Assignment 5

Due at 11:59pm on November 25.

Jianing Zou

Github link: https://github.com/lainhhhhhh/SURV727

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)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.1     v stringr   1.5.2
v ggplot2   4.0.0     v tibble    3.3.0
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.1.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```
library(magrittr)
```

```
Attaching package: 'magrittr'

The following object is masked from 'package:purrr':

    set_names
```

```
The following object is masked from 'package:tidyr':

    extract
```

```
library(factoextra)
```

```
Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(stringr)
library(tidyverse)
library(purrr)
```

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

```
cs_key <- "c1323df5b7eceba665c7f5f1136ce2f6ed68bb0f"

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)
```

```
  state county                         NAME    pop hh_income income
1    17    067   Hancock County, Illinois  18633     50077  25647
2    17    063    Grundy County, Illinois  50338     67162  30232
3    17    091  Kankakee County, Illinois 111493     54697  25111
4    17    043    DuPage County, Illinois 930514     81521  40547
5    17    003 Alexander County, Illinois   7051     29071  16067
6    17    129    Menard County, Illinois  12576     60420  31323
```

Pull map data for Illinois into a data frame.

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

```
       long      lat group order   region subregion
1 -91.49563 40.21018     1     1 illinois     adams
2 -90.91121 40.19299     1     2 illinois     adams
3 -90.91121 40.19299     1     3 illinois     adams
4 -90.91121 40.10704     1     4 illinois     adams
5 -90.91121 39.83775     1     5 illinois     adams
6 -90.91694 39.75754     1     6 illinois     adams
```

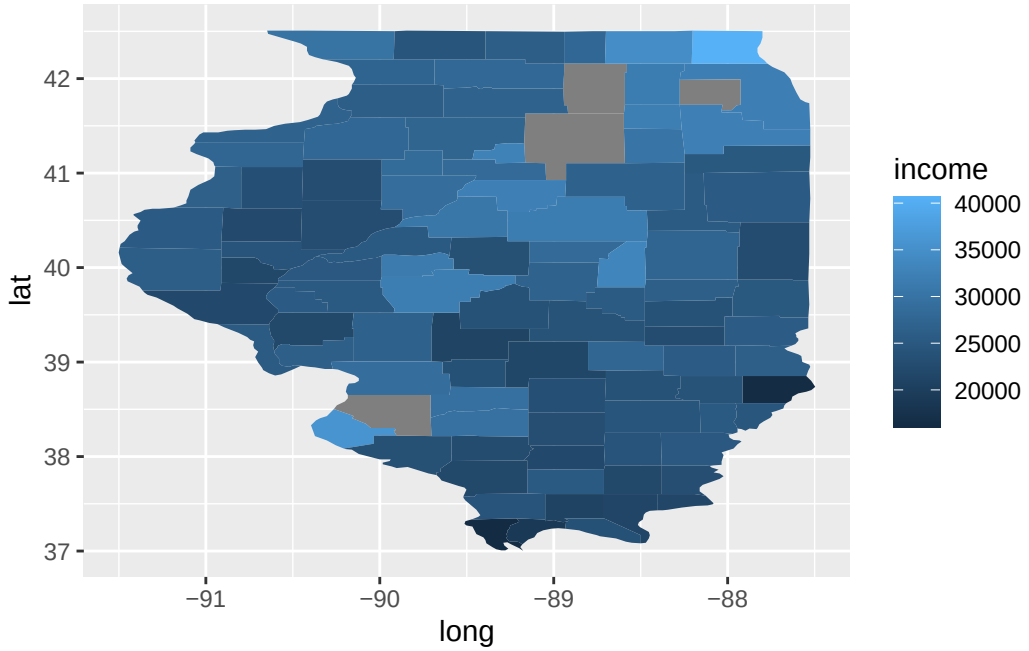
Join the ACS data with the map data. Not that `il_map` has a column `subregion` which includes county names. We need a corresponding variable in the ACS data to join both data sets. This needs some transformations, among which the function `tolower()` might be useful. Call the joined data `acs_map`.

```
acs_il_clean <- acs_il_c %>%
  mutate(
    subregion = NAME %>%
      str_replace(" County, Illinois", "") %>%
      tolower()
  )


acs_map <- il_map %>%
  left_join(acs_il_clean, by = "subregion")
```

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`.

```
acs_c <- acs_il_clean %>%
  select(subregion,
         pop,
         hh_income,
         income)

head(acs_c)
```

```
  subregion    pop hh_income income
1   hancock  18633     50077  25647
```

```
2    grundy  50338     67162  30232
3  kankakee 111493     54697  25111
4    dupage 930514     81521  40547
5 alexander   7051     29071  16067
6    menard  12576     60420  31323
```

We use Euclidean distance to compute matrix distance here:

Euclidean Distance: $\|x_a - x_b\|_2 = \sqrt{\sum_{j=1}^{P}(x_{aj} - x_{bj})^2}$

