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.

Repository: https://github.com/sarahglidden97/Assignment-5.git

```
library(censusapi)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.2     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 becor
```

```
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(striprtf)
library(maps)
```

```
Attaching package: 'maps'

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

    map
```

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 <- read_rtf("~/Desktop/SURV 727/sg_key.rtf")

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)
```

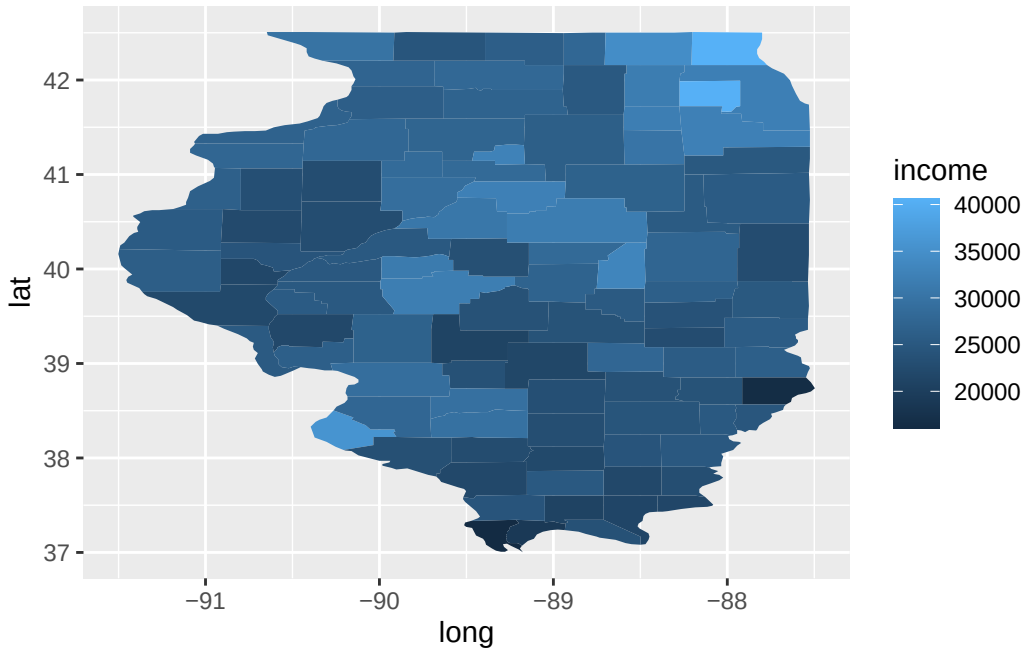
```
       long      lat group order   region subregion
1 -91.49563 40.21018     1     1 illinois     adams
2 -90.91121 40.19299     1     2 illinois     adams
3 -90.91121 40.19299     1     3 illinois     adams
4 -90.91121 40.10704     1     4 illinois     adams
5 -90.91121 39.83775     1     5 illinois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

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

```
# first write code to remove "County, Illinois" from NAME
words_to_remove <- c("County, Illinois", ".")

regex_pattern <- paste0("\\b(", paste(words_to_remove, collapse = "|"), ")\\b")

acs_il_c$NAME <- gsub(regex_pattern, "", acs_il_c$NAME)

# next, make all text lowercase
acs_il_c$NAME <- tolower(acs_il_c$NAME)

# remove "." from "st. clair" and rename column subregion
acs_il_c <- acs_il_c %>%
  mutate(NAME = ifelse(NAME == "st. clair", "stclair", NAME)) %>%
  rename(subregion = NAME)
```

```
# remove spaces from il_map subregion
il_map$subregion <- gsub(" ", "", il_map$subregion)

# join the two files
acs_map <- full_join(acs_il_c, il_map, by = "subregion")

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.

```
# population
hclust_data_pop <- acs_map %>%
  select(long, lat, pop, income, hh_income) %>%
```

```
  mutate_all(scale)

hclust_d_pop <- dist(hclust_data_pop)

hc_ward_pop <- hclust(hclust_d_pop, method = "ward.D2")
```

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

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

rect.hclust(hc_ward_pop,
            k = 5,
            border = "red")
```

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`.

