Assignment 3: K Means Clustering Response

Timothy Lee

19/10/2019

Note that I didn't save the clustering results, so each time I knit the document we get different clustering results. Hence, the output for the graphs and network diagrams will vary, unfortunately.

In this assignment we will be applying the K-means clustering algorithm we looked at in class. At the following link you can find a description of K-means:

https://www.cs.uic.edu/~wilkinson/Applets/cluster.html

```
#For K-means analysis
library(klaR)
```

```
## Loading required package: MASS
```

```
#For Data Wrangling
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.1          v purrr   0.3.2
## v tibble  2.1.3          v dplyr   0.8.3.9000
## v tidyr   1.0.0          v stringr 1.4.0
## v readr   1.3.1          v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::select() masks MASS::select()
```

```
#For mosaic plot
library(ggmosaic)
```

Now, upload the file "Class_Motivation.csv" from the Assignment 3 Repository as a data frame called "K1""

```
#Read in our data frame
K1 <- read.csv("Class_Motivation.csv")
```

This file contains the self-reported motivation scores for a class over five weeks. We are going to look for patterns in motivation over this time and sort people into clusters based on those patterns.

But before we do that, we will need to manipulate the data frame into a structure that can be analyzed by our clustering algorithm.

The algorithm will treat each row as a value belonging to a person, so we need to remove the id variable.

```r
# We don't want the ID variable included in the clustering. If the ID variable is included, it will be
# as a variable in the clustering analysis.
# Get rid of it.
K2 <- select(K1, -id)
```

It is important to think about the meaning of missing values when clustering. We could treat them as having meaning or we could remove those people who have them. Neither option is ideal. What problems do you foresee if we recode or remove these values? Write your answers below:

Removing

- Removing NAs might require removing other values that are not NAs - removes information
- Reduction in sample size which may rule out certain types of analysis - here we go from 38 (decent for simple analyses) to 23 (too few for most things)
- Create a systematic bias against a potential group of people who like to produce NA values
  - The data might fail to capture an important relationship between this group and an outcome variable of interest

Recoding

- Risk misrepresenting the NA values - maybe everyone who gave an NA to something put a 6 on a scale of 5, but you recoded it to 3
- Choosing to recode may mask or amplify relationships depending on how it is done and what the data should have been.

We will remove people with missing values for this assignment, but keep in mind the issues that you have identified.

```r
# Create a data frame with only those people with no missing values.

# na.omit() "omits" all rows with missing values, also known as a "listwise deletion".
# EG - It runs down the list deleting rows as it goes.
K3 <- na.omit(K2)
```

Another pre-processing step used in K-means is to standardize the values so that they have the same range. We do this because we want to treat each week as equally important - if we do not standardise then the week with the largest range will have the greatest impact on which clusters are formed. We standardise the values by using the "scale()" command.

```r
# Scale the variables to reduce the impact of range on clustering.
# Note that scale() returns a matrix
K3 <- scale(K3)
```

Now we will run the K-means clustering algorithm we talked about in class. 1) The algorithm starts by randomly choosing some starting values 2) Associates all observations near to those values with them 3) Calculates the mean of those clusters of values 4) Selects the observation closest to the mean of the cluster 5) Re-associates all observations closest to this observation 6) Continues this process until the clusters are no longer changing

Notice that in this case we have 5 variables and in class we only had 2. It is impossible to vizualise this process with 5 variables.

Also, we need to choose the number of clusters we think are in the data. We will start with 2.

```
#Set seed
set.seed(4050)

#Run k means clustering analysis
fit <- kmeans(K3, 2)

#We have created an object called "fit" that contains all the details of our clustering including which

#We can access the list of clusters by typing "fit$cluster", the top row corresponds to the original or
fit$cluster
```

```
##  1  3  7 10 12 14 15 17 18 19 20 22 25 26 27 28 29 30 31 32 33 34 35
##  2  1  2  2  2  2  1  1  1  1  2  2  1  1  1  1  1  1  1  1  1  1  2
```

```
#We can also attach these clusters to the original dataframe by using the "data.frame" command to creat
K4 <- data.frame(K3, fit$cluster)

#Have a look at the K4 dataframe. Lets change the names of the variables to make it more convenient wit
names(K4) <- c("1", "2", "3", "4", "5", "cluster")
```

Now we need to visualize the clusters we have created. To do so we want to play with the structure of our data. What would be most useful would be if we could visualize average motivation by cluster, by week. To do this we will need to convert our data from wide to long format. Remember your old friends tidyr and dplyr!