```
acs_d <- dist(acs_c)
```

```
Warning in dist(acs_c): NAs introduced by coercion
```

Hierarchical clustering is implemented in `hclust()`. To demonstrate that Clustering is very sensitive to the parameters being used, we create three cluster objects based on three types of linkage methods.

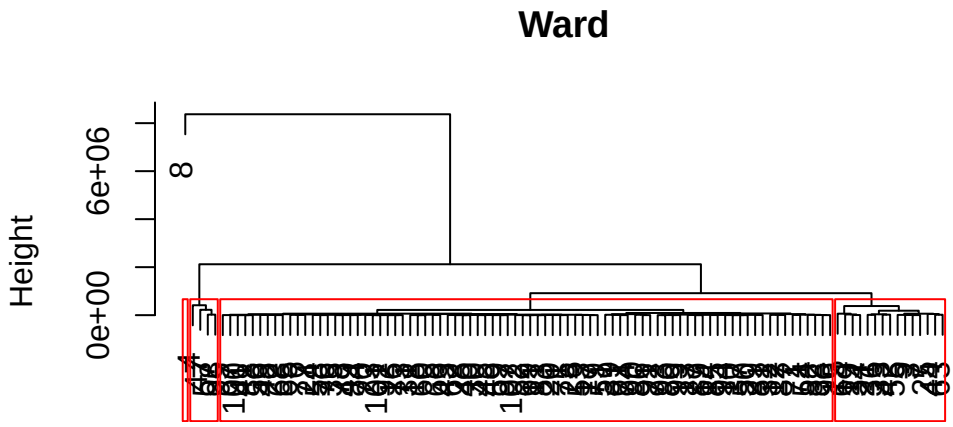
```
acs_ward <- hclust(acs_d, method = "ward.D2")
```

The dendograms of the cluster results show how observations are merged to create clusters. On this basis, we can pick the number of clusters we want to extract by stopping the fusion process at a certain point. `rect.hclust()` can be used to highlight a specific cluster solution.

From the plot, we can see that 4 is the reasonable cluster number in our case.

```
plot(acs_ward, main = "Ward", xlab = "", sub = "")

rect.hclust(acs_ward,
            k = 4,
            border = "red")
```



We want to create 4 clusters based on the object.

```
cutree(acs_ward, 4)
```

```
  [1] 1 1 2 3 1 1 2 4 2 1 1 1 1 1 2 1 1 1 2 1 2 1 1 1 2 1 1 1 1 1 1 2 2 1 2 1
 [38] 1 1 1 1 2 1 2 1 1 3 1 1 1 1 1 1 1 2 1 3 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1
 [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1
```

One way to make sense of a cluster solution is to simply compute the mean of the variables we used to generate the clusters for each cluster.

The dominant county of each cluster is adams for county 1, champaign for cluster 2, dupage for cluster 3, cook for cluster 4.

```
acs_c_new <- acs_c %>%
  mutate(cluster = cutree(acs_ward, 4))


acs_c_new %>%
  group_by(cluster) %>%
  summarise(mean(pop), mean(hh_income), mean(income), subregion =  names(table(subregion))[wh
```

```
# A tibble: 4 x 5
  cluster `mean(pop)` `mean(hh_income)` `mean(income)` subregion
    <int>       <dbl>             <dbl>          <dbl> <chr>
1       1      24946.            48762.         25137. adams
2       2     182208.            57375.         28712. champaign
3       3     711349.            77629          36322. dupage
4       4    5227575             56902          32179  cook
```

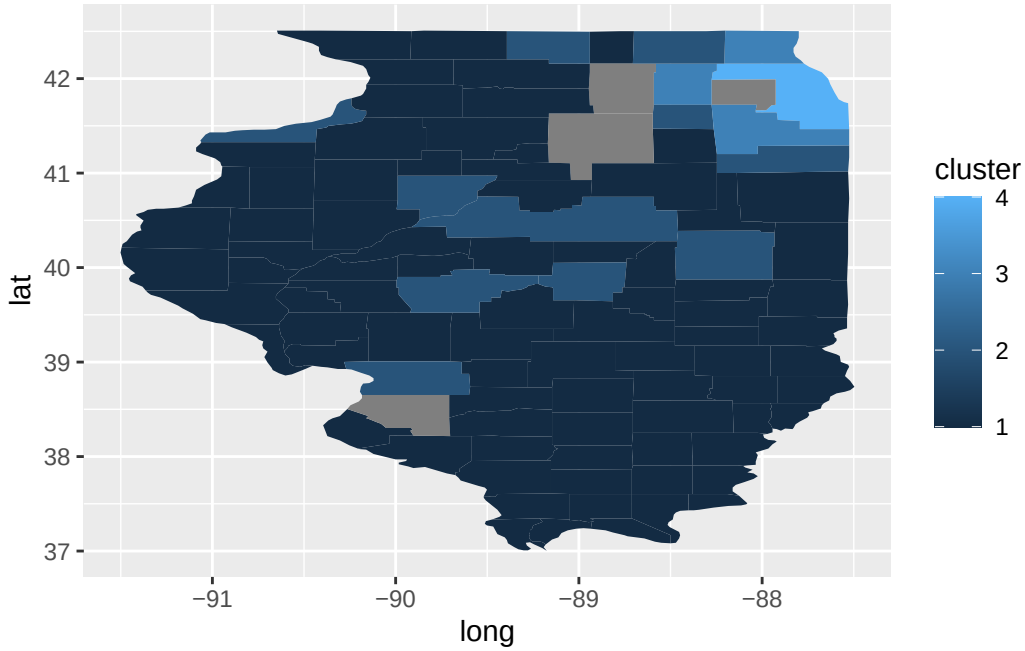
```
acs_map_new <- il_map %>%
  full_join(acs_c_new , by = "subregion")