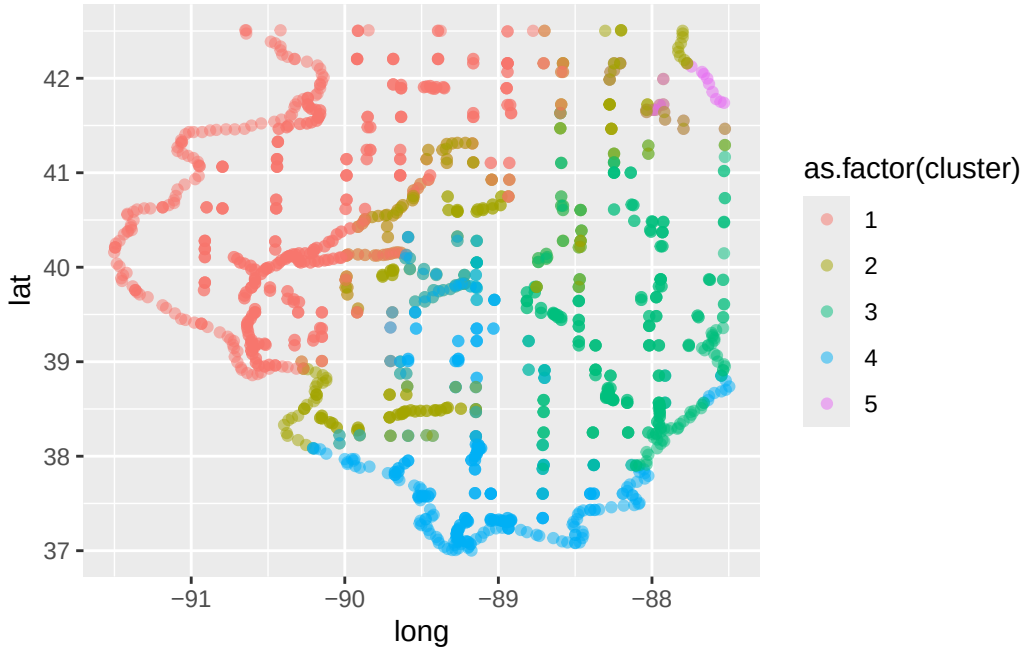
```
cutree(hc_ward_pop, 5)
```

```
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
 [38] 2 2 2 2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 [75] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[112] 3 3 3 1 1 1 1 1 1 1 1 1 3 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
[149] 5 5 5 5 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[186] 1 1 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 4 4 4 4
[223] 4 4 4 4 4 4 4 4 4 4 4 4 4 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
[260] 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
[297] 4 4 4 4 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
```

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

```
acs_map <- acs_map %>%
  mutate(cluster = cutree(hc_ward_pop, 5))

ggplot() +
```

```
  geom_point(data = acs_map, aes(x = long, y = lat, color = as.factor(cluster)), alpha = 0.5)
```



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)
```

```
Warning: `funs()` was deprecated in dplyr 0.8.0.
```

```
i Please use a list of either functions or lambdas:

# Simple named list: list(mean = mean, median = median)

# Auto named with `tibble::lst()`: tibble::lst(mean, median)

# Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
```

```
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
```

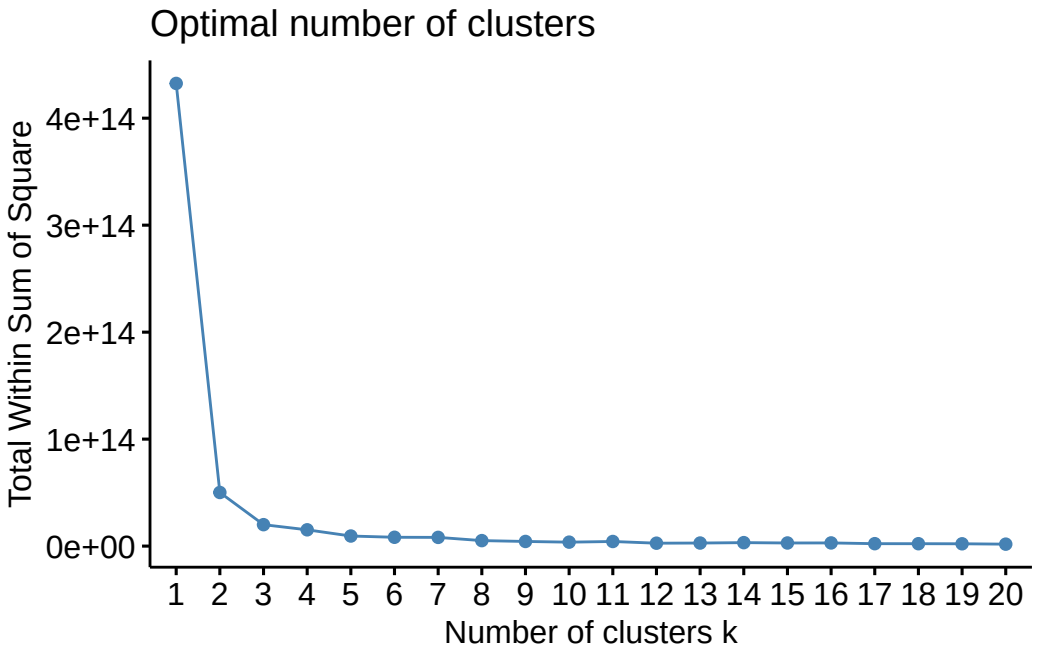
k-Means

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

