MATH 4323, Homework # 5.

Name: Hyundoo Jeong
PSID:2212332

Instructions: Submit the solutions as a file (type it up and save as a *.pdf* or a *Word*-file) via UH Blackboard. Keep responses brief and to the point. For code & output: include only pieces that are of utmost relevance to the question.

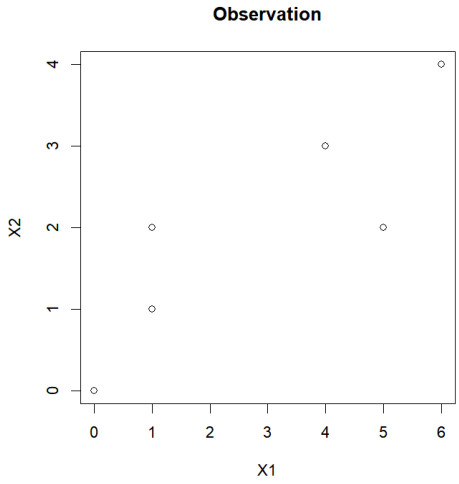
Conceptual.

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.

Obs.	X_1	X_2
1	6	4
2	5	2
3	4	3
4	1	2
5	1	1
6	0	0

(a) Plot the observations.

Answer:



(b) Randomly assign a cluster label to each observation as follows:

```
RNGkind(sample.kind = "default")
set.seed(2)
```

labels <- sample(2, nrow(x), replace = T)

Report the cluster labels for each observation.

Answer:
```
> labels <- sample(2, nrow(x), replace = T)
> labels
[1] 1 2 1 1 2 1
```

(c) Compute the centroids for each cluster.

Answer:
```
> centroid1 <- c(mean(x[labels == 1, 1]), mean(x[labels == 1, 2]))
> centroid2 <- c(mean(x[labels == 2, 1]), mean(x[labels == 2, 2]))
> print(centroid1)
[1] 2.75 2.25
> print(centroid2)
[1] 3.0 1.5
```

Red cluster 's X11 is 2.75, X12 is 2.25.

Green cluster's X11 is 3.0, and x12 is 1.5.

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

Answer:
```
> distance <- function (x, y){
+     return(sqrt((x[1] - y[1])^2 + (x[2] - y[2])^2))
+ }
> distance(x[1,], centroid1)
[1] 3.691206
> distance(x[1,],centroid2)
[1] 3.905125
> distance(x[2,], centroid1)
[1] 2.263846
> distance(x[2,],centroid2)
[1] 2.061553
> distance(x[3,], centroid1)
[1] 1.457738
> distance(x[3,], centroid2)
[1] 1.802776
> distance(x[4,], centroid1)
[1] 1.767767
> distance(x[4,], centroid2)
[1] 2.061553
> distance(x[5,], centroid1)
[1] 2.150581
> distance(x[5,], centroid2)
[1] 2.061553
> distance(x[6,], centroid1)
[1] 3.553168
> distance(x[6,], centroid2)
[1] 3.354102
```

(e) Update the cluster assignments via K-Means algorithm until they stop changing. How many iterations did it take? Hint: it won't take long.