First lets use tidyr to convert from wide to long format.

```
# DF cols should look like this: cluster, Week, Motivation Score
K5 <- K4 %>% gather(key = week,
                    value = motivation_score,
                    c(1, 2, 3, 4, 5)
                    )
```

Now lets use dplyr to average our motivation values by week and by cluster.

```
#Use summarise to get average motivation values by week and cluster
K6 <- K5 %>% group_by(cluster, week) %>% summarise("mean_motivation" = mean(motivation_score))
```

Now it's time to do some visualization:

https://www.cs.uic.edu/~wilkinson/TheGrammarOfGraphics/GOG.html

And you can see the range of available graphics in ggplot here:

http://ggplot2.tidyverse.org/reference/index.html

We are going to create a line plot similar to the one created in the school dropout paper we looked at in class (Bowers, 2010). It will have motivation on the Y-axis and weeks on the X-axis. To do this we will want our weeks variables to be treated as a number, but because it was created from a variable name it is currently being treated as a character variable. You can see this if you click on the arrow on the left of K6 in the Data pane. Week is designated by "chr". To convert it to numeric, we use the as.numeric command.

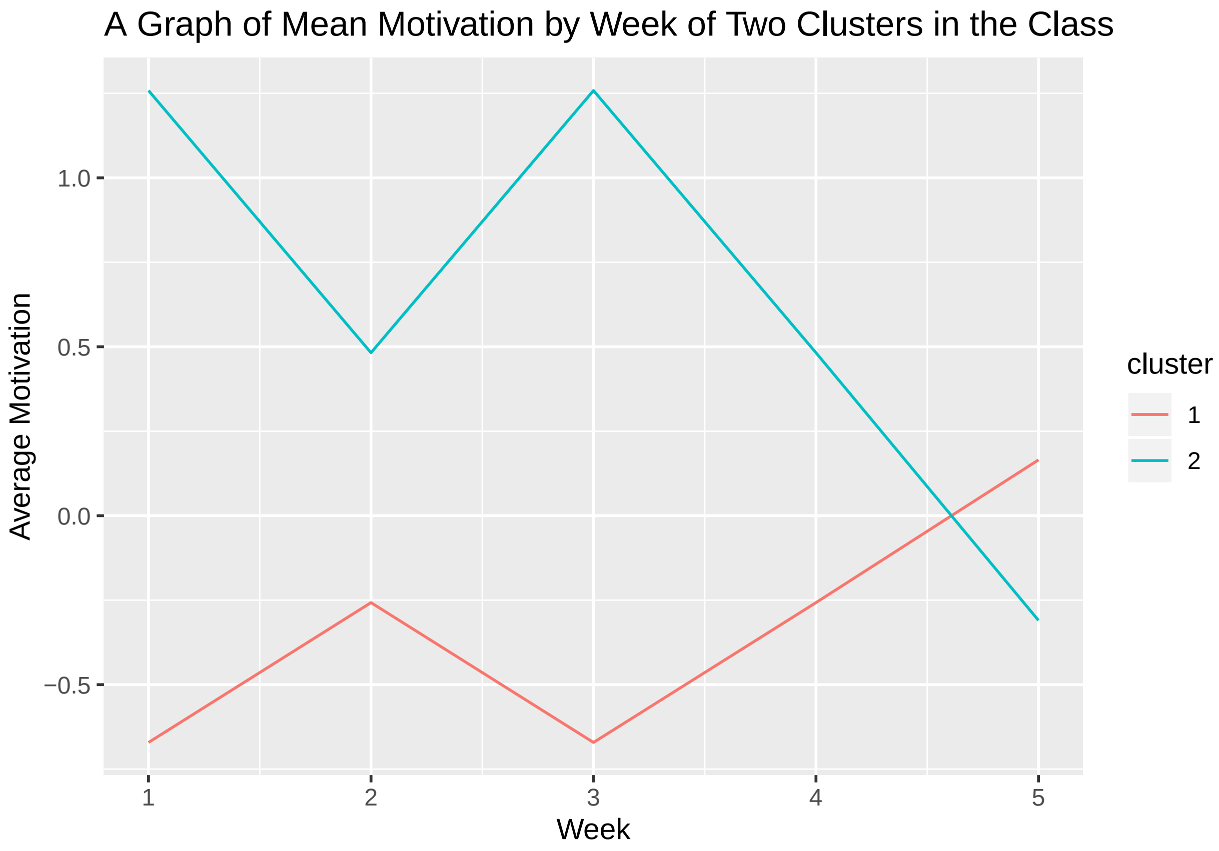
Likewise, since "cluster" is not numeric but rather a categorical label we want to convert it from an "integer" format to a "factor" format so that ggplot does not treat it as a number. We can do this with the as.factor() command.

```r
#Turn week into a numeric variable
K6$week <- as.numeric(K6$week)

#Turn cluster into a factor variable
K6$cluster <- as.factor(K6$cluster)
```