head(acs_map_new)
```

```
       long      lat group order   region subregion   pop hh_income income
1 -91.49563 40.21018     1     1 illinois     adams 66949     48065  26053
2 -90.91121 40.19299     1     2 illinois     adams 66949     48065  26053
3 -90.91121 40.19299     1     3 illinois     adams 66949     48065  26053
4 -90.91121 40.10704     1     4 illinois     adams 66949     48065  26053
5 -90.91121 39.83775     1     5 illinois     adams 66949     48065  26053
6 -90.91694 39.75754     1     6 illinois     adams 66949     48065  26053
  cluster
1       1
2       1
3       1
4       1
5       1
6       1
```

The visualization of map here shows the clusters (Color).

```
ggplot(acs_map_new) +
  geom_polygon(aes(x = long,
                   y = lat,
                   group = group,
                   fill = 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(
    across(
      .fns = ~ ifelse(. == -666666666, NA, .)
    )
  ) %>%
  rename(
```

```
    pop        = B01003_001E,
    hh_income  = B19013_001E,
    income     = B19301_001E
  )
```

```
Warning: There was 1 warning in `mutate()`.
i In argument: `across(.fns = ~ifelse(. == -666666666, NA, .))`.
Caused by warning:
! Using `across()` without supplying `.cols` was deprecated in dplyr 1.1.0.
i Please supply `.cols` instead.
```

```
head(acs_il_t)
```

```
  state county  tract                                          NAME  pop
1    17    031 806002 Census Tract 8060.02, Cook County, Illinois 7304
2    17    031 806003 Census Tract 8060.03, Cook County, Illinois 7577
3    17    031 806400    Census Tract 8064, Cook County, Illinois 2684
4    17    031 806501 Census Tract 8065.01, Cook County, Illinois 2590
5    17    031 750600    Census Tract 7506, Cook County, Illinois 3594
6    17    031 310200    Census Tract 3102, Cook County, Illinois 1521
  hh_income income
1     56975  23750
2     53769  25016
3     62750  30154
4     53583  20282
5     40125  18347
6     63250  31403
```

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).

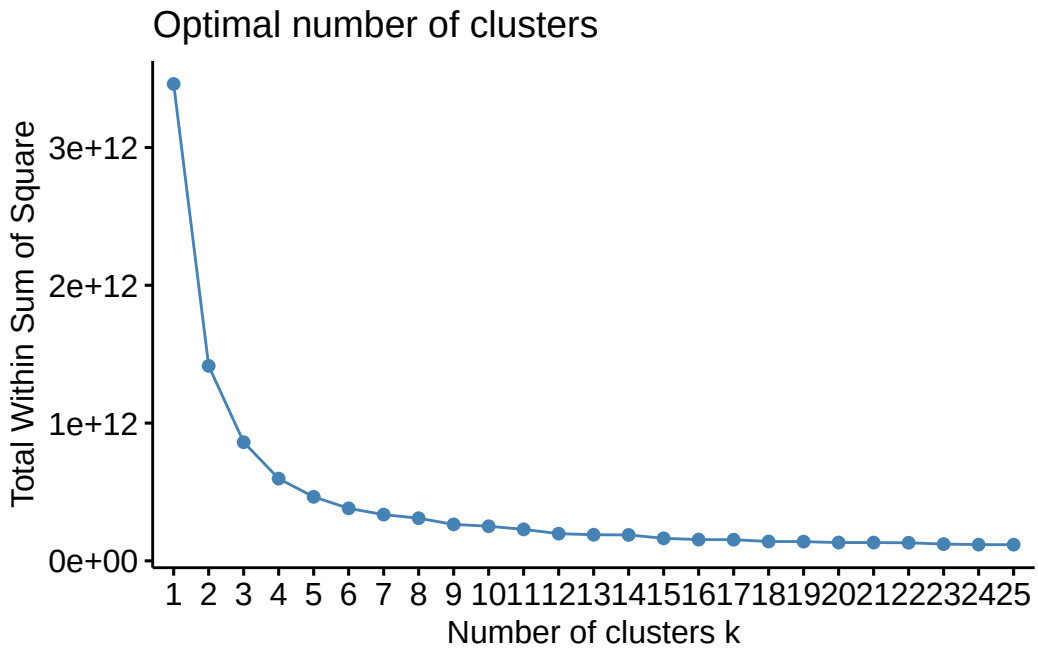
```
acs_k <- acs_il_t %>%
  select(pop, hh_income, income, county, tract, NAME) %>%  mutate(
    subregion = NAME %>%
      str_extract(",\\s*([A-Za-z]+) County") %>%
      str_remove_all("[, ]") %>%
      str_remove("County") %>%
      tolower(),
     tract = str_remove(NAME, ",.*$")
  )%>%
  drop_na()


acs_k_mat <- acs_k %>%
  select(pop, hh_income, income)  %>%
  drop_na()
```

Based on the "elbow" criterion, we may want to choose k = 3 as the optimal number of clusters in this case. Therefore, we run `kmeans()` again and also plot the new cluster solution to inspect the new result.