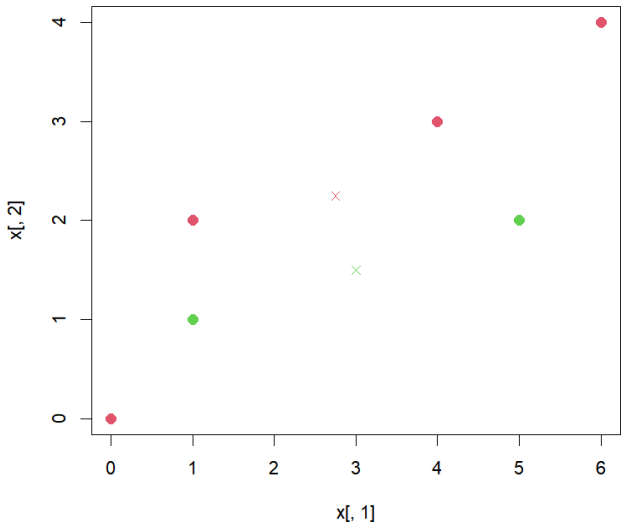
Answer: R-Programing code

```
set.seed(1)
centroid1 <- c(mean(x[, 1]), mean(x[, 2]))
centroid2 <- c(mean(x[, 1]), mean(x[, 2]))
```

```
# Function to find the closest centroid for each observation
find_closest_centroid <- function(x, centroid1, centroid2) {
  labels <- numeric(nrow(x))
  for (i in 1:nrow(x)) {
    dist1 <- distance(x[i,], centroid1)
    dist2 <- distance(x[i,], centroid2)
    labels[i] <- ifelse(dist1 < dist2, 1, 2)
  }
  return(labels)
}

# Function to update cluster assignments and centroids
update_kmeans <- function(x, labels, centroid1, centroid2) {
  new_labels <- find_closest_centroid(x, centroid1, centroid2)

  # Check if cluster assignments have changed
  if (!identical(labels, new_labels)) {
    # Recompute centroids
    centroid1 <- colMeans(x[new_labels == 1, ])
    centroid2 <- colMeans(x[new_labels == 2, ])

    return(list(labels = new_labels, centroid1 = centroid1, centroid2 = centroid2))
  } else {
    # If no change in cluster assignments, return NULL
    return(NULL)
  }
}

# Set the initial labels randomly
set.seed(1)
labels <- sample(2, nrow(x), replace = TRUE)

# Perform K-Means iterations until convergence
iteration <- 1
while (TRUE) {
  result <- update_kmeans(x, labels, centroid1, centroid2)

  # Break the loop if no change in cluster assignments
  if (is.null(result)) {
    break
  }
```

```
 # Update labels and centroids
 labels <- result$labels
 centroid1 <- result$centroid1
 centroid2 <- result$centroid2

 iteration <- iteration + 1
}

# Print the final cluster labels and the number of iterations
    print(paste("Number of iterations:", iteration))
"Number of iterations: 3"
```

(f) Update your plot from (a), by coloring the observations according to the cluster labels obtained.



Answer:

2. Considering a data set of temperature measurements for a certain place of interest, suppose you focus on an eight-hour period from 1:00 PM to 9:00 PM. The sensor reports the following temperatures:

Time	1:00	2:00	3:00	4:00	5:00	6:00	7:00	8:00	9:00
Temperature	48	54	55	60	62	61	56	55	51

You run a *K*-means clustering algorithm on 9 measurements, where $k = 3$, and the distance between items is the difference in temperature values.

(a) What are the resulting clusters? Please specify the clusters as sets of time values.

Answer: To perform K-means clustering, we need to pick three random initial centroids. Let's say we pick initial centroids: {C1 = 54,C2 = 62, C3= 56}

Now, let's assign each temperature measurement to the closest centroid and form three clusters.

Cluster1:{48,55,51} =>(1:00, 3:00, 9:00)

Cluster2:{60,61,62}=>(4:00, 6:00,5:00)

Cluster3:{54,55,56}=>(2:00,8:00,7:00)

(b) List the cluster means corresponding to the clusters you listed in part (a).

