CW4

Wen Hans Tan and Eirshad Fahim

2025-02-30

Problem 2: Practical (20 marks)

```
library(readr)
moons_data <- read_csv("C:/Users/tanwe/OneDrive/Documents/Stats_Machine_Learning/CW4/moons.csv")
```

```
## Rows: 300 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (2): X, Y
##
## 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 message.
```

```
colnames(moons_data)
```

```
## [1] "X" "Y"
```

```
head(moons_data)
```

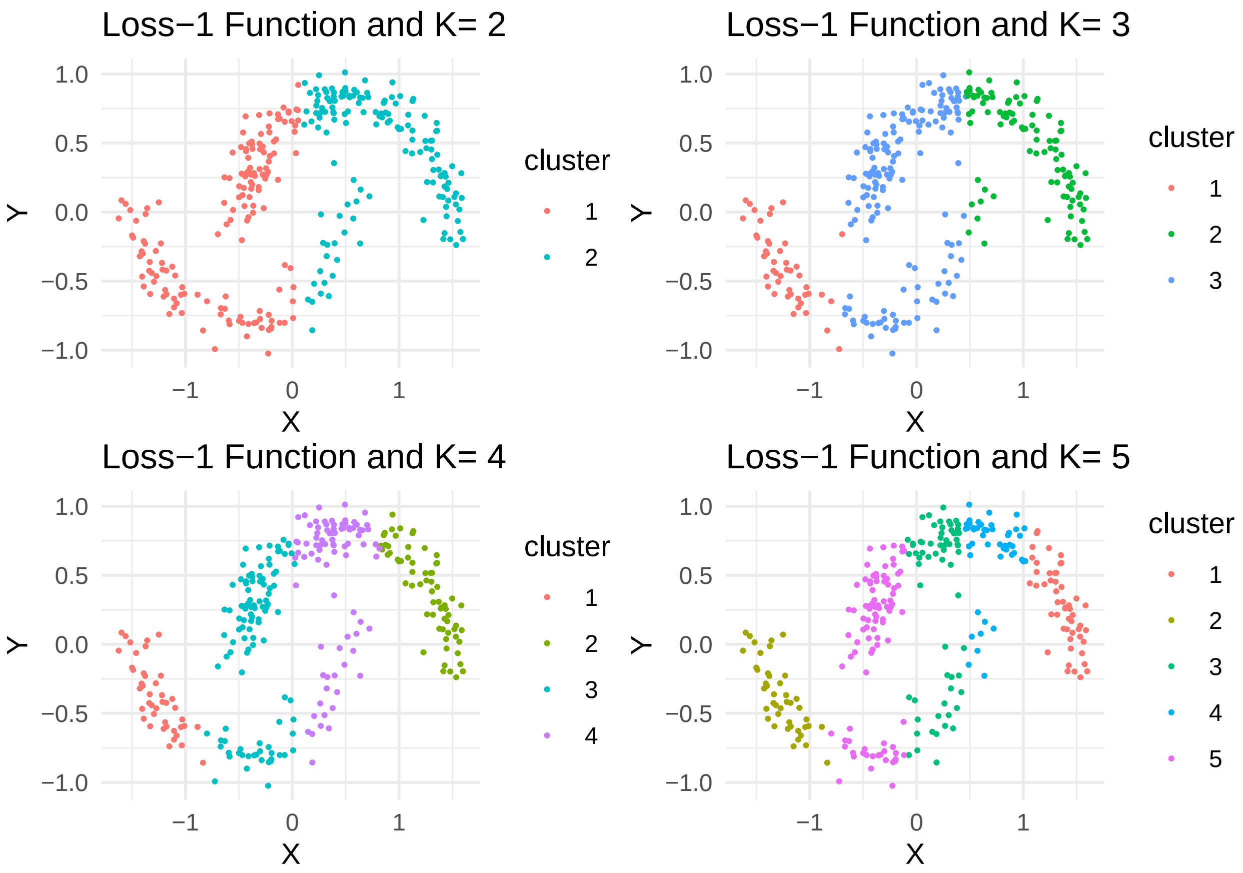
```
## # A tibble: 6 x 2
##        X       Y
##    <dbl>   <dbl>
## 1 -0.277  0.474
## 2 -1.11  -0.627
## 3  1.23  -0.0580
## 4  0.601  0.0766
## 5 -0.246  0.317
## 6 -0.408  0.287
```

(i) **{10 marks}** Load in $ `moons.csv`$, and use the native $ `kmeans`$ function in R to cluster the data into $K = \{2, 3, 4, 5\}$ components, according to each of the following loss functions :

```
#Loading necessary libraries
library(ggplot2)
library(gridExtra)
```