Now we can plot our line plot using the ggplot command, “ggplot()”.

- The first argument in a ggplot is the dataframe we are using: K6
- Next is what is called an aesthetic (aes), the aesthetic tells ggplot which variables to use and how to use them. Here we are using the variables “week” and “avg” on the x and y axes and we are going color these variables using the “cluster” variable
- Then we are going to tell ggplot which type of plot we want to use by specifiying a “geom()”, in this case a line plot: geom_line()
- Finally we are going to clean up our axes labels: xlab(“Week”) & ylab(“Average Motivation”)

```r
#Plot a line graph
ggplot(K6, aes(x = week, y = mean_motivation, colour = cluster)) +
  geom_line() +
  labs(x = "Week",
       y = "Average Motivation",
       title = "A Graph of Mean Motivation by Week of Two Clusters in the Class"
       )
```



What patterns do you see in the plot?

- One Cluster starts with high motivation that declines quickly after week 3
- Another cluster starts with low motivation, but it begins to increase after week 3

It would be useful to determine how many people are in each cluster. We can do this easily with dplyr.

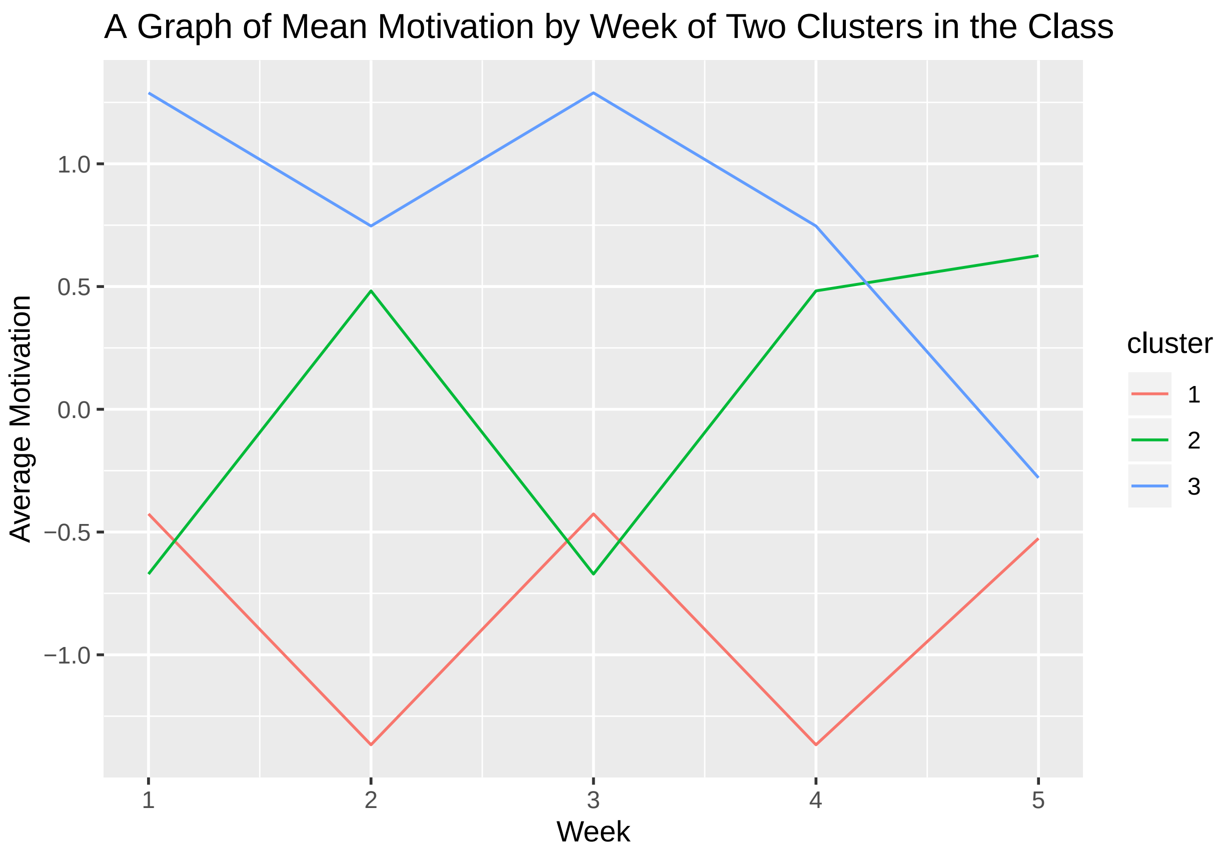
```
K7 <- K4 %>% group_by(cluster) %>% summarise("count" = n())
```

Look at the number of people in each cluster, now repeat this process for 3 rather than 2 clusters. Which cluster grouping do you think is more informative? Write your answer below:

```
#Set seed
set.seed(4050)

#Run k means clustering analysis
fit_3 <- kmeans(K3, 3)

#We have created an object called "fit" that contains all the details of our clustering including which

#We can access the list of clusters by typing "fit$cluster", the top row corresponds to the original or