Answer:

Cluster1 = (48 + 55 + 51) /3 = 51.33

Cluster2 = (60 + 61 + 62) /3 = 61

Cluster3 = (54 + 55 + 56) / 3 = 55

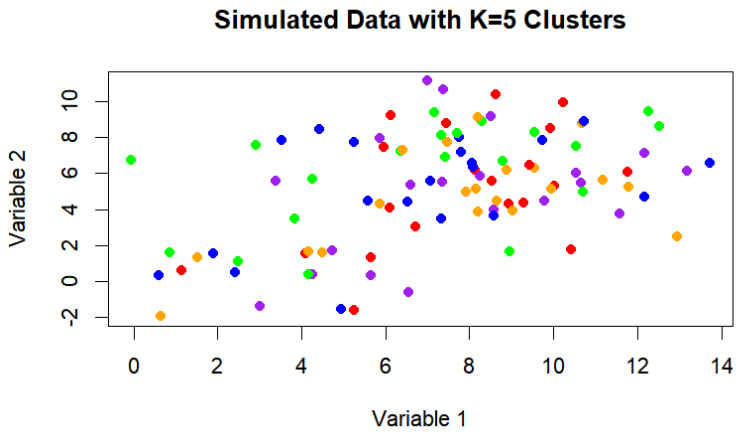
Applied.

3. In this problem, you will generate simulated data, and then perform *K*-means clustering on the data.

(a) Generate a simulated data set with 20 observations in each of **K=5** well-separated clusters, with $p = 2$ variables describing each observation. Do it in similar fashion to $K = 3$ case in "*K*-means: Selecting *K*" lecture slides. **Plot** the resulting data points.

Answer:

```
> set.seed(42)
>
> # Number of clusters (K), observations per cluster, and variables (p)
> K <- 5
> n_per_cluster <- 20
> p <- 2
>
> # Generate well-separated clusters
> centers <- matrix(runif(K * p) * 10, ncol = p)  # Random centers for ea
ch cluster
> data <- matrix(0, nrow = 0, ncol = p)  # Initialize an empty matrix
>
> # Vector of colors for each cluster
> cluster_colors <- c("red", "blue", "green", "purple", "orange")
>
> for (i in 1:K) {
+     # Generate data for each cluster with adjusted mean
+     cluster_data <- matrix(rnorm(n_per_cluster * p, mean = rep(centers
[i,], each = n_per_cluster), sd = 2), ncol = p)
+
+     # Combine data using rbind
+     data <- rbind(data, cluster_data)
+ }
>
> # Plot the resulting data points with different colors for each cluster
> plot(data[, 1], data[, 2], pch = 16, col = cluster_colors,
+      main = "Simulated Data with K=5 Clusters",
+      xlab = "Variable 1", ylab = "Variable 2")
```



(b) Run *K*-means algorithm on your simulated data for $K = 4, 5 \& 6$. Use 50 random starts in each case. Provide the code and report the total within-cluster sum of squares (WSS) for each solution. Which transition caused the bigger drop in total WSS, from $K = 4$ to $K = 5$, or from $K = 5$ to $K = 6$?

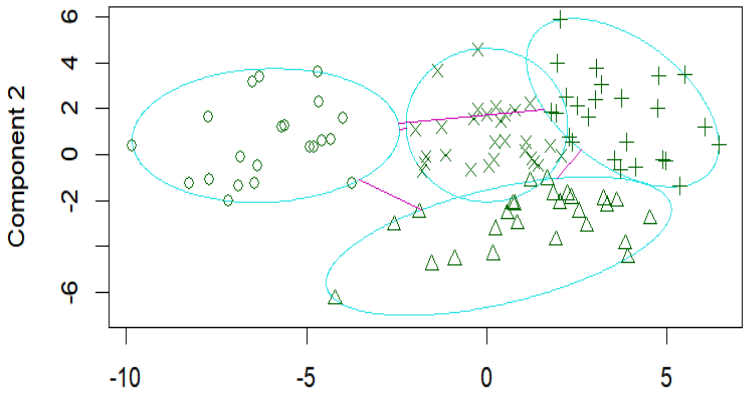
Answer:

```
> set.seed(42)
>
> # Number of clusters for K-means
> k_values <- c(4, 5, 6)
>
> # Initialize a list to store results
> results <- list()
>
> # Iterate over different values of K
> for (k in k_values) {
+     # Run K-means with 50 random starts
+     kmeans_result <- kmeans(data, centers = k, nstart = 50)
+
+     # Store the results
+     results[[as.character(k)]] <- kmeans_result
+ }
>
> # Print the total within-cluster sum of squares (WSS) for each so
lution
> for (k in k_values) {
+     cat("K =", k, "Total WSS:", sum(results[[as.character(k)]]$wi
thinss), "\n")
+ }
K = 4 Total WSS: 462.5907
K = 5 Total WSS: 363.9605
K = 6 Total WSS: 307.9966
```