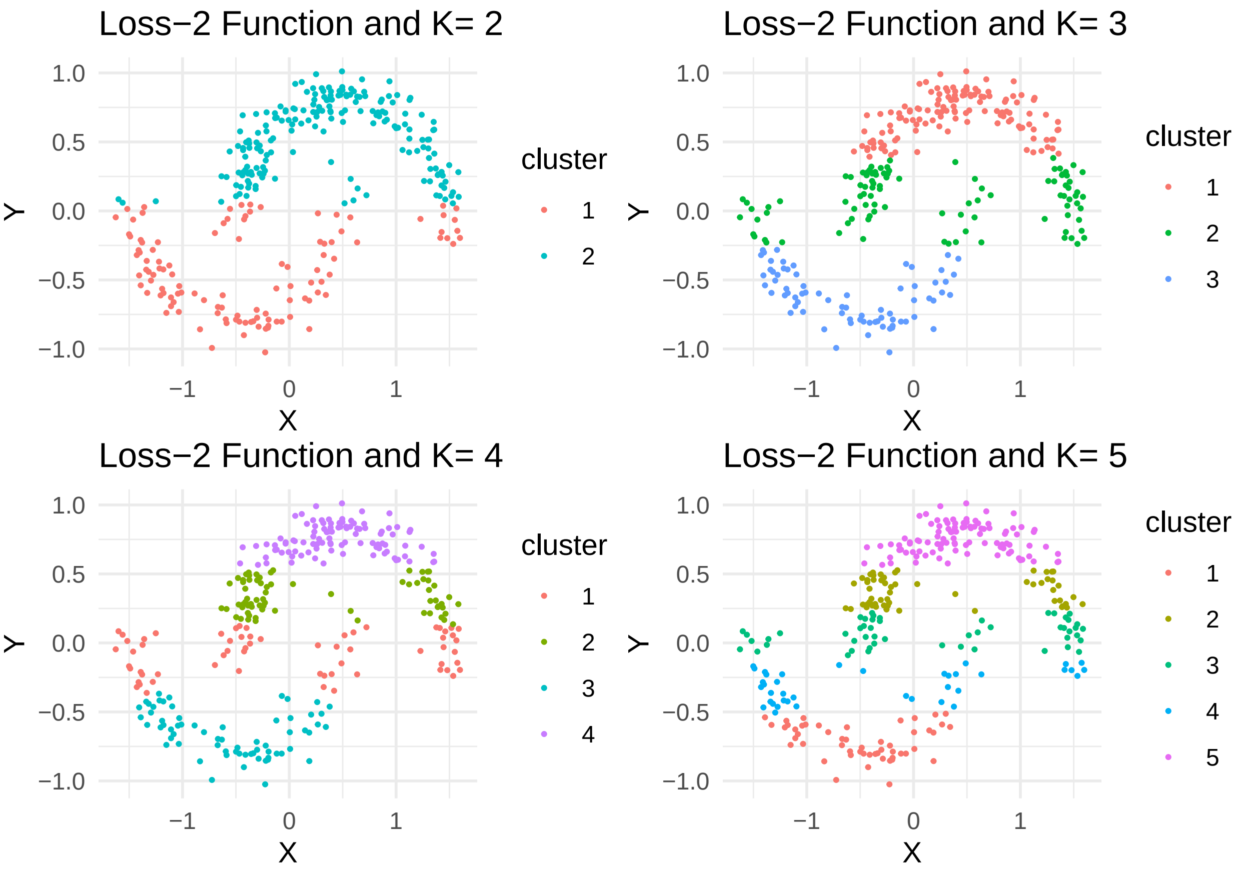
```
## Warning: package 'gridExtra' was built under R version 4.4.3
```

```r
set.seed(420)

#Define K values
K_values <- c(2,3,4,5)

#Seperate Data
X_1 <- moons_data[,1]

#List to store plots
plots_list <- list()

#Loooping through each value
for (k in K_values){
  #Apply K-Means Clustering Algorithm
  result <- kmeans(X_1, centers=k, nstart=50)

  #Store cluster assignments
  moons_data$cluster <- as.factor(result$cluster)

  #Scatter Plot for each K-value
  plot <- ggplot(moons_data, aes(x= X, y= Y, color=cluster))+
          geom_point(size=0.4) +
          ggtitle(paste("Loss-1 Function and K=",k)) +
          labs(x = "X", y= "Y") +
          theme_minimal() +
          theme(plot.margin = margin(1, 1, 1, 1, "pt"))

  #Store plot in the list
  plots_list[[length(plots_list) + 1]] <- plot
}

grid.arrange(grobs = plots_list, nrow = 2, ncol = 2, widths = c(3,3), height = c(3,3))
```



