Assignment 5

Due at 11:59pm on November 25.

Zachary Winograd

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.

Github Link:https://github.com/Zackwin73/Assignment_5

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

```r
cs_key <- readLines("census-key.txt")
```

```
Warning in readLines("census-key.txt"): incomplete final line found on
'census-key.txt'
```

```r
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_c <- acs_il_c %>%
  separate(NAME, into = c("county_name", "state_name"), sep = ",") %>%
  mutate(
    county_clean = county_name %>%
      str_replace(" County", "") %>%
      str_trim() %>%
      tolower()
  )

acs_map <- il_map %>%
  left_join(acs_il_c, by = c("subregion" = "county_clean"))
```

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))
```
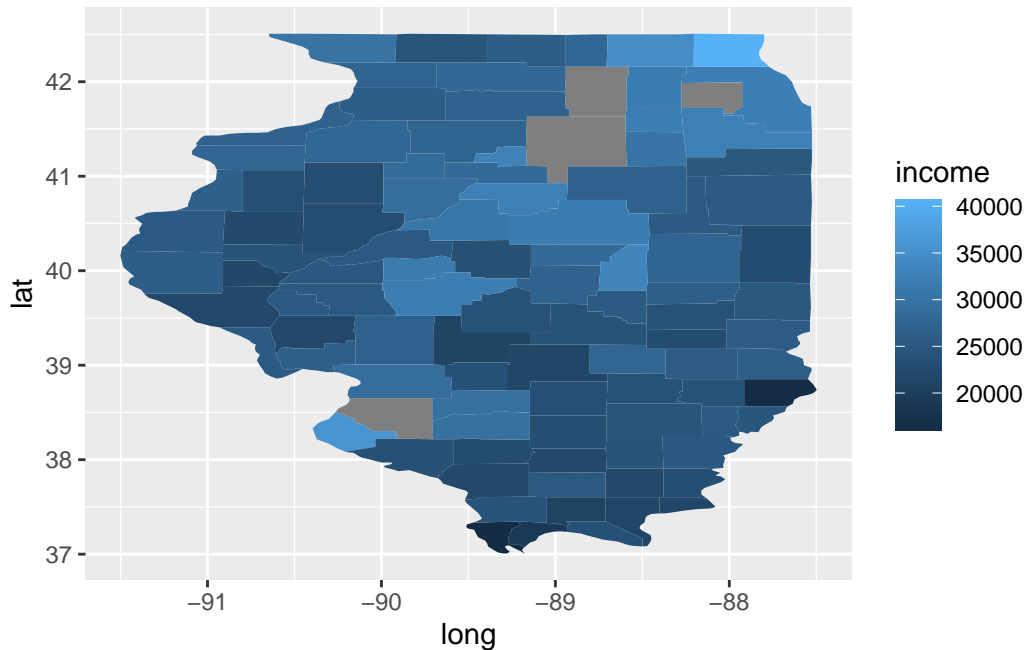

Hierarchical Clustering

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.

```
county_clust <- acs_il_c %>%
  select(pop, hh_income, income) %>%
  drop_na() %>%
  scale()

county_dist <- dist(county_clust)

county_hc <- hclust(county_dist, method = "ward.D2")
```

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

```
plot(county_hc)
rect.hclust(county_hc, k = 4)
```

Cluster Dendrogram

county_dist
hclust (*, "ward.D2")
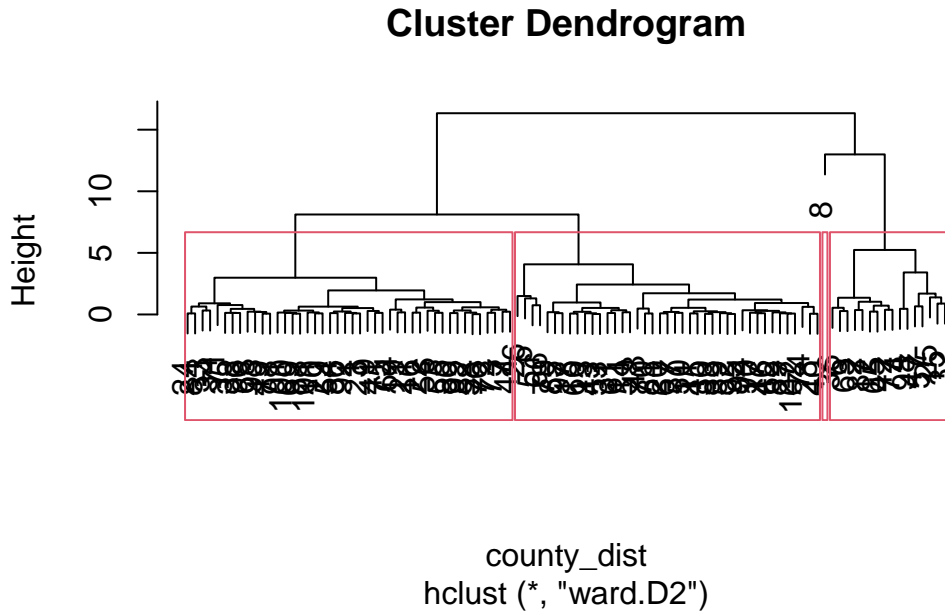
Visualize the county clusters on a map. For this task, create a new `acs_map` object that now also includes cluster membership as a new column. This column should be called `cluster`.