fit_3$cluster
```

```
##  1  3  7 10 12 14 15 17 18 19 20 22 25 26 27 28 29 30 31 32 33 34 35
##  3  2  3  3  1  3  1  1  1  1  3  3  1  2  2  2  2  2  2  2  2  1  3
```

```
#We can also attach these clusters to the original dataframe by using the "data.frame" command to creat
K4_3 <- data.frame(K3, fit_3$cluster)

#Have a look at the K4 dataframe. Lets change the names of the variables to make it more convenient wit
names(K4_3) <- c("1", "2", "3", "4", "5", "cluster")

# Convert from wide to long format to get summary statistics by week and cluster.
#DF cols should look like this: cluster, Week, Motivation Score
K5_3 <- K4_3 %>% gather(key = week,
                        value = motivation_score,
                        c(1, 2, 3, 4, 5)
                        )

#Use summarise to get average motivation values by week and cluster
K6_3 <- K5_3 %>% group_by(cluster, week) %>% summarise("mean_motivation" = mean(motivation_score))

#Turn week into a numeric variable
K6_3$week <- as.numeric(K6_3$week)

#Turn cluster into a factor variable
K6_3$cluster <- as.factor(K6_3$cluster)

#Plot a line graph
ggplot(K6_3, aes(x = week, y = mean_motivation, colour = cluster)) +
  geom_line() +
  labs(x = "Week",
       y = "Average Motivation",
       title = "A Graph of Mean Motivation by Week of Two Clusters in the Class"
       )
```



```
# Find number of people in each cluster
K7 <- K4_3 %>% group_by(cluster) %>% summarise("count" = n())
```

I think that the using three clusters is more informative than using two clusters.

Using two clusters suggests a single trend for initially high motivation performing students, and a single trend for initally low motivation students.