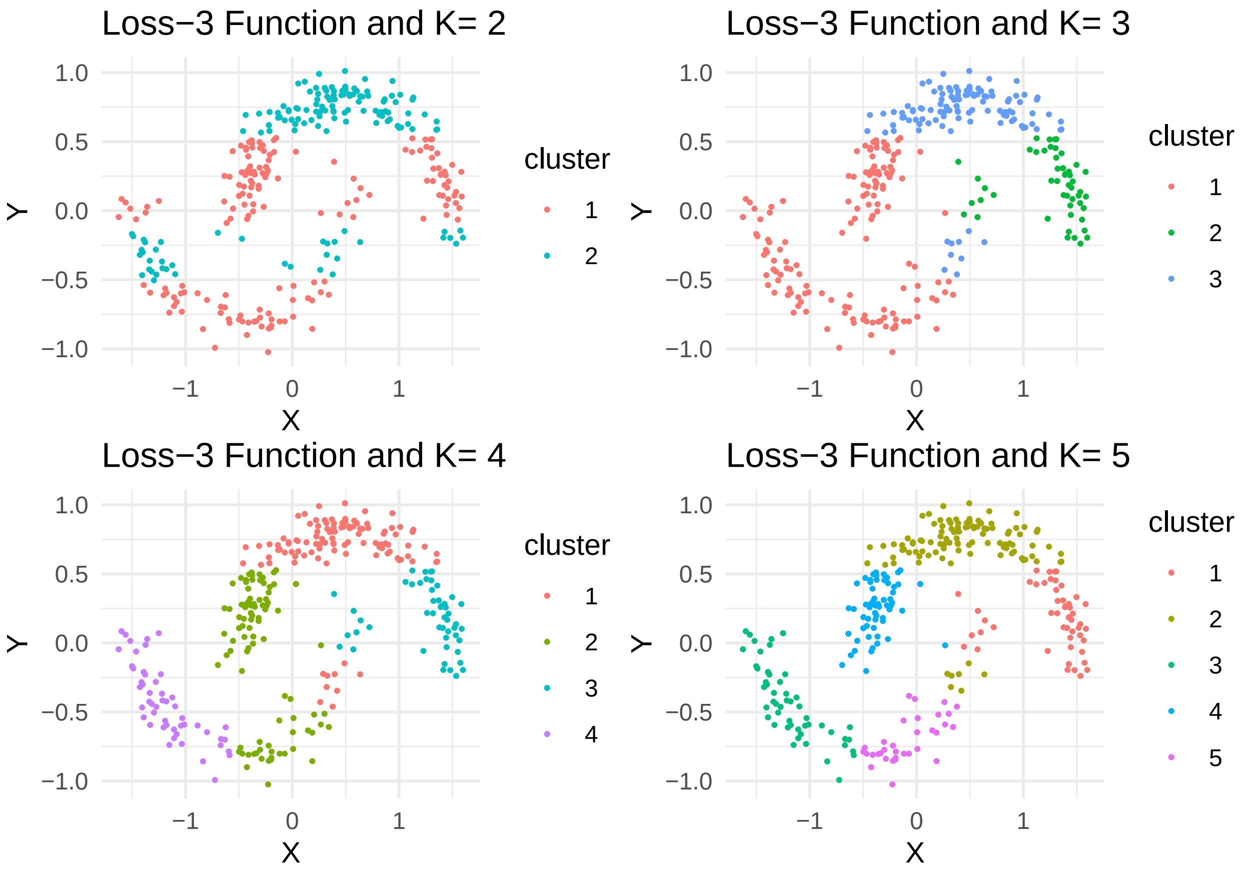
Now, using the Loss-2 Function:

```
#Seperate Data
X_2 <- moons_data[,2]

#List to store plots
plots_list <- list()

#Define K values
K_values <- c(2,3,4,5)

#Loooping through each value
for (k in K_values){
  #Apply K-Means Clustering algorithm
  result <- kmeans(X_2, centers=k, nstart=50)

  #Store cluster assignments
  moons_data$cluster <- as.factor(result$cluster)

  #Scatter Plot for each K-value
  plot <- ggplot(moons_data, aes(x= X, y= Y, color=cluster))+
          geom_point(size=0.4) +
          ggtitle(paste("Loss-2 Function and K=",k)) +
          labs(x = "X", y= "Y") +
          theme_minimal() +
          theme(plot.margin = margin(1, 1, 1, 1, "pt"))
  #Store plot in the list
  plots_list[[length(plots_list) + 1]] <- plot
}

grid.arrange(grobs = plots_list, nrow = 2, ncol = 2, widths = c(3,3), height = c(3,3))
```



Now, using the Loss-3 Function, where the full feature vector is utilised:

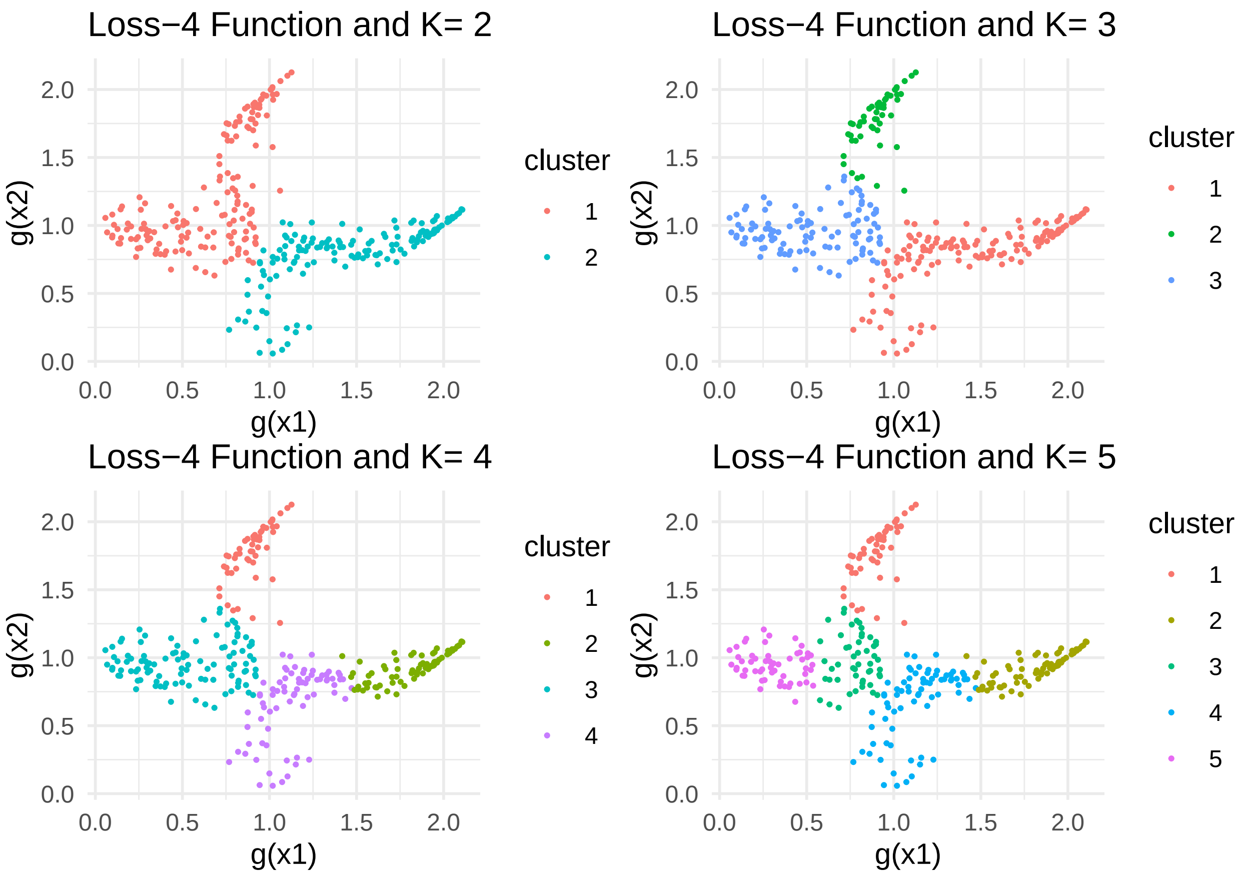
```
#List to store plots
plots_list <- list()

#Define K values
K_values <- c(2,3,4,5)

#Loooping through each value
for (k in K_values){
  #Apply K-Means Clustering Algorithm
  result <- kmeans(moons_data, centers=k, nstart=50)

  #Store cluster assignments
  moons_data$cluster <- as.factor(result$cluster)

  #Scatter Plot for each K-value
  plot <- ggplot(moons_data, aes(x= X, y= Y, color=cluster))+
          geom_point(size=0.4) +
          ggtitle(paste("Loss-3 Function and K=",k)) +
          labs(x = "X", y= "Y") +
          theme_minimal() +
          theme(plot.margin = margin(1, 1, 1, 1, "pt"))

  #Store plot in the list
  plots_list[[length(plots_list) + 1]] <- plot
}

grid.arrange(grobs = plots_list, nrow = 2, ncol = 2, widths = c(3,3), heights = c(3,3))
```