```
# Assign cluster membership to each county
acs_il_c$cluster <- cutree(county_hc, k = 4)

# Join cluster info into the map data
acs_map <- il_map %>%
  left_join(acs_il_c %>% select(county_clean, cluster),
            by = c("subregion" = "county_clean"))


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

4
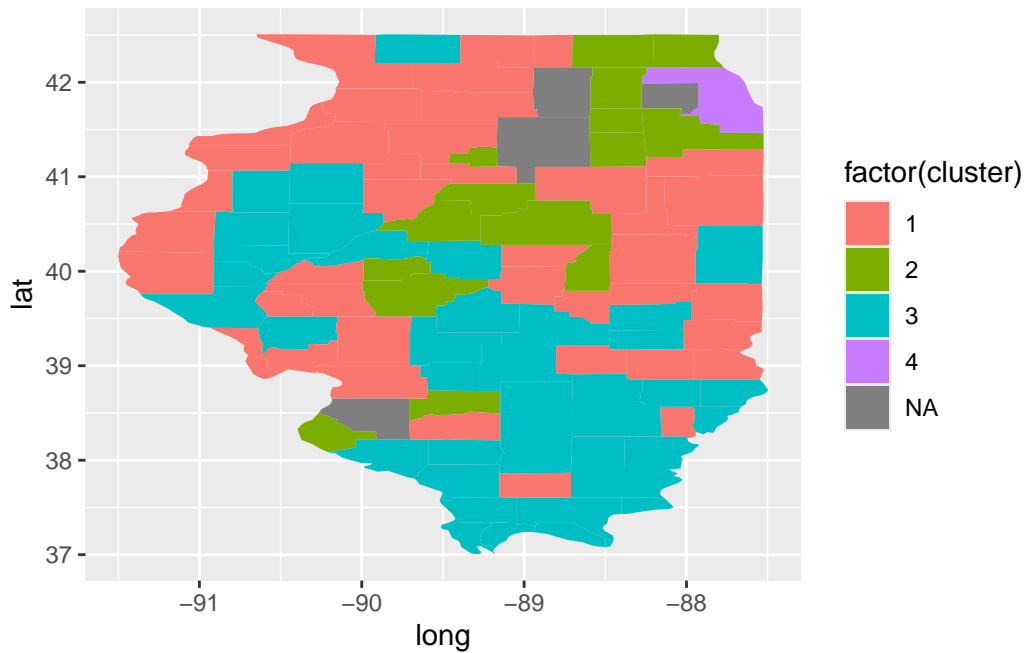
Census Tracts

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)

acs_il_t <- acs_il_t %>%
  mutate_all(function(x) ifelse(x == -666666666, NA, x)) %>%
  rename(pop = B01003_001E,
         hh_income = B19013_001E,
         income = B19301_001E)

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 <- acs_il_t %>% drop_na(pop, hh_income, income)

tract_clust <- acs_il_t %>%
  select(pop, hh_income, income) %>%
  drop_na() %>%
  scale()
```

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).

```
wss <- numeric(20)

for (k in 1:20) {
  km <- kmeans(tract_clust, centers = k, nstart = 20)
  wss[k] <- km$tot.withinss
}
```

```
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

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

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

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
```
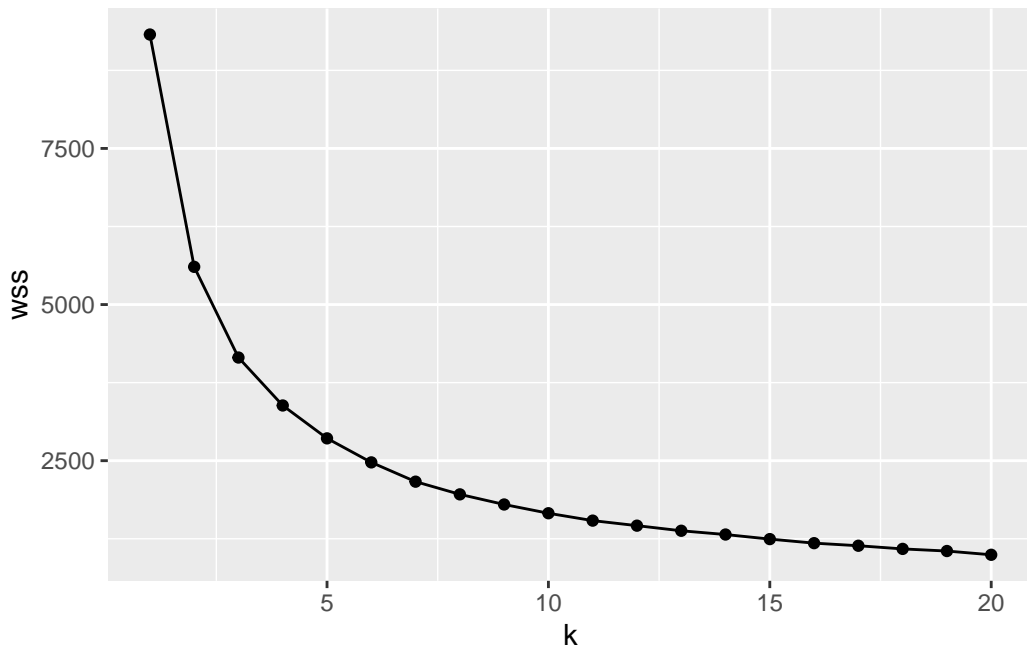
```
wss_df <- tibble(k = 1:20, wss = wss)