```
fviz_nbclust(acs_k_mat,
             kmeans,
             method = "wss",
             k.max = 25)
```

```
Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations
```




```
km <- kmeans(acs_k_mat, 6, nstart = 20)
km
```

```
K-means clustering with 6 clusters of sizes 223, 775, 913, 443, 73, 571

Cluster means:
       pop hh_income   income
1 4504.740 112344.15 60414.62
2 4336.809  64648.81 31588.68
3 4051.014  46742.24 23473.16
4 5125.011  87822.97 41227.10
5 4126.795 163369.92 78493.00
6 3072.538  27951.76 15708.86

Clustering vector:
   [1] 3 3 2 3 3 2 6 2 6 6 3 3 5 5 5 5 1 3 1 4 2 4 2 6 2 2 2 2 6 2 6 2 3 6 3 4 2
  [38] 3 3 3 4 2 3 4 2 4 2 2 4 4 3 3 4 2 2 2 2 6 3 6 6 6 6 6 4 4 6 6 6 3 3 2 2 6
  [75] 3 3 3 6 6 2 6 6 4 4 3 4 4 2 2 4 1 1 1 4 1 1 2 3 3 2 3 1 5 5 1 4 1 5 1 5 1
 [112] 5 2 4 2 4 2 4 4 3 4 1 5 1 4 4 3 2 2 4 6 6 6 3 6 3 6 3 4 3 6 6 3 3 1 6 3 3
 [149] 2 3 2 3 3 3 3 3 6 3 2 6 6 3 6 2 6 3 3 6 6 1 2 2 6 6 6 6 3 3 3 3 2 2 3 6 3
 [186] 3 3 3 3 3 3 2 3 3 3 3 2 6 6 2 2 2 2 3 3 2 3 2 4 6 2 2 2 2 3 2 3 3 6 6 6 6
 [223] 1 3 4 2 6 2 2 3 3 3 6 6 6 3 6 3 6 3 2 2 3 2 6 2 3 2 2 2 3 2 3 3 3 2 3 3 6
 [260] 6 3 2 2 2 4 4 3 1 2 2 1 2 1 2 2 1 4 4 2 4 1 5 2 2 2 2 4 4 4 4 1 5 4 1 2 2
```