Now, we use the last loss function, where there is a transformation:

```
#List to store plots
plots_list <- list()

#Define K values
K_values <- c(2,3,4,5)

#Transform Function
transform_to_g <- function(x1,x2) {
  g_x1 <- sqrt((x1 + 1/2)^2 + x2^2)
  g_x2 <- sqrt((x1 - 1/2)^2 + x2^2)
  return(c(g_x1, g_x2))
}
```

```r
# Apply the transformation using sapply()
transformed_data <- as.data.frame(t(sapply(1:nrow(moons_data),
                                           function(i) transform_to_g(moons_data[i,1], moons_data[i,2]))

#Set new column names
colnames(transformed_data) <- c("gx1", "gx2")

#Convert into numeric data
transformed_data$gx1 <- as.numeric(transformed_data$gx1)
transformed_data$gx2 <- as.numeric(transformed_data$gx2)

#Loooping through each value
for (k in K_values){
  #Apply K-Means Clustering Algorithm
  result <- kmeans(transformed_data, centers=k, nstart=50)

  #Store cluster assignments
  transformed_data$cluster <- as.factor(result$cluster)

  #Scatter Plot for each K-value
  plot <- ggplot(transformed_data, aes(x= gx1, y= gx2, color=cluster))+
          geom_point(size=0.4) +
          ggtitle(paste("Loss-4 Function and K=",k)) +
          labs(x = "g(x1)", y= "g(x2)") +
          theme_minimal() +
          theme(plot.margin = margin(1, 1, 1, 1, "pt"))

  #Store plot in the list
  plots_list[[length(plots_list) + 1]] <- plot
}

grid.arrange(grobs = plots_list, nrow = 2, ncol = 2, widths = c(3,3), heights = c(3,3))
```

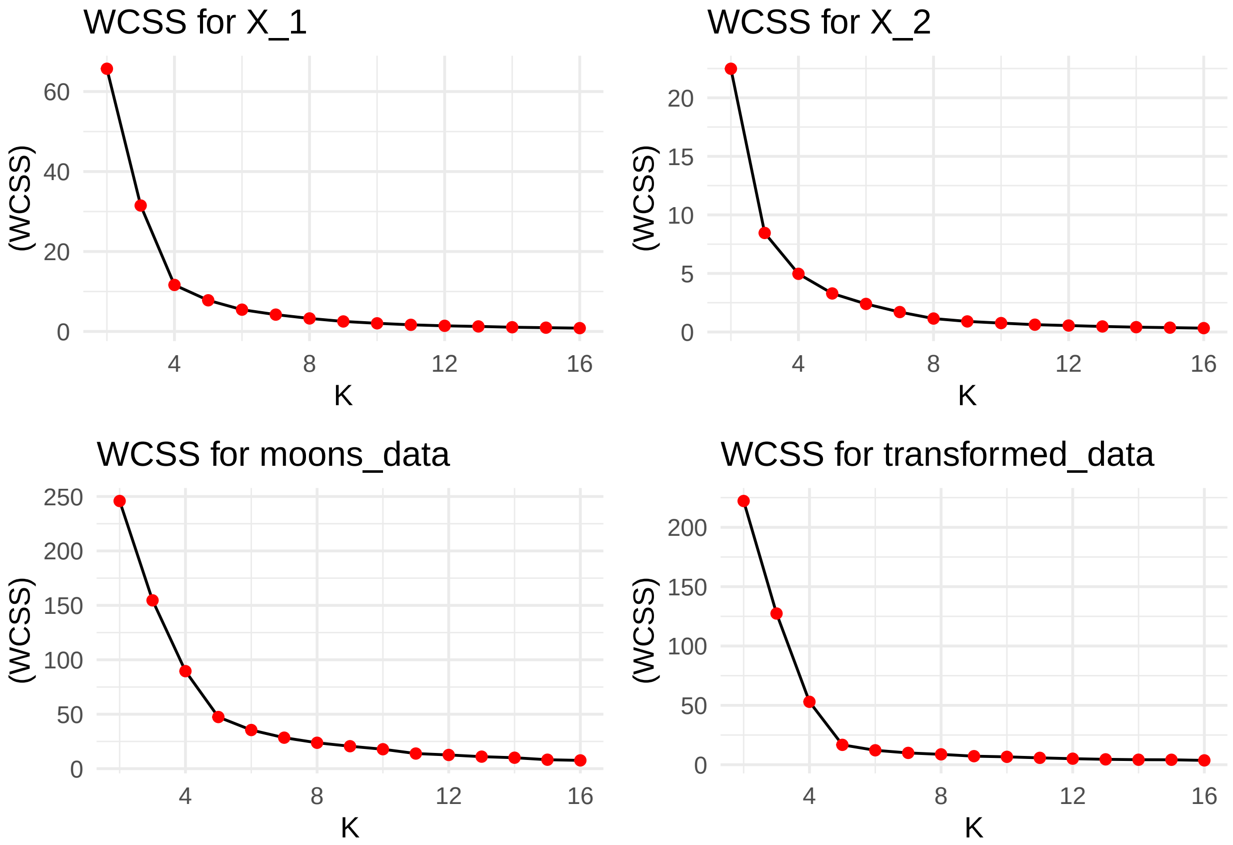


(ii) (**5 marks**) Write a function `cluster_eval` which, for a given data set X and a range of natural numbers $K_1 \leq K_2$,

- Computes the K-Means clusterings for each $K_1 \leq K \leq K_2$, and
- Given this collection of clusterings, plots the values of the K-Means loss function at the optimal clusterings, as a function of K.