K= 4 to K = 5 caused the biggest drop in total WSS, as the total WSS decreased 462.5907 to 363.9605. The difference between K= 4 to K =5 is 98.63

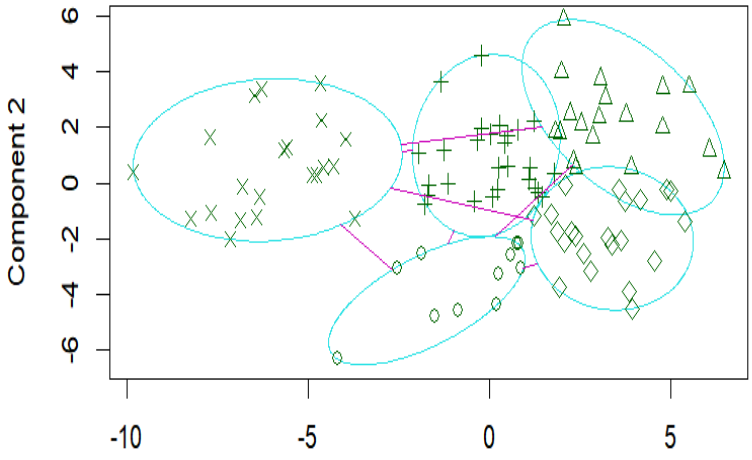
(c) Proceed to plot the clustering solutions for $K = 4, 5 \& 6$ (just use the plots automatically generated by *eclust*() function). Judging by the plots, explain why the respective sizes of WSS drops in part (*b*) were expected. Use lecture slides (the $K = 3$ simulation study) for reference.

Clustering Plot for K = 4

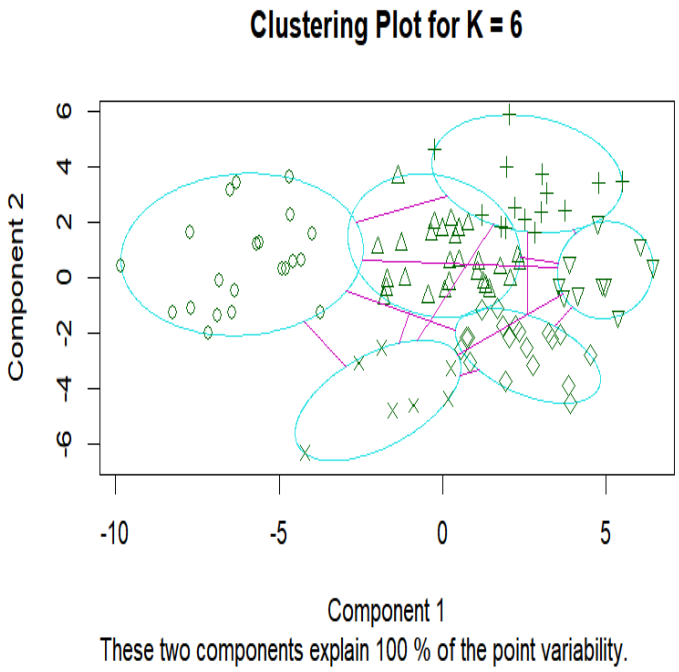


These two components explain 100 % of the point variability.