```
acs_il_t_2 <- acs_il_t %>%
  filter(!(is.na(hh_income)))

acs_il_t_2_km <- acs_il_t_2 %>%
  select(tract, pop, hh_income, 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).

```
fviz_nbclust(acs_il_t_2_km, #data set we want to use
             kmeans, #cluster method
             method = "wss", #method used for estimating the optimal number of clusters
             k.max = 20)
```



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

```
km_1 <- kmeans(acs_il_t_2_km, 3, nstart = 20)
km_1
```

```
K-means clustering with 3 clusters of sizes 430, 926, 1753

Cluster means:
     tract      pop hh_income   income
1 417620.0 3498.560  45131.63 24217.13
2  37423.4 3935.695  55553.98 30931.06
3 853484.5 4387.902  68077.49 32047.34

Clustering vector:
   [1] 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 1 3 3
  [38] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
  [75] 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
 [112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 1 3 3 2 1 3 3 1 1 1 1 1 1 1 2 2 2 2
```

```
 [149] 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 3
 [186] 2 2 2 3 3 3 2 1 1 3 3 3 1 1 3 3 3 3 2 1 1 1 3 3 3 3 3 2 2 2 2 1 1 1 3 3 3
 [223] 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
 [260] 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [297] 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [334] 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 3 3 3 3 3 3
 [371] 3 3 3 3 3 3 3 3 2 2 2 2 2 1 1 1 2 3 3 3 2 2 2 2 2 1 1 1 1 1 1 1 1 3 3 2 2
 [408] 2 2 2 3 3 3 1 1 1 1 1 3 3 3 3 3 2 2 2 2 1 1 1 3 2 2 3 3 3 3 2 2 2 2 2 2 2
 [445] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3
 [482] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [519] 3 3 3 3 1 1 3 3 1 1 2 2 2 2 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [556] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [593] 3 3 3 3 3 3 2 2 2 2 2 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [630] 2 1 1 1 1 1 1 1 3 3 3 3 3 3 3 1 3 3 3 3 3 3 1 1 1 3 1 2 3 3 3 3 3 3 3 3 3
 [667] 3 3 3 3 3 3 3 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
 [704] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2
 [741] 2 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 1 1
 [778] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 3 1 3 3 3 1 1
 [815] 1 1 2 2 2 3 3 3 3 3 3 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [852] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3
 [889] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 3 3 3 3 3 3 2 2 2 2 2 2 2 2
 [926] 2 2 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 3 3 3 3 3
 [963] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[1000] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[1037] 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 3 3 3 3 3 3 3 3
[1074] 2 2 2 2 2 2 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 3 3 3
[1111] 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3
[1148] 3 1 1 2 2 2 2 1 1 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 2 2 2
[1185] 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[1222] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[1259] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 2 2 2 2 2 3 3 3 2 2 1 1 1 1 1 1 1
[1296] 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[1333] 3 3 3 3 3 3 3 3 3 3 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1
[1370] 1 1 3 3 3 2 3 3 3 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[1407] 3 3 3 1 1 2 2 2 2 1 1 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2
[1444] 2 2 2 2 2 2 2 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3
[1481] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2
[1518] 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 3 3 3
[1555] 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[1592] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3
[1629] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 2 2 2 2 2 2 2 2 2 2 1 1 3 3 1 1 1 1 1
[1666] 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 3 3 1 1 3 3 3 3 3 3
[1703] 2 2 2 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 3 3 3 3 3 3 3
```

```
[1740] 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 2 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[1777] 3 3 3 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 2 2 3 3 2 2 2 2 2 2 2 2 2
[1814] 2 2 2 2 3 3 3 3 3 2 2 2 2 2 2 2 2 2 3 3 3 3 2 2 2 2 3 3 3 3 3 3 3 3 1 1 1
[1851] 1 3 3 3 3 3 3 1 1 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 2 2 2 2 2 2 2 3 3 3
[1888] 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[1925] 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 3 1 1 3 3 3 1 1 3 3 3 3 3 3 3 3 2 1 3 3
[1962] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[1999] 3 2 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[2036] 1 1 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2
[2073] 2 2 2 2 2 2 2 3 3 3 2 2 3 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[2110] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 3 2 2 3 2 2 2 3 3
[2147] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 1 2 2 2 2 2 2 2 2 2 2
[2184] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[2221] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 2 1 1 1 1 1 1 1 1 1
[2258] 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[2295] 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2
[2332] 3 3 3 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 2 2 3 2 3 3 3 3
[2369] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2
[2406] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 2 2 2 3 3 3 3 3 3 3 3
[2443] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[2480] 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 3 3 2 2 2 2 1 1 1 1 1 1 1 1 1 1 3
[2517] 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 3 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[2554] 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[2591] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[2628] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2
[2665] 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[2702] 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 2 1 1 2 2 2 2 1 1 1
[2739] 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 1 3 3 3 3 2 2 3 3 3 3 2 2 2 2
[2776] 2 2 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 3 3 3 2 1 2 1 1 1 1 1 1 1 1 1 1 1
[2813] 1 1 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1
[2850] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[2887] 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
[2924] 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3
[2961] 3 3 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2
[2998] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 1 1 1 2 2 2 2 2 2 2
[3035] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2
[3072] 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2
[3109] 2

Within cluster sum of squares by cluster:
[1] 6.169149e+12 3.834391e+12 9.966867e+12
 (between_SS / total_SS =  95.4 %)
```

```
Available components:

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

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.

In cluster 1, the mean population is 3498.560 people. The average household income is $45131.63 and the average income is $24217.13. In cluster 2, the mean population is 3935.695 people. The average household income is $55553.98 and the average income is $30931.06. In cluster 3, the mean population is 4387.902 people. The average household income is $68077.49 and the average income is $32047.34.