With three clusters, we see that the trend for initally low motivation students is in fact divided. Some of these students maintain low motivation throughout, while others increase their motivation across the weeks. They eventually achieve around the same motivation levels, or higher, as the initally high performing students.

##Part II

Using the data collected for Assignment 2 (which classes students were in), cluster the students, then redraw the graph of the class but color the students according the cluster they are in.

```
#Import igraph
library(igraph)
```

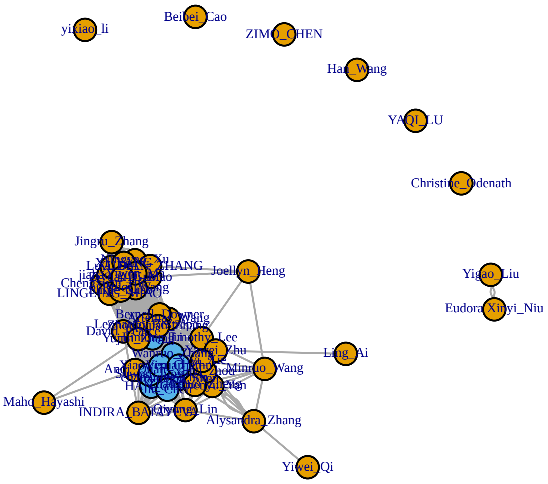
```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
## The following objects are masked from 'package:purrr':
##
##     compose, simplify

## The following object is masked from 'package:tidyr':
##
##     crossing

## The following object is masked from 'package:tibble':
##
##     as_data_frame

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
#Load the data
load("assn_2.RData")

#Assign rownames and get rid of ID column
rownames(person_class_DF) <- person_class_DF$id
```

```
## Warning: Setting row names on a tibble is deprecated.
```

```
person_class_DF_2 <- person_class_DF %>% dplyr::select(-id)

#Set seed
set.seed(4050)

#Perform k modes clustering with 2 clusters
class_fit <- kmodes(person_class_DF_2, 2)


#Attach cluster info to the original dataframe
plot.igraph(person_person_graph_data,
     layout = layout.fruchterman.reingold,
     vertex.size = 10,
     vertex.color = class_fit$cluster,
     vertex.label.cex = 0.4
     )
```



##Part III

In class activity 6 you clustered students in the class by the answers to a questionaire. Create a visualization that shows the overlap between these clusters and the clusters generated in part II.