Clustering Plot for K = 5

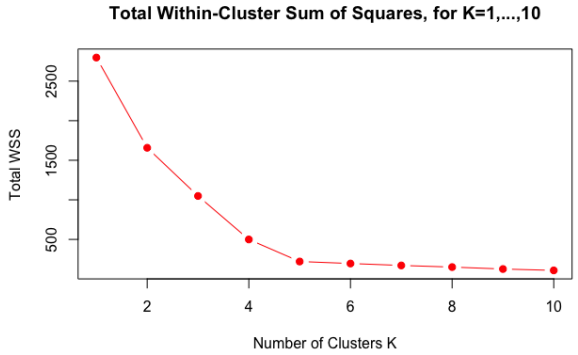


These two components explain 100 % of the point variability.



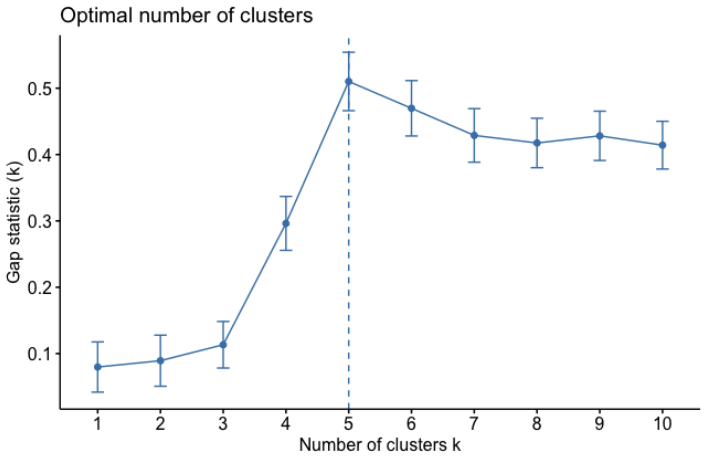
Answer: Looking at the values for each k value, we can visualize the clusters and it makes sense why the WSS decreased for each k. When looking at 4 clusters, we can see that having 5 clusters would be better, because k =4 cluster line overlaps some area, but k = 5, cluster doesn't overlap each cluster. In WSS, we can see it decrease substantially to reflect what we see. When looking at 6 clusters, we see that the clusters have become little overlapping again.

(d) Run *K*-means clustering on your simulated data for $K = 1, 2, \ldots, 10$, record total WSS for each K value. Plot the progression of WSS values. According to "elbow" method logic, which K value appears as the optimal one?



Answer: "Elbow" is k=5 which is consistent with our analysis in parts (b) and (c). Therefore, K = 5 appears to be the optimal number of clusters for this dataset

(e) For a more formal approach (alternative to elbow method from part (d)), run the gap statistic calculations for $K = 1, \ldots, 10$, with 50 replicates each. Which K value is optimal? Does it match with the # of well-separated clusters in our simulated data? Provide the plot of gap statistic values.



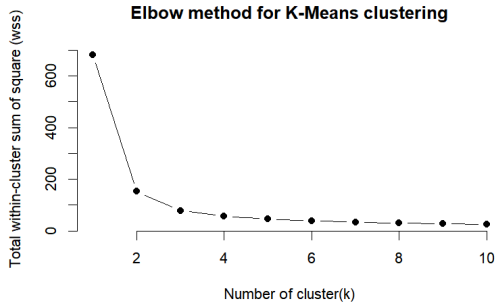
Answer:

K = 5 is optimal and it matches with gap statistic plot.

4. This problem will work with *iris* data (available in base *R*). Make sure to exclude the *Species* column before applying clustering algorithms.