```
 [297] 3 3 3 2 3 3 3 3 3 3 3 3 3 2 2 2 2 3 6 3 2 3 3 4 4 2 2 4 1 2 1 4 2 4 2 2 2
 [334] 4 3 3 2 2 4 4 4 5 5 5 1 6 6 2 4 4 2 2 4 4 3 6 6 4 2 1 2 3 3 2 2 3 3 3 3 3
 [371] 6 3 3 2 3 3 6 2 6 3 3 2 2 3 2 3 2 6 2 3 6 6 1 3 4 1 2 6 3 2 6 6 2 2 4 4 4
 [408] 4 6 6 6 1 6 4 1 1 1 4 4 3 2 1 2 6 2 6 4 1 3 3 3 6 5 4 5 5 5 1 1 2 6 3 3 6
 [445] 2 1 3 2 3 2 2 2 3 3 2 3 3 3 3 5 2 3 2 2 2 2 4 6 3 3 3 3 4 2 4 4 4 1 1 1 4
 [482] 4 4 4 3 4 2 1 2 4 1 4 2 3 2 2 4 4 2 6 6 4 4 1 1 3 1 4 1 3 6 6 3 3 3 6 6 6
 [519] 6 6 1 1 3 5 1 1 1 4 1 1 6 2 3 3 3 3 3 2 2 3 2 4 3 4 4 1 4 4 4 3 3 2 1 1 1
 [556] 5 1 1 5 4 1 1 4 5 1 2 2 2 3 6 3 6 6 1 4 3 1 4 6 4 2 3 3 2 2 6 6 2 2 4 3 2
 [593] 2 6 6 3 3 3 2 6 3 3 6 6 6 2 3 6 4 6 6 6 6 3 6 3 3 3 6 3 2 6 6 6 6 2 1 3 6
 [630] 6 3 2 3 1 3 1 6 2 3 3 2 6 3 3 3 2 3 3 2 3 6 6 3 4 6 2 2 6 3 3 3 1 4 4 6 1
 [667] 1 4 2 4 5 4 1 4 4 2 2 2 2 4 3 4 4 4 3 1 4 1 4 1 1 2 3 3 2 2 2 1 3 1 6 2 2
 [704] 3 2 2 2 2 2 2 2 2 3 2 2 3 2 3 2 3 2 3 3 2 3 2 5 3 3 3 3 3 6 3 3 2 6 3 3 6
 [741] 6 3 6 3 4 3 6 3 3 6 4 4 1 1 1 4 1 1 1 4 4 2 1 6 1 4 3 3 2 2 6 3 3 2 3 6 4
 [778] 5 6 3 3 6 6 3 2 6 2 2 3 2 3 2 2 3 3 2 2 2 2 3 3 3 2 3 3 3 3 2 3 3 3 6 6 6
 [815] 2 2 3 2 2 4 4 4 4 3 2 1 3 3 2 1 1 1 4 2 2 2 1 2 3 3 2 1 1 1 2 4 2 2 4 4 1
 [852] 4 1 2 2 3 4 4 2 2 1 4 3 3 6 3 2 4 2 2 3 3 2 2 3 3 6 3 3 3 3 6 3 6 2 3 4 3
 [889] 4 3 3 6 6 3 6 3 6 3 3 3 3 3 2 2 2 2 3 3 4 6 3 3 3 3 2 2 4 4 4 4 4 4 4 1 6
 [926] 6 3 2 2 3 3 2 3 2 3 3 3 3 2 2 2 4 4 2 4 4 4 2 4 4 6 3 6 6 6 6 6 4 2 6 1 1
 [963] 3 6 2 6 6 6 6 3 6 6 6 6 1 2 2 3 5 1 6 3 3 5 5 1 1 2 4 3 2 2 3 1 5 2 4 2 2
[1000] 3 3 6 3 3 3 3 6 6 3 3 6 2 3 3 3 3 3 6 3 6 6 6 6 2 4 3 4 4 4 3 3 3 3 3 2 3
[1037] 2 2 6 1 4 4 4 2 4 2 2 2 3 2 3 3 3 2 4 5 3 2 2 4 2 3 4 3 2 2 3 2 6 6 3 6 3
[1074] 2 3 6 6 2 3 6 6 3 2 6 6 6 2 4 1 4 3 3 3 3 4 6 6 6 6 6 3 3 6 6 6 6 3 6 3 3
[1111] 6 6 6 6 6 6 2 1 4 4 6 2 6 3 6 3 1 2 6 3 3 6 6 4 4 2 3 3 4 4 2 2 1 5 4 2 4
[1148] 1 1 4 1 4 2 4 4 1 4 2 1 4 1 1 3 3 3 3 4 4 2 2 2 5 1 1 4 3 2 2 2 4 4 2 6 5
[1185] 5 3 4 3 2 5 2 6 4 2 2 5 3 3 4 2 5 1 2 2 2 3 2 1 3 6 2 3 3 3 4 4 2 4 4 2 3
[1222] 6 2 3 6 6 4 4 6 6 6 2 6 3 6 6 6 3 3 3 3 6 6 3 2 6 6 6 6 6 6 2 3 6 6 6 2 3
[1259] 3 2 1 1 2 4 1 4 1 3 6 3 6 6 3 2 3 3 2 3 3 2 4 2 2 2 4 2 2 4 5 3 3 3 2 3 3
[1296] 2 3 6 2 6 6 3 6 6 6 6 2 3 2 3 2 3 3 3 3 3 2 3 3 2 1 2 2 2 3 4 3 3 2 6 2 3
[1333] 2 3 4 3 2 6 6 3 6 3 4 6 2 2 2 3 3 3 3 6 2 3 3 3 6 3 2 6 3 3 3 2 3 3 6 3 2
[1370] 2 3 3 2 4 1 2 2 3 4 3 3 6 6 2 2 3 6 1 2 3 6 3 2 6 2 6 2 4 3 2 3 3 2 1 1 4
[1407] 3 1 1 5 1 2 3 4 2 1 4 1 1 5 4 2 1 2 4 2 3 3 2 3 2 2 2 3 3 3 3 2 3 3 3 3 3
[1444] 4 2 3 6 6 6 6 2 6 2 3 4 3 3 3 3 3 3 2 3 6 6 6 6 6 2 3 3 3 6 4 4 4 4 4 1 1
[1481] 3 3 4 3 5 2 6 3 3 2 4 3 3 2 6 4 3 3 3 6 4 3 5 2 5 5 4 2 2 4 2 4 1 4 6 3 3
[1518] 6 4 3 2 2 3 3 3 2 2 6 2 2 6 6 3 2 3 2 4 4 3 3 1 4 1 1 2 3 3 2 3 4 4 2 3 2
[1555] 2 4 2 6 6 6 3 3 3 6 2 4 3 2 6 6 2 3 6 6 6 3 6 3 3 2 6 6 3 6 3 6 6 2 1 4 2
[1592] 3 3 1 3 6 1 2 2 2 2 3 4 3 2 4 1 6 2 4 2 4 2 2 3 3 2 3 3 3 3 6 6 3 6 6 1 1
[1629] 1 2 2 4 4 6 6 2 3 3 3 3 3 6 6 6 3 2 2 3 4 4 3 3 3 2 3 2 3 6 6 6 3 3 3 3 3
[1666] 6 6 6 6 3 2 3 3 2 4 2 3 6 3 2 6 6 3 6 6 6 6 6 6 6 3 2 3 4 2 2 2 6 2 1 6 4
[1703] 4 4 2 2 4 4 2 2 3 3 3 5 2 3 6 2 2 2 3 3 2 3 2 4 3 3 2 3 3 3 3 3 2 3 2 3 6
[1740] 6 3 2 3 3 2 2 2 2 3 2 2 1 2 4 4 4 1 3 3 3 6 6 6 4 4 2 2 5 3 6 6 3 6 3 6 6
[1777] 6 6 3 3 3 2 1 4 4 2 6 3 6 4 3 2 6 6 6 6 2 3 2 2 3 3 4 4 3 6 3 4 2 6 2 3 3
[1814] 6 6 4 1 6 6 3 6 4 3 3 6 4 6 6 6 4 2 2 6 3 2 2 3 2 4 4 3 5 1 3 6 6 6 3 3 6
[1851] 6 3 6 6 6 3 4 4 2 6 6 6 6 5 2 1 3 2 2 3 6 2 3 2 2 3 6 2 1 1 2 2 1 4 4 3 2
```

