Assignment 5

Due at 11:59pm on November 25.

You may work in pairs or individually for this assignment. Make sure you join a group in Canvas if you are working in pairs. Turn in this assignment as an HTML or PDF file to ELMS. Make sure to include the R Markdown or Quarto file that was used to generate it. Include the GitHub link for the repository containing these files.

```
library(censusapi)
library(tidyverse)
library(magrittr)
library(factoextra)
```

Exploring ACS Data

In this notebook, we use the Census API to gather data from the American Community Survey (ACS). This requires an access key, which can be obtained here:

https://api.census.gov/data/key_signup.html

```
acs_il_c <- getCensus(name = "acs/acs5",
                      vintage = 2016,
                      vars = c("NAME",
                               "B01003_001E",
                               "B19013_001E",
                               "B19301_001E"),
                      region = "county:*",
                      regionin = "state:17",
                      key = cs_key) %>%
             rename(pop = B01003_001E,
                    hh_income = B19013_001E,
                    income = B19301_001E)
head(acs_il_c)
```

Pull map data for Illinois into a data frame.

```
il_map <- map_data("county", region = "illinois")
head(il_map)
```

Join the ACS data with the map data. Not that `il_map` has a column `subregion` which includes county names. We need a corresponding variable in the ACS data to join both data sets. This needs some transformations, among which the function `tolower()` might be useful. Call the joined data `acs_map`.

After you do this, plot a map of Illinois with Counties colored by per capita income.

```
ggplot(acs_map) +
  geom_polygon(aes(x = long,
                   y = lat,
                   group = group,
                   fill = income))
```

Hierarchical Clustering

We want to find clusters of counties that are similar in their population, average household income and per capita income. First, clean the data so that you have the appropriate variables to use for clustering. Next, create the distance matrix of the cleaned data. This distance matrix can be used to cluster counties, e.g. using the ward method.

Plot the dendrogram to find a reasonable number of clusters. Draw boxes around the clusters of your cluster solution.

Visualize the county clusters on a map. For this task, create a new `acs_map` object that now also includes cluster membership as a new column. This column should be called `cluster`.

Census Tracts

For the next section we need ACS data on a census tract level. We use the same variables as before.

```
acs_il_t <- getCensus(name = "acs/acs5",
                      vintage = 2016,
                      vars = c("NAME",
                               "B01003_001E",
                               "B19013_001E",
                               "B19301_001E"),
```

```
                    region = "tract:*",
                    regionin = "state:17",
                    key = cs_key) %<>%
            mutate_all(funs(ifelse(.==-666666666, NA, .))) %>%
            rename(pop = B01003_001E,
                   hh_income = B19013_001E,
                   income = B19301_001E)
head(acs_il_t)
```

k-Means

As before, clean our data for clustering census tracts based on population, average household income and per capita income.

Since we want to use K Means in this section, we start by determining the optimal number of K that results in Clusters with low within but high between variation. Plot within cluster sums of squares for a range of K (e.g. up to 20).

Run `kmeans()` for the optimal number of clusters based on the plot above.

Find the mean population, household income and per capita income grouped by clusters. In addition, display the most frequent county that can be observed within each cluster.

As you might have seen earlier, it's not always clear which number of clusters is the optimal choice. To automate K Means clustering, program a function based on `kmeans()` that takes K as an argument. You can fix the other arguments, e.g. such that a specific dataset is always used when calling the function.

We want to utilize this function to iterate over multiple Ks (e.g., K = 2, ..., 10) and – each time – add the resulting cluster membership as a new variable to our (cleaned) original data frame (`acs_il_t`). There are multiple solutions for this task, e.g. think about the `apply` family or `for` loops.

Finally, display the first rows of the updated data set (with multiple cluster columns).