Use your implementation of `cluster_eval` to compare the clusterings which you obtained in Part (i) of this problem. Do you find that you different clusterings agree on what the 'correct' number of clusters in this data are? **):**

```r
library(ggplot2)

cluster_eval <- function(K1, K2){
  #Create empty list to store plots
  plots_list <- list()

  # Create empty numeric vectors to store WCSS
  wcss_values <- list(
    wcss_value_1 = numeric(length = K2 - K1 + 1),
    wcss_value_2 = numeric(length = K2 - K1 + 1),
    wcss_value_3 = numeric(length = K2 - K1 + 1),
    wcss_value_4 = numeric(length = K2 - K1 + 1)
  )

  #Loop values from K1 to K2
  for (k in K1:K2){

    #Apply k-means algorithm to all clusterings
    result_loss_1 <-kmeans(X_1, centers=k, nstart = 50)
    result_loss_2 <- kmeans(X_2, centers=k, nstart = 50)
    result_loss_3 <- kmeans(moons_data, centers=k, nstart = 50)
    result_loss_4 <- kmeans(transformed_data, centers=k, nstart = 50)

    #Store the WCSS values
    wcss_values$wcss_value_1[k - K1 + 1] <- result_loss_1$tot.withinss
    wcss_values$wcss_value_2[k - K1 + 1] <- result_loss_2$tot.withinss
    wcss_values$wcss_value_3[k - K1 + 1] <- result_loss_3$tot.withinss
    wcss_values$wcss_value_4[k - K1 + 1] <- result_loss_4$tot.withinss
  }

  # Create individual plots for each dataset
  datasets <- list("X_1" = wcss_values$wcss_value_1,
                   "X_2" = wcss_values$wcss_value_2,
                   "moons_data" = wcss_values$wcss_value_3,
                   "transformed_data" = wcss_values$wcss_value_4)

  for (i in seq_along(datasets)) {
    wcss_df <- data.frame(K = K1:K2, WCSS = datasets[[i]])

    plot <- ggplot(wcss_df, aes(x = K, y = WCSS)) +
      geom_line(color = "black") +
      geom_point(size = 1.5, color = "red") +
      ggtitle(paste("WCSS for", names(datasets)[i])) +
      labs(x = "K", y = "(WCSS)") +
      theme_minimal()

    # Store each plot in the list
    plots_list[[i]] <- plot
  }

  #Arrange plots in a 2x2 grid
  grid.arrange(grobs = plots_list, nrow = 2, ncol = 2, widths = c(3, 3), heights = c(3, 3))
}

#Use cluster_eval function
cluster_eval(2, 16)
```


WCSS for X_1

WCSS for X_2

WCSS for moons_data

WCSS for transformed_data

We plotted the WCSS function against k for all types of loss function. We can examine the elbow point in the plot ; it looks like the optimal value of K for their respective loss functions are:

- $Loss^1$: K= 4
- $Loss^2$: K= 4
- $Loss^3$: K = 5
- $Loss^4$: K= 5

From part (i), depending on the loss function, we can inspect **visually** what the optimal value of K could be. In my opinion, these are the optimal values of K for each function:

- $Loss^1$: K=3
- $Loss^2$: K= 4
- $Loss^3$: K = 4
- $Loss^4$: K= 4

Again, these are completely subjective to each person. Since most of the "correct" cluster number is differs at most by 1, choices of K chosen using the elbow method is more or less correct.

iii. **(5 marks)**Load in R's quakes dataset. This dataset stores information about 1000 seismic events (i.e. earthquakes and similar) which have occurred in the vicinity of the island of Fiji over the past 60 or so years. The first three columns (lat, long, depth) record the geographical location of the event, the fourth (mag) records the strength of the event (as measured on the Richter scale), and the the fifth (station) records how many stations documented the event in question.