```
[1888] 3 3 3 2 2 3 3 3 6 3 6 2 3 3 2 3 3 2 3 2 6 3 3 3 6 3 3 3 3 2 3 3 3 3 6 3 3
[1925] 3 3 6 3 4 3 2 2 3 2 2 3 4 3 2 3 3 2 2 3 2 3 2 2 2 6 2 2 6 6 2 2 6 3 2 3 2
[1962] 2 6 6 2 2 3 2 2 6 6 2 3 6 2 3 6 2 2 4 6 2 6 3 3 2 3 4 6 3 6 3 6 4 4 3 6 2
[1999] 4 2 2 3 3 4 4 2 4 4 4 2 4 4 3 3 3 4 1 2 2 6 3 6 6 3 6 3 3 3 2 2 3 4 2 5 3
[2036] 1 1 3 6 6 2 3 2 3 2 2 2 2 3 6 3 3 6 4 1 1 4 2 3 3 2 2 3 6 3 2 2 3 3 2 3 3
[2073] 3 6 3 6 3 3 4 6 3 6 3 3 3 6 3 2 2 4 2 6 3 4 2 6 6 6 6 6 2 6 3 6 3 2 3 3 2
[2110] 6 5 6 3 3 2 3 2 6 6 3 3 6 6 3 2 2 3 3 3 3 2 2 2 2 3 4 2 2 2 6 2 4 2 2 2 6
[2147] 3 3 3 2 4 3 2 4 3 2 2 3 3 2 2 1 3 6 6 6 6 6 1 2 4 6 3 3 3 2 2 2 5 2 2 2 2
[2184] 2 3 4 6 4 2 4 2 4 4 4 3 4 4 2 6 2 3 6 6 6 6 6 3 6 3 4 3 3 2 2 3 3 6 2 6 6
[2221] 6 3 4 4 2 4 5 1 2 4 1 4 1 4 1 1 6 3 3 3 3 3 3 6 3 6 3 3 2 2 3 2 3 2 2 2 4
[2258] 2 2 4 2 4 2 4 2 4 2 2 3 3 4 6 6 3 4 5 4 3 2 5 5 5 4 4 2 2 1 4 4 3 4 4 4 1
[2295] 1 3 3 3 3 3 6 6 6 2 3 2 3 3 3 6 6 2 2 2 2 1 1 4 1 3 2 2 4 1 3 2 3 6 4 3 2
[2332] 2 6 6 3 3 2 3 3 3 3 3 5 3 2 1 4 4 1 5 2 2 2 3 4 2 6 3 6 3 1 5 6 2 3 2 5 5
[2369] 3 5 4 1 3 2 2 6 3 3 3 4 4 1 1 4 4 4 2 2 2 3 2 3 2 2 3 2 2 6 3 1 1 4 1 4 4
[2406] 3 2 2 1 3 2 6 3 6 4 2 1 6 6 6 6 6 6 3 3 4 2 2 2 2 4 4 4 3 2 6 2 2 6 3 3 3
[2443] 2 4 3 3 3 3 3 3 2 2 3 2 3 6 2 3 3 2 3 3 3 2 3 3 6 6 2 2 2 2 2 6 6 3 3 4 3
[2480] 4 4 3 4 5 1 2 2 3 3 3 3 3 3 4 2 2 2 4 4 4 4 4 2 2 4 2 1 1 2 2 4 4 2 4 3 2
[2517] 4 3 2 3 3 2 6 6 2 3 3 2 3 2 3 4 3 3 3 4 2 4 2 2 6 2 4 4 4 2 2 2 3 2 3 2 6
[2554] 6 3 2 3 3 2 3 3 6 6 3 3 2 3 6 3 6 3 3 2 2 2 3 3 2 1 1 1 3 2 4 2 2 6 5 5 4
[2591] 2 4 4 4 4 2 3 2 4 2 2 2 4 1 4 3 2 4 4 4 4 2 3 6 1 4 2 3 4 6 3 2 4 2 4 3 4
[2628] 4 6 3 3 1 1 1 4 3 6 2 6 6 3 6 2 6 6 3 6 6 1 3 6 6 1 1 2 2 3 6 4 4 6 6 6 2
[2665] 3 3 2 3 6 3 6 6 6 2 3 1 2 4 2 2 4 2 2 2 3 4 2 2 4 4 1 4 1 1 4 4 2 2 3 4 2
[2702] 2 3 2 3 3 6 2 3 2 2 6 5 1 4 3 3 2 3 2 2 3 2 4 4 4 4 4 2 1 2 4 2 5 4 4 6 6
[2739] 6 6 6 6 6 6 6 6 6 6 6 6 6 4 3 3 6 3 4 4 4 2 3 6 2 6 4 2 2 3 3 3 3 2 3 6 5
[2776] 2 1 1 4 6 6 6 2 2 3 4 6 3 3 2 3 6 6 6 6 6 2 4 6 6 5 3 2 3 2 4 1 2 5 1 1 1
[2813] 1 1 1 5 2 3 2 3 3 6 3 3 3 3 3 3 3 3 6 3 2 2 2 2 4 2 4 3 2 4 6 3 3 3 3 2 3
[2850] 2 3 6 6 6 1 6 1 4 4 4 4 2 4 4 4 4 3 4 2 4 4 4 6 6 6 6 6 6 6 3 6 6 3 6 3 2
[2887] 2 4 6 6 2 3 6 3 6 2 3 2 4 6 6 6 6 6 3 3 4 3 3 3 4 2 2 6 6 6 3 3 3 2 2 4 3
[2924] 6 4 2 2 2 3 2 2 2 2 3 2 4 2 2 4 3 2 2 2 3 2 6 2 3 2 3 3 3 3 3 3 3 3 3 2 3
[2961] 4 2 2 3 2 3 3 3 3 2 3 4 4 2 2 3 3 2 3 3 6 3 6 6 3 6 4 2 1 3 3 4 2 2 3 6 3
[2998] 3