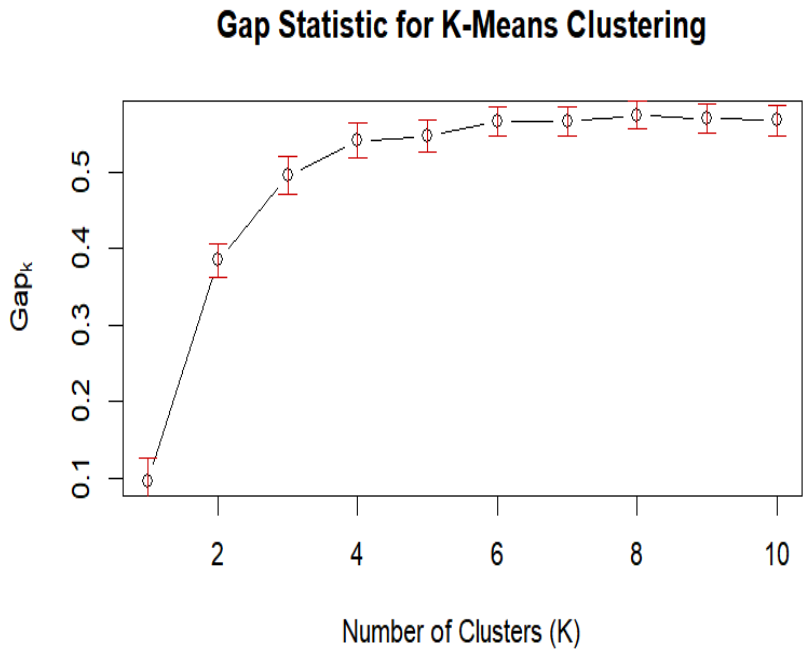
   (a) Apply *K*-Means clustering with $K = 1, 2, \ldots, 10$ via *eclust*(), for *nstarts* = 50 each. Provide the code and the plot of total within-cluster sum of squares (WSS) progression for the resulting clustering solutions.

Answer:
```
> data(iris)
> iris_data<-iris[,-5]
> k_value <- 1:10
> kmeans_results <-lapply(k_value,function(k) kmeans(iris_data, centers = k, nstart = 50))
> wss_values <- sapply(kmeans_results, function(x) x$tot.withinss)
>
> plot(k_value, wss_values, type = "b",pch = 19, frame = FALSE
+      ,xlab = "Number of cluster(k)"
+      , ylab = "Total within-cluster sum of square (wss)"
+      ,main = "Elbow method for K-Means clustering")
```



   (b) Run the gap statistic calculations for $K = 1, \ldots, 10$, with 50 replicates each. Provide the gap statistic plot. Which *K* value appears to be optimal?

Answer:
```
> gap_stat <- clusGap(iris_data, FUN = kmeans, nstart = 50, K.max = 10, B = 50)
Clustering k = 1,2,..., K.max (= 10): .. done
Bootstrapping, b = 1,2,..., B (= 50)  [one "." per sample]:
.................................................. 50
> plot(gap_stat, main = "Gap Statistic for K-Means Clustering", xlab = "Number of Clusters (K)")
> optimal_k <- which.max(gap_stat$Tab[, "gap"])
> cat("Optimal K value according to gap statistic:", optimal_k, "\n")
Optimal K value according to gap statistic: 8
```



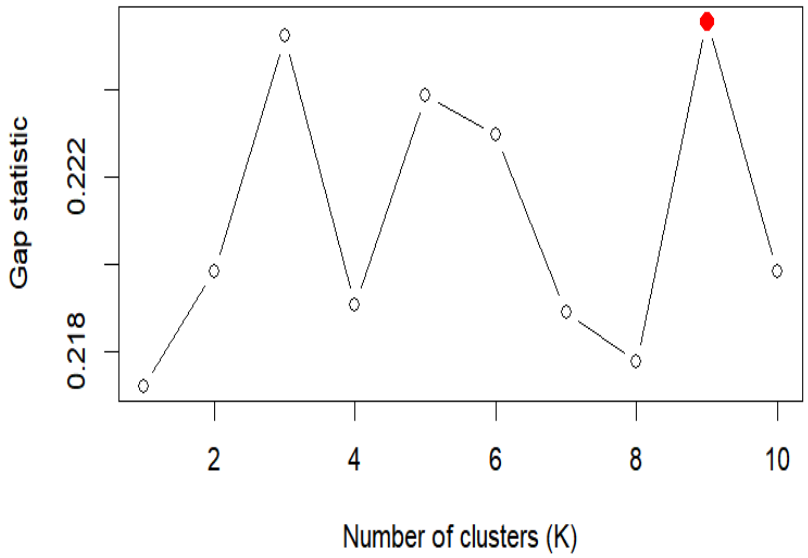
(c) Using external cluster validation, and comparing results of your clustering to the actual # of distinct iris species in the data (≡ 3), is the answer in part (b) close to it?