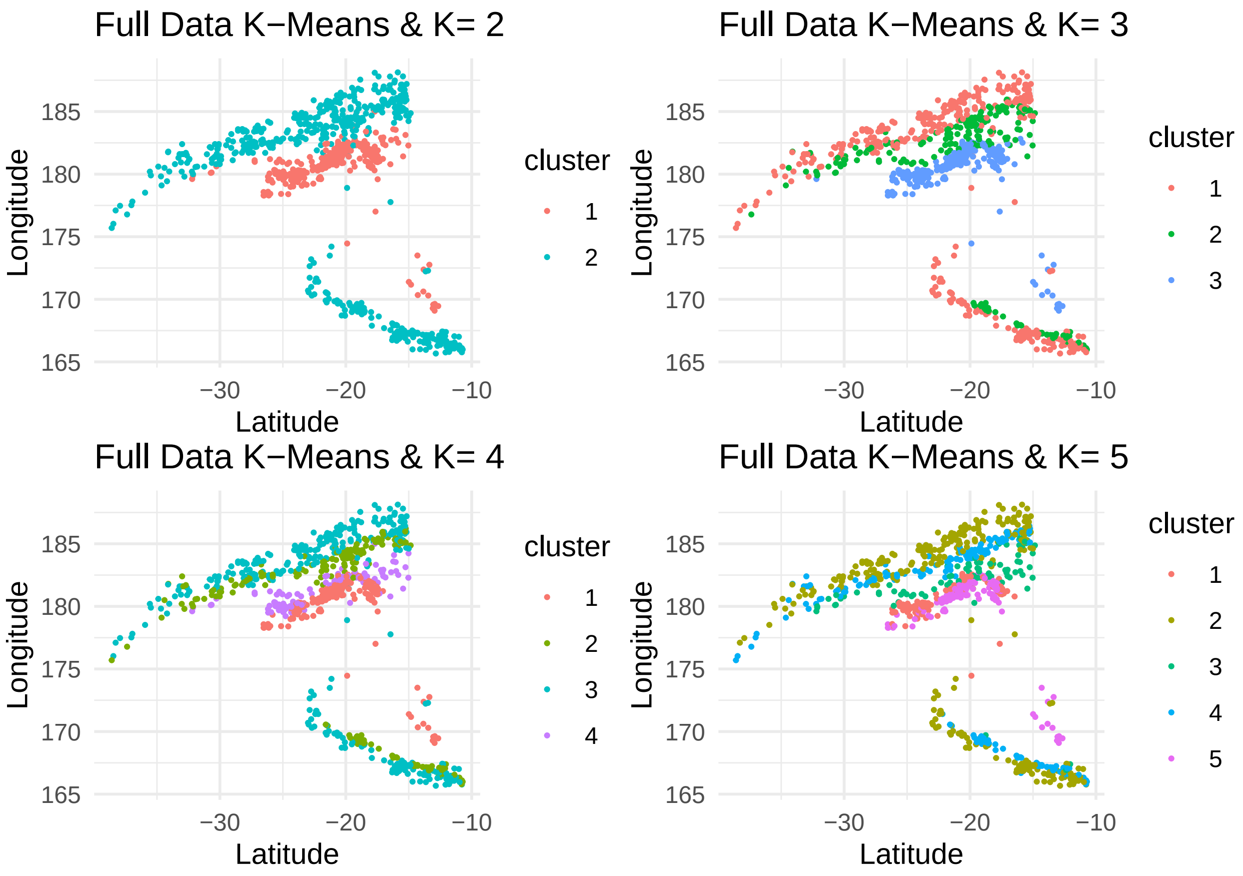
In the context of this question, we are interested in whether the locations at which these seismic events take place are spatially localised, i.e. if they cluster around specific areas of the island.

- Run a standard K-Means clustering on the full data set for $K = \{2, 3, 4, 5\}$, and plot the outcome, focusing on the `lat` and `long` dimensions.
- Does the clustering which you obtain in this way look spatially reasonable? Propose an alternative approach to obtaining a clustering which is more coherent (still using K-Means).

```r
data("quakes")
head(quakes)
```