Within cluster sum of squares by cluster:
[1] 90958654151 67248402619 50394605535 68897394152 61733408117 41571932246
 (between_SS / total_SS =  89.0 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

For cluster 1, the most frequent tract is Census Tract 8422.

For cluster 2, the most frequent tract is Census Tract 103.

For cluster 3, the most frequent tract is Census Tract 101.

For cluster 4, the most frequent tract is Census Tract 105.

For cluster 5, the most frequent tract is Census Tract 3204.

For cluster 6, the most frequent tract is Census Tract 4.

```
acs_k$cluster <- km$cluster

summary_clusters <- acs_k %>%
  group_by(cluster) %>%
  summarise(
    mean_pop     = mean(pop, na.rm = TRUE),
    mean_hh_inc  = mean(hh_income, na.rm = TRUE),
    mean_income  = mean(income, na.rm = TRUE),
    tract = names(table(tract)[which.max(table(tract))]
  ))


summary_clusters
```

```
# A tibble: 6 x 5
  cluster mean_pop mean_hh_inc mean_income tract
    <int>    <dbl>       <dbl>       <dbl> <chr>
1       1    4505.     112344.      60415. Census Tract 8422
2       2    4337.      64649.      31589. Census Tract 103
3       3    4051.      46742.      23473. Census Tract 101
4       4    5125.      87823.      41227. Census Tract 105
5       5    4127.     163370.      78493  Census Tract 3204
6       6    3073.      27952.      15709. Census Tract 4
```

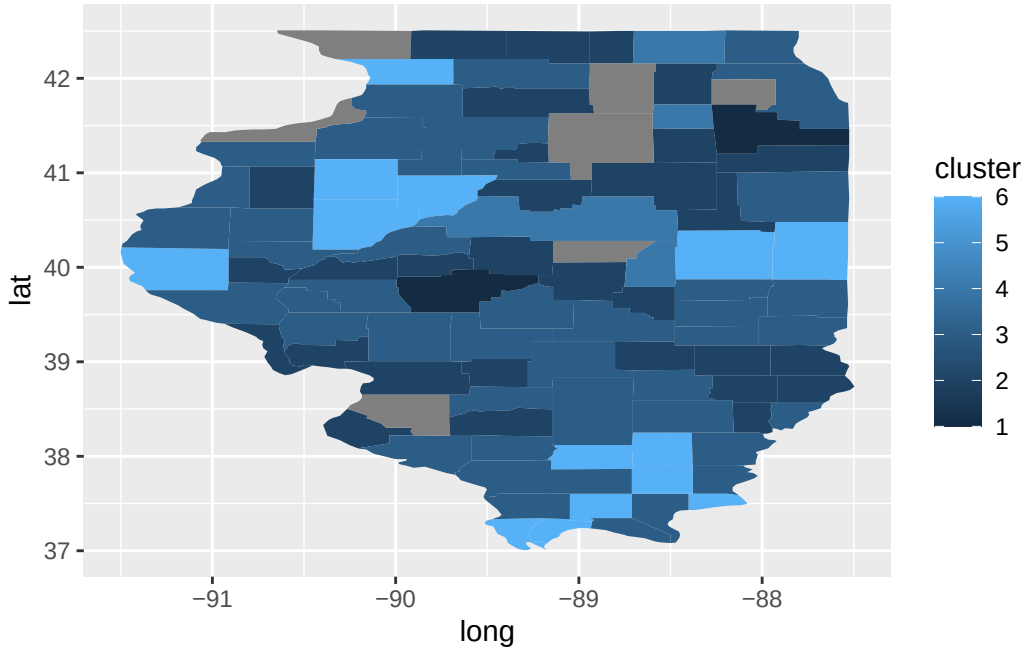
```
acs_map_k <- il_map %>%
  full_join(acs_k , by = "subregion")
```

```
Warning in full_join(., acs_k, by = "subregion"): Detected an unexpected many-to-many relati
i Row 1 of `x` matches multiple rows in `y`.
i Row 1559 of `y` matches multiple rows in `x`.
i If a many-to-many relationship is expected, set `relationship =
  "many-to-many"` to silence this warning.
```

```
head(acs_map_k)
```

```
       long      lat group order   region subregion  pop hh_income income
1 -91.49563 40.21018     1     1 illinois     adams 2700     26012  16130
2 -91.49563 40.21018     1     1 illinois     adams 2298     32313  14527
3 -91.49563 40.21018     1     1 illinois     adams 4323     50156  24763
4 -91.49563 40.21018     1     1 illinois     adams 3764     44324  29072
5 -91.49563 40.21018     1     1 illinois     adams 2671     40475  18335
6 -91.49563 40.21018     1     1 illinois     adams 3403     28819  18268
  county            tract                                    NAME cluster
1    001   Census Tract 8   Census Tract 8, Adams County, Illinois       6
2    001   Census Tract 5   Census Tract 5, Adams County, Illinois       6
3    001 Census Tract 101 Census Tract 101, Adams County, Illinois       3
4    001   Census Tract 6   Census Tract 6, Adams County, Illinois       3
5    001   Census Tract 9   Census Tract 9, Adams County, Illinois       3
6    001   Census Tract 4   Census Tract 4, Adams County, Illinois       6
```

```
ggplot(acs_map_k) +
  geom_polygon(aes(x = long,
                   y = lat,
                   group = group,
                   fill = cluster))
```