Answer:

```
• iris_data <- iris[,-5]
• > set.seed(123)
• > k8<-kmeans(iris_data, centers = 8, nstart = 50)
• > library(mclust)
• true_labels<-rep(c("Iris-setosa","Iris-versicolor","Iris-v
    irginica"),each =50)
• > adjustedRandIndex(true_labels,k8$cluster)

    It is 0.4365301 what I got for distinct, it's actually cl
    oser 0 more than 1 so it is not really close part(b)
```

(d) Proceed to scale iris data and run the gap statistic calculations for K =1, . . . , 10, with 50 replicates each. Which K value is optimal?

Answer: It seems to be k = 9 because optimal K value is between 8 and 10



(e) Using external cluster validation, and comparing results of your clustering to the actual # of distinct iris species in the data (≡ 3), is the answer in part (d) close to it?

Answer:
```
> data(iris)
> iris_data <- scale(iris[,-5])
> set.seed(123)
> k8<-kmeans(iris_data, centers = 9, nstart = 50)
> library(mclust)
    __           __
   ____ ___  _____/ /_  _______/ /_
  / __ `__ \/ ___/ / / / / ___/ __/
 / / / / / / /__/ / /_/ (__  ) /_
/_/ /_/ /_/\___/_/\__,_/____/\__/   version 6.0.0
Type 'citation("mclust")' for citing this R package in publications.
Warning message:
패키지 'mclust'는 R 버전 4.2.3 에서 작성되었습니다
> true_labels<-rep(c("Iris-setosa","Iris-versicolor","Iris-virginic
a"),each =50)
> adjustRandIndex(true_labels,k8$cluster)
Error in adjustRandIndex(true_labels, k8$cluster) :
  could not find function "adjustRandIndex"
> adjustedRandIndex(true_labels,k8$cluster)
[1] 0.3509633
```

Answer: The value is 0.3509633 so it is still not closed

(f) Apply K-means with optimal K selected in part (d). Compare the resulting cluster assignments to the actual species labels. Is there some correspondence? E.g. do any of your clusters contain only elements of a certain species? Or maybe a combination of two species?

```
> data(iris)
> iris_data <- iris[,-5]
> set.seed(123)
> k8<-kmeans(iris_data, centers = 9, nstart = 50)
> true_labels<-rep(c("Iris-setosa","Iris-versicolor","Iris-virginica"),ea
ch =50)
> table(k8$cluster,true_labels)
   true_labels
    Iris-setosa Iris-versicolor Iris-virginica
  1           0              19              0
  2           0               0             12
  3           0              20              1
  4           0               7              0
  5           0               4             15
  6           8               0              0
  7           0               0             22
  8          19               0              0
  9          23               0              0
```

Answer: Most of clusters contained at least 1 factor but majority of clusters didn't contain 2 factors because both distinct are closer 0 which means result is pretty bad.

5. On the book website, www.StatLearning.com, there is a gene expression data set (Ch10Ex11.csv) that consists of 40 tissue samples with measurements on 1, 000 genes. The first 20 samples are from healthy patients, while the second 20 are from a diseased group.

(a) Load in the data using read.csv(). You will need to select header = F. Also make sure to transpose the resulting matrix (use t() operation in R), as initially, for some reason, it contains variables as rows, and observations as columns. Should be the other way around.