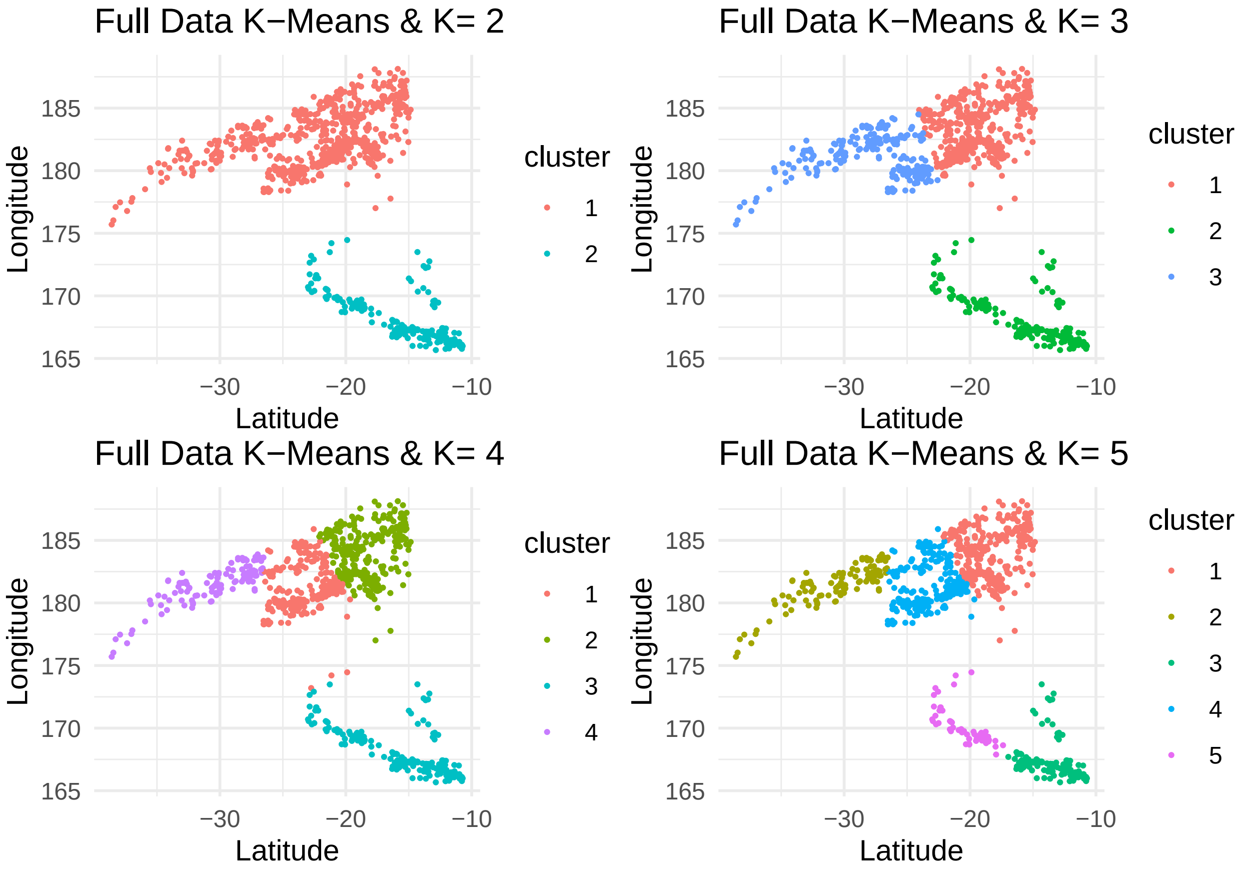
```
##      lat   long depth mag stations
## 1 -20.42 181.62   562 4.8       41
## 2 -20.62 181.03   650 4.2       15
## 3 -26.00 184.10    42 5.4       43
## 4 -17.97 181.66   626 4.1       19
## 5 -20.42 181.96   649 4.0       11
## 6 -19.68 184.31   195 4.0       12
```

```r
#List to store plots
plots_list <- list()

#Define K values
K_values <- c(2,3,4,5)

#Loooping through each value
for (k in K_values){
  #Apply K-Means Clustering Algorithm
  result <- kmeans(quakes, centers=k, nstart=50)

  #Store cluster assignments
  quakes$cluster <- as.factor(result$cluster)

  #Scatter Plot for each K-value
  plot <- ggplot(quakes, aes(x= lat, y= long, color=cluster))+
          geom_point(size=0.4) +
          ggtitle(paste("Full Data K-Means & K=",k)) +
          labs(x = "Latitude", y= "Longitude") +
          theme_minimal() +
          theme(plot.margin = margin(1, 1, 1, 1, "pt"))

  #Store plot in the list
  plots_list[[length(plots_list) + 1]] <- plot
}

grid.arrange(grobs = plots_list, nrow = 2, ncol = 2, widths = c(3,3), heights = c(3,3))
```



For $K = 2$, the clusters are well seperated but there are some overlapping points ; overall, it looks spatially reasonable. As the number of clusters increase, they appear more fragmented and artificially split, especially for $K \in \{4, 5\}$. An alternative approach while still using K-Means will be to do k-means with only `lat` and `long` , as including the other features does not help in finding whether location of the seismic events are *spatially localised.*

```
#List to store plots
plots_list <- list()

#Chooses lat and long only
quakes_clean <- subset(quakes, select = -c(depth, mag, stations))

#Loooping through each value
for (k in K_values){
  #Apply K-Means Clustering Algorithm
  result <- kmeans(quakes_clean, centers=k, nstart=50)

  #Store cluster assignments
  quakes_clean$cluster <- as.factor(result$cluster)

  #Scatter Plot for each K-value
  plot <- ggplot(quakes_clean, aes(x= lat, y= long, color=cluster))+
          geom_point(size=0.4) +
          ggtitle(paste("Full Data K-Means & K=",k)) +
          labs(x = "Latitude", y= "Longitude") +
          theme_minimal() +
          theme(plot.margin = margin(1, 1, 1, 1, "pt"))

  #Store plot in the list
  plots_list[[length(plots_list) + 1]] <- plot
}

grid.arrange(grobs = plots_list, nrow = 2, ncol = 2, widths = c(3,3), heights = c(3,3))
```



Now, it looks spatially localised! Clusters are more identifiable!