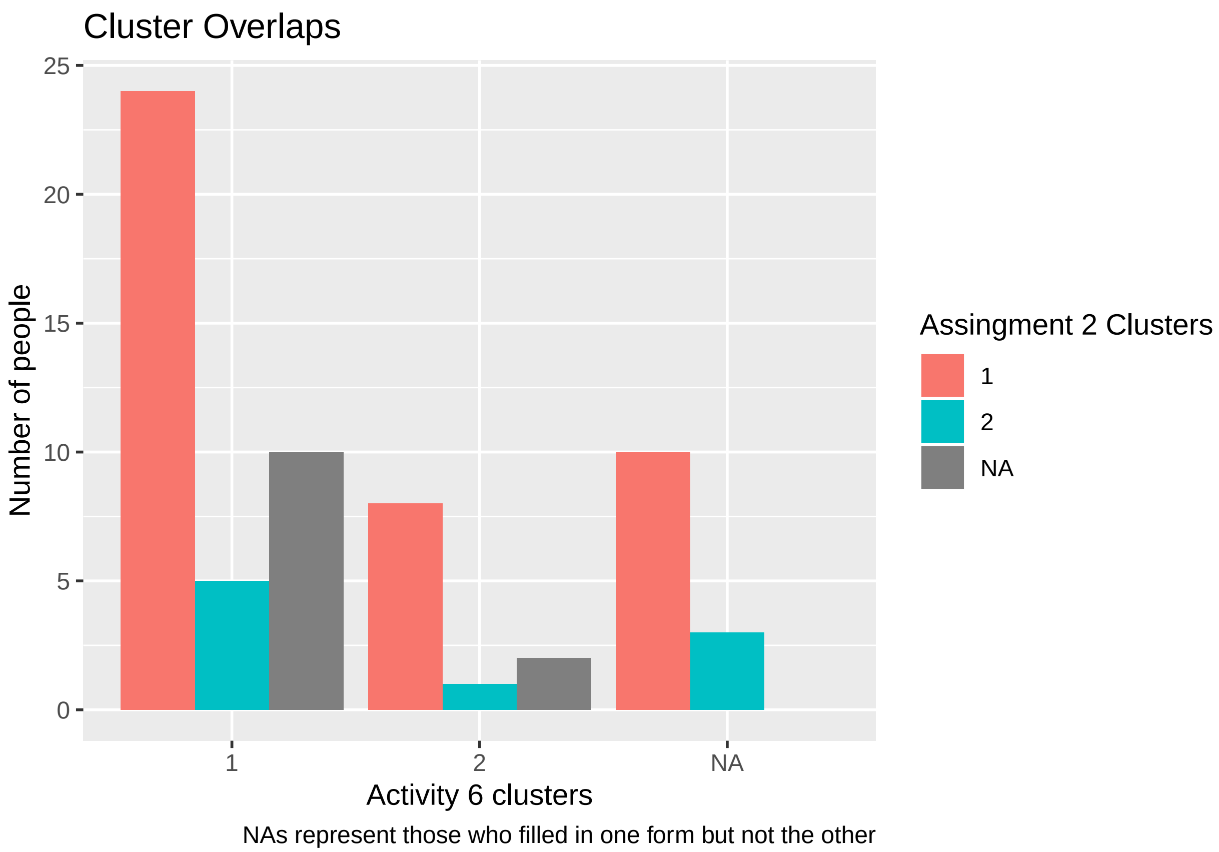
```
##Wrangling Assignment 2 Data
#Get all the names to upper case
person_class_DF$id <- str_to_upper(person_class_DF$id) %>% str_remove_all("[[:space:]]")

assn_2_new_DF <- data.frame(person_class_DF,
                            class_fit$cluster) %>% rename("assn_3_cluster_assn2" = class_fit.cluster)
```

```
##Wrangling Class Activity 6 Data

#Load Class Activity 6 Data
load("class_act_6.RData")

#Get all the names to upper case
class_act_6_names <- str_to_upper(class_act_6_names) %>% str_remove_all("[[:space:]]")

#Computing K means clustering with 2 clusters
fit_act_6 <- kmeans(class_act_6_scaled_data, 2)

#Attach these clusters to the original dataframe by using the "data.frame" command.
#Bring in the names as well
act_6_new_df <- data.frame(class_act_6_names,
                           class_act_6_df,
                           fit_act_6$cluster) %>%
                  rename("id" = class_act_6_names,
                         "act_6_cluster" = cluster,
                         "assn_3_cluster_act6" = fit_act_6.cluster
                         )
```