```
acs_il_t_2$cluster <- km_1$cluster

county_cluster <- acs_il_t_2 %>%
  group_by(cluster, county) %>%
  summarise(county_count = n()) %>%
  arrange(desc(county_count))
```

```
`summarise()` has grouped output by 'cluster'. You can override using the
`.groups` argument.
```

For all three clusters, the county that shows up most frequently is 031, which is Cook County.

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.

```
km_option <- function(K) {
  kmeans(acs_il_t_2_km, K, nstart = 20)
}

# check to see that function matches current output, it does
km_option(3)
```

```
K-means clustering with 3 clusters of sizes 1753, 430, 926

Cluster means:
     tract      pop hh_income   income
```

```
1 853484.5 4387.902  68077.49 32047.34
2 417620.0 3498.560  45131.63 24217.13
3  37423.4 3935.695  55553.98 30931.06

Clustering vector:
   [1] 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 2 1 1
  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  [75] 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1
 [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 2 1 1 3 2 1 1 2 2 2 2 2 2 2 3 3 3 3
 [149] 3 3 3 3 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1
 [186] 3 3 3 1 1 1 3 2 2 1 1 1 2 2 1 1 1 1 3 2 2 2 1 1 1 1 1 3 3 3 3 2 2 2 1 1 1
 [223] 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
 [260] 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [297] 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [334] 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 1 1 1 1 1 1
 [371] 1 1 1 1 1 1 1 1 3 3 3 3 3 2 2 2 3 1 1 1 3 3 3 3 3 2 2 2 2 2 2 2 2 1 1 3 3
 [408] 3 3 3 1 1 1 2 2 2 2 2 1 1 1 1 1 3 3 3 3 2 2 2 1 3 3 1 1 1 1 3 3 3 3 3 3 3
 [445] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1
 [482] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [519] 1 1 1 1 2 2 1 1 2 2 3 3 3 3 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [556] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [593] 1 1 1 1 1 1 3 3 3 3 3 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [630] 3 2 2 2 2 2 2 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 2 2 1 2 3 1 1 1 1 1 1 1 1 1
 [667] 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1
 [704] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3
 [741] 3 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 2 2
 [778] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 1 2 1 1 1 2 2
 [815] 2 2 3 3 3 1 1 1 1 1 1 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [852] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1
 [889] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3
 [926] 3 3 1 1 1 1 1 1 1 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 1 1 1 1 1 1
 [963] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[1000] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[1037] 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 1 1 1 1 1 1 1 1 1
[1074] 3 3 3 3 3 3 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 1 1 1
[1111] 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1
[1148] 1 2 2 3 3 3 3 2 2 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 3 3 3
[1185] 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[1222] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[1259] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 3 3 3 3 3 1 1 1 3 3 2 2 2 2 2 2 2 2
[1296] 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1
[1333] 1 1 1 1 1 1 1 1 1 1 1 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
[1370] 2 2 1 1 1 3 1 1 1 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
[1407] 1 1 1 2 2 3 3 3 3 2 2 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3
[1444] 3 3 3 3 3 3 3 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1
[1481] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3
[1518] 3 3 3 3 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 1 1 1
[1555] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[1592] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1
[1629] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 3 3 3 3 3 3 3 3 3 2 2 1 1 2 2 2 2 2
[1666] 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 1 1 2 2 1 1 1 1 1 1