Function build:

There is the function that runs kmeans() for a given K and returns the clusters. We use that function in a loop for k = 2 to k = 9. Each time, we add a new cluster K column to the dataset.

```
kmeans_for_k <- function(K) {
  set.seed(123)  # for reproducibility
  km_3 <- kmeans(acs_k_mat, centers = K, nstart = 25)
  return(km_3$cluster)  # just the cluster membership
}

for (k in 2:10) {
  acs_k[[paste0("cluster_", k)]] <- kmeans_for_k(k)
}
```

```
cluster_cols <- map_dfc(2:10, ~ tibble(!!paste0("cluster_", .x) := kmeans_for_k(.x)))

acs_k_3 <- bind_cols(acs_k, cluster_cols)
```

```
New names:
* `cluster_2` -> `cluster_2...9`
* `cluster_3` -> `cluster_3...10`
* `cluster_4` -> `cluster_4...11`
* `cluster_5` -> `cluster_5...12`
* `cluster_6` -> `cluster_6...13`
* `cluster_7` -> `cluster_7...14`
* `cluster_8` -> `cluster_8...15`
* `cluster_9` -> `cluster_9...16`
* `cluster_10` -> `cluster_10...17`
* `cluster_2` -> `cluster_2...18`
* `cluster_3` -> `cluster_3...19`
* `cluster_4` -> `cluster_4...20`
* `cluster_5` -> `cluster_5...21`
* `cluster_6` -> `cluster_6...22`
* `cluster_7` -> `cluster_7...23`
* `cluster_8` -> `cluster_8...24`
* `cluster_9` -> `cluster_9...25`
* `cluster_10` -> `cluster_10...26`
```

```
head(acs_k_3)
```

```
   pop hh_income income county               tract
1 7304     56975  23750    031 Census Tract 8060.02
2 7577     53769  25016    031 Census Tract 8060.03
3 2684     62750  30154    031    Census Tract 8064
4 2590     53583  20282    031 Census Tract 8065.01
5 3594     40125  18347    031    Census Tract 7506
6 1521     63250  31403    031    Census Tract 3102
                                        NAME subregion cluster cluster_2...9
1 Census Tract 8060.02, Cook County, Illinois      cook       3             1
2 Census Tract 8060.03, Cook County, Illinois      cook       3             1
3    Census Tract 8064, Cook County, Illinois      cook       2             1
4 Census Tract 8065.01, Cook County, Illinois      cook       3             1
5    Census Tract 7506, Cook County, Illinois      cook       3             1
6    Census Tract 3102, Cook County, Illinois      cook       2             1
  cluster_3...10 cluster_4...11 cluster_5...12 cluster_6...13 cluster_7...14
1              1              4              5              5              7
2              1              4              5              5              5
3              2              4              5              4              7
4              1              4              5              5              5
5              1              2              1              5              5
6              2              4              5              4              7
  cluster_8...15 cluster_9...16 cluster_10...17 cluster_2...18 cluster_3...19
1              3              6               9              1              1
2              3              6               1              1              1
3              3              6               9              1              2
4              3              6               1              1              1
5              4              7              10              1              1
6              3              6               9              1              2
  cluster_4...20 cluster_5...21 cluster_6...22 cluster_7...23 cluster_8...24
1              4              5              5              7              3
2              4              5              5              5              3
3              4              5              4              7              3
4              4              5              5              5              3
5              2              1              5              5              4
6              4              5              4              7              3
  cluster_9...25 cluster_10...26
1              6               9
2              6               1
3              6               9
4              6               1
5              7              10
6              6               9
```