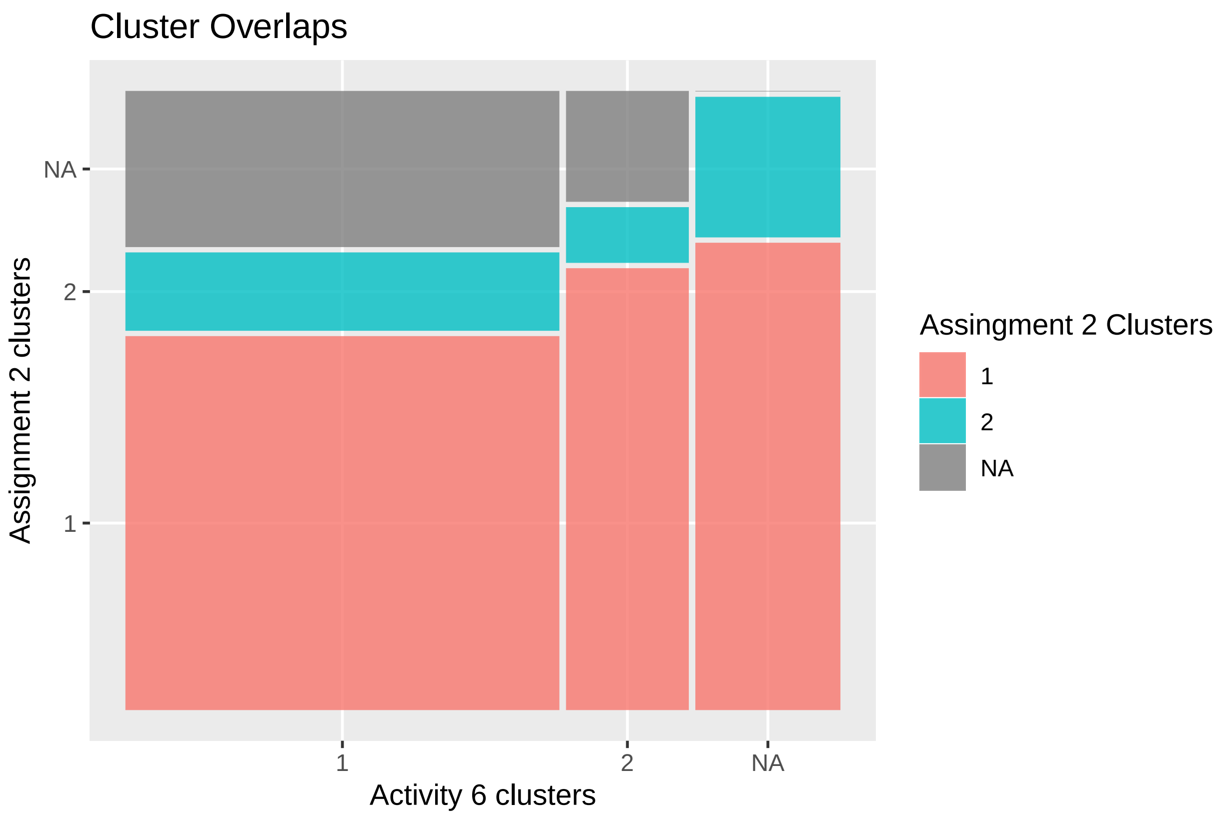
```
# Merging DFs, then coerce to factor with NA as a level
# NAs included as a factor level:
# They represent those who filled in one form but not the other
assn_3_combined_df <- full_join(act_6_new_df, assn_2_new_DF, by = "id") %>%
                        mutate(act6_clusters = factor(assn_3_cluster_act6, exclude = NULL),
                               assn2_clusters = factor(assn_3_cluster_assn2, exclude = NULL)
                               ) %>%
                        select(id, act6_clusters, assn2_clusters)
```

```
## Warning: Column `id` joining factor and character vector, coercing into
## character vector
```

```
#Bar Plot. The bar plot will tell us how many people in class act 6 cluster 1 are in assn 2 cluster 1,
# class act 6 cluster 1 x assn 2 cluster 2, and so on...
ggplot(assn_3_combined_df, aes(x = act6_clusters, fill = assn2_clusters)) +
  geom_bar(position = position_dodge(preserve = "single")) +
  labs(x = "Activity 6 clusters",
       y = "Number of people",
       title = "Cluster Overlaps",
       fill = "Assingment 2 Clusters",
       caption = "NAs represent those who filled in one form but not the other")
```



```
assn_3_summary <- assn_3_combined_df %>% group_by(act6_clusters, assn2_clusters) %>%
                      summarise("Freq" = n())

#Mosaic plot.
ggplot(assn_3_summary) +
  geom_mosaic(aes(weight = Freq, x = product(act6_clusters), fill = assn2_clusters)) +
  labs(x = "Activity 6 clusters",
       y = "Assignment 2 clusters",
       title = "Cluster Overlaps",
       fill = "Assignment 2 Clusters",
       caption = "NAs represent those who filled in one form but not the other")
```



NAs represent those who filled in one form but not the other

Please render your code as an .html file using knitr and Pull Resquest both your .Rmd file and .html files to the Assignment 3 repository.