[1703] 3 3 3 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 1 1 1 1 1 1 1
[1740] 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 3 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[1777] 1 1 1 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 3 3 1 1 3 3 3 3 3 3 3 3 3 3
[1814] 3 3 3 3 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 1 1 1 1 1 1 1 1 2 2 2
[1851] 2 1 1 1 1 1 2 2 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 3 3 3 3 3 3 3 1 1 1
[1888] 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[1925] 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 1 1 2 2 1 1 1 2 2 1 1 1 1 1 1 1 1 3 2 1 1
[1962] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[1999] 1 3 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[2036] 2 2 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[2073] 3 3 3 3 3 3 1 1 1 3 3 1 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[2110] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 1 3 3 1 3 3 3 1 1
[2147] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 2 3 3 3 3 3 3 3 3 3
[2184] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[2221] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 3 2 2 2 2 2 2 2 2 2
[2258] 2 2 2 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[2295] 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3
[2332] 1 1 1 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 1 3 1 1 1 1
[2369] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3
[2406] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 3 3 3 1 1 1 1 1 1 1 1 1
[2443] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[2480] 3 3 3 3 3 3 3 3 3 3 2 2 3 3 3 3 3 3 3 1 1 3 3 3 3 2 2 2 2 2 2 2 2 2 2 1
[2517] 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 1 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[2554] 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[2591] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[2628] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 1 1 3 3 3 3 3 3 3 3 3 3 3
[2665] 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[2702] 1 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 3 3 2 2 3 3 3 3 2 2 2
[2739] 2 2 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 2 1 1 1 1 3 3 1 1 1 1 3 3 3 3
[2776] 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 1 1 1 3 2 3 2 2 2 2 2 2 2 2 2 2
[2813] 2 2 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2
[2850] 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[2887] 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
[2924] 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1
[2961] 1 1 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3
```

```
[2998] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 2 2 2 3 3 3 3 3 3 3
[3035] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3
[3072] 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3
[3109] 3

Within cluster sum of squares by cluster:
[1] 9.966867e+12 6.169149e+12 3.834391e+12
 (between_SS / total_SS =  95.4 %)

Available components:

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

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.

```
for(i in 2:10) {
  colname <- paste0("kmeans_", i)

  cluster <- km_option(i)$cluster

  acs_il_t_2[[colname]] <- cluster
}
```

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

```
head(acs_il_t_2)
```

```
  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 cluster kmeans_2 kmeans_3 kmeans_4 kmeans_5 kmeans_6
1     56975  23750       3        1        3        4        1        2
2     53769  25016       3        1        3        4        1        2
```

```
3     62750  30154       3        1        3        4        1        2
4     53583  20282       3        1        3        4        1        2
5     40125  18347       3        1        3        4        1        2
6     63250  31403       1        2        2        3        2        6
  kmeans_7 kmeans_8 kmeans_9 kmeans_10
1        2        5        1         6
2        2        5        1         6
3        2        5        1         6
4        2        5        1         6
5        2        5        1         6
6        1        6        9         3
```