Answer:

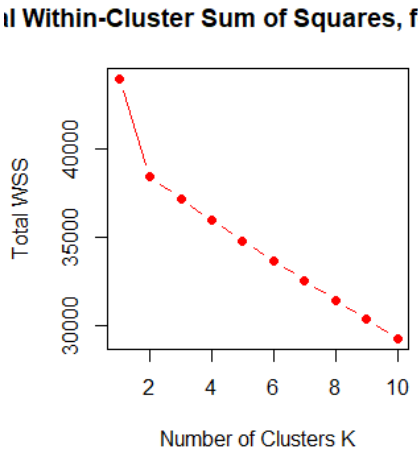
```
> library(readr)
> Ch10Ex11 <- read_csv("C:/Users/sec/Downloads/Ch10Ex11.csv")
Rows: 999 Columns: 40

── Column specification ────────────────────────────────────────────────
────────────────
Delimiter: ","
dbl (40): -0.9619334, 0.4418028, -0.9750051, 1.417504, 0.8188148, 0.31629
37, -0.02496682...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this m
essage.
> > t(Ch10Ex11)
```

(b) Apply K-Means clustering with K = 1, 2, . . . , 10 via eclust(), for nstarts = 50 each. Provide the code and the plot of total within-cluster sum of squares (WSS) progression for the resulting clustering solutions. Does there appear to be an "elbow"? If yes, at which value of K?

Answer:

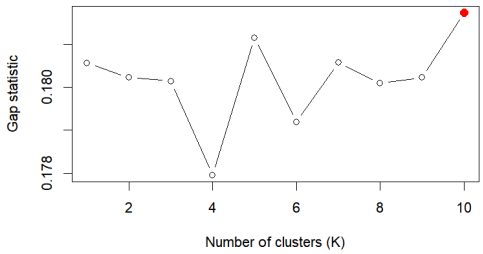
```
> k.max <- 10
> wss <- numeric(k.max)
> for (k in 1:k.max){
+   wss[k] <- eclust(x,
```

```
+                 FUNcluster = "kmeans",
+                 k=k,
+                 nstart=50)$tot.withinss
+         }
> plot(wss,
+     type="b", pch=19, col=2,
+     main = "Total Within-Cluster Sum of Squares, for K=1,...,10",
+     xlab = "Number of Clusters K", ylab = "Total WSS")
```



Answer: The elbow is k = 2

(c) Run the gap statistic calculations for K = 1, . . . , 10, with 20 replicates each (Note: it might take some time). Provide the plot. Which K value is optimal?



Answer:
It seems to be k = 10 is optimal

(d) Using external cluster validation, and comparing results of your clustering to the actual # of patient group (which is 2 - healthy and diseased), is the answer in part (c) close to it?

Answer: The ARI for the clustering with K=10 is 0.23, which is relatively low. This suggests that the lustering is not very similar to the true labels of healthy vs diseased patients. It's possible that the gap statistic method may have overestimated the optimal number of clusters for this dataset.

(e)Apply K-means with optimal K selected in part (c). Compare the resulting cluster assignments to the actual patient groups. Is there correspondence? E.g. do any of your clusters contain only healthy (or only diseased) patients?

Answer:

```
> set.seed(123)
> k10 <- kmeans(gene_date, centers = 10, nstart = 50)
> true_labels <- rep(c("healthy","diseased"),each = 20)
> table(k10$cluster,true_labels)
    true_labels
     diseased healthy
  1          4       0
  2          0       4
  3          1       0
  4          5       0
  5          0       4
  6          0       4
  7          4       0
  8          0       2
  9          6       0
  10         0       6
```

Answer: We can see that cluster 1,2,3,4,9 contains all healthy patients, while cluster 5,6,7,8,10, contains diseased patients (out of 20). Overall, there is some correspondence between the cluster assignments and the actual patient groups, but it's not perfect.