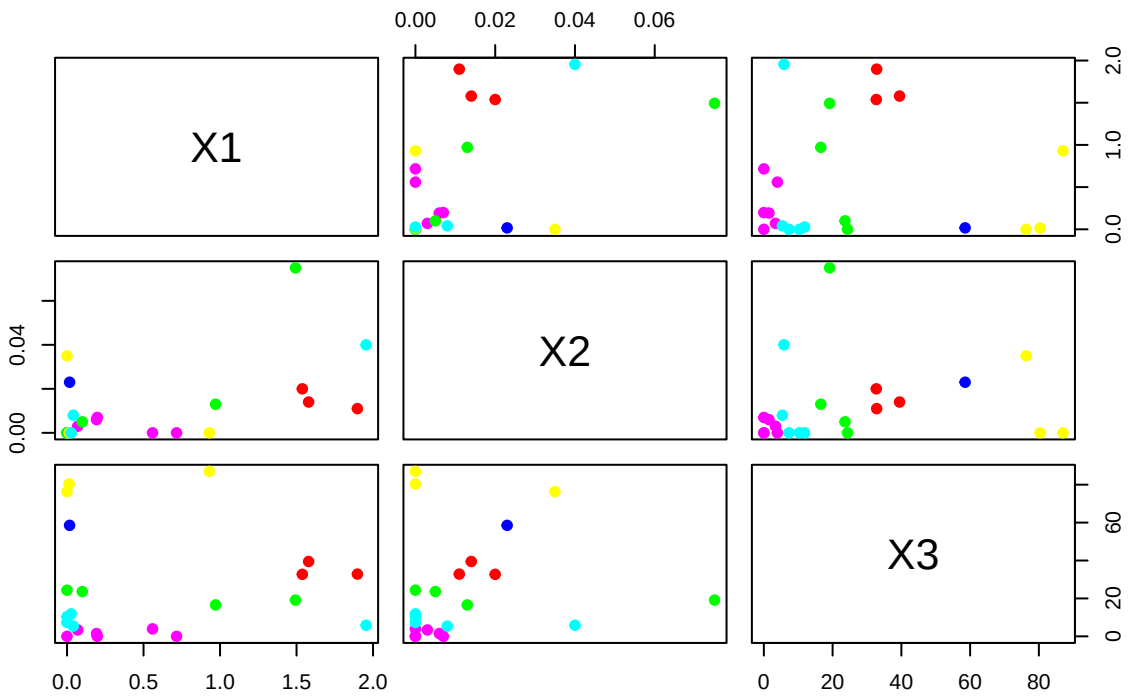
```r
# Perform K-means clustering for 2 clusters
kmeans2 <- kmeans(X, centers = 2)
# Perform K-means clustering for 4 clusters
kmeans4 <- kmeans(X, centers = 4)
# Perform K-means clustering for 6 clusters
kmeans6 <- kmeans(X, centers = 6)

plot( X, col = rainbow_colors_a[kmeans2$cluster], pch=19,
      main = "K-Means Clustering Result 4 Clusters")
```




```r
plot(X, col = rainbow_colors_b[kmeans4$cluster], pch=19,
      main = "K-Means Clustering Result 3 Clusters")
```




```
plot(X, col = rainbow_colors_c[kmeans6$cluster], pch=19,
     main = "K-Means Clustering Result 6 Clusters")
```




```
cancer_labels <- data$Cancer

# Create contingency tables for 2, 4, and 6 clusters
cross_table2 <- table(kmeans2$cluster, cancer_labels)
cross_table2
```

```
##    cancer_labels
##      1  2
##   1  3  1
##   2  7 11
```

```
cross_table4 <- table(kmeans4$cluster, cancer_labels)
cross_table4
```

```
##    cancer_labels
##      1  2
##   1  1 10
##   2  1  1
##   3  3  0
##   4  5  1
```

```
cross_table6 <- table(kmeans6$cluster, cancer_labels)
cross_table6
```

```
##    cancer_labels
##     1 2
##   1 3 0
##   2 3 0
##   3 3 1
##   4 0 5
##   5 0 1
##   6 1 5
```

The 4-cluster K-means result demonstrates better separation, with Clusters 1 and 3 primarily corresponding to Cancer 1 and Clusters 2 and 4 mainly corresponding to Cancer 2, suggesting a partial matching pattern, while the 2-cluster K-means result lacks perfect alignment with the cancer types, and the 6-cluster K-means result shows a more complex, less clear-cut relationship with the cancer types.

*worked with Quyen and Katie