ggplot(wss_df, aes(x = k, y = wss)) +
  geom_line() +
  geom_point()
```


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

```
k_opt <- 5
set.seed(123)
km_tract <- kmeans(tract_clust, centers = k_opt, nstart = 20)

km_tract
```

```
K-means clustering with 5 clusters of sizes 276, 1016, 274, 744, 799

Cluster means:
        pop  hh_income     income
1 -0.1195090  2.0571594  2.3390317
2 -0.7364571 -0.8085000 -0.6855636
3  1.8913853  0.8349089  0.4751140
4  0.6382506 -0.4005122 -0.4646770
5 -0.2651728  0.4041002  0.3335397
```

```
Clustering vector:
   [1] 4 4 5 2 2 5 2 5 2 2 4 2 1 1 1 1 1 4 1 5 4 5 5 2 2 5 3 5 2 4 2 2 2 2 2 5 5
  [38] 4 4 2 5 4 4 5 5 5 5 5 5 3 4 2 5 2 4 3 2 2 2 2 2 2 4 2 3 3 2 2 2 4 2 5 5 2
  [75] 2 2 2 2 2 5 2 2 1 1 5 1 5 1 5 1 1 1 1 1 1 1 5 2 2 4 4 1 1 1 3 3 3 1 3 3 3
 [112] 3 4 5 5 5 4 3 5 4 1 1 1 1 5 5 4 5 2 5 2 4 2 4 2 2 2 4 5 2 2 2 4 4 1 2 4 4
 [149] 2 4 5 4 2 2 4 4 2 2 4 2 4 2 2 2 5 2 4 2 2 2 1 5 3 2 2 2 2 2 4 5 4 3 5 4 2 2
 [186] 4 2 2 2 2 2 5 2 4 4 2 5 2 4 4 4 3 4 4 2 5 2 4 5 4 5 5 1 5 4 5 4 2 2 2 2 2
 [223] 5 2 5 5 2 3 3 2 2 4 4 2 4 2 2 2 2 4 5 3 4 5 2 4 2 3 5 4 2 5 2 2 4 5 2 4 4
 [260] 2 2 5 5 5 5 5 4 1 4 5 1 5 1 5 5 1 5 5 5 5 1 1 5 5 4 5 5 1 5 5 1 1 5 5 5 5
 [297] 2 2 4 4 4 2 4 4 2 4 4 4 2 5 5 5 4 2 2 4 3 4 4 5 5 5 3 5 1 5 3 5 4 3 4 4 4
 [334] 5 2 2 5 4 3 3 3 1 1 1 1 2 4 5 5 3 4 4 5 3 4 2 2 1 3 3 5 4 2 5 5 5 5 2 5 2
 [371] 2 4 2 2 2 2 2 4 5 2 4 2 5 2 4 2 5 5 4 4 2 5 2 5 2 4 4 1 2 1 1 3 2 2 5 2 2
 [408] 4 5 5 5 3 3 2 2 2 1 2 5 1 1 3 3 5 2 5 1 5 2 5 2 5 1 2 4 2 2 4 2 5 4 2 2 2
 [445] 2 5 4 4 2 2 5 2 2 5 4 1 1 1 1 1 1 1 5 2 4 4 2 5 1 4 5 4 4 5 5 4 2 5 4 4 4
 [482] 4 1 4 4 5 4 5 3 5 4 2 4 4 2 5 5 5 5 3 3 1 3 5 5 5 5 5 5 5 1 5 3 5 5 5 2 4
 [519] 3 5 3 3 2 2 5 5 3 3 4 1 5 1 4 2 2 2 4 4 2 2 2 2 2 5 5 2 1 1 1 1 3 3 1 4 5
 [556] 4 4 4 2 2 5 5 2 5 5 2 5 5 3 3 5 5 2 4 5 1 1 1 1 1 1 1 1 5 1 1 5 1 1 5 5 5 4
 [593] 2 2 2 2 1 5 4 3 5 2 5 4 2 4 2 5 2 2 4 3 3 4 5 3 2 2 2 4 4 5 2 2 2 2 2 2 4
 [630] 2 2 1 2 2 2 2 2 4 2 2 2 2 2 5 4 2 2 4 3 1 4 2 2 2 5 2 1 3 1 2 5 4 2 5 2 2
 [667] 2 4 5 4 2 2 2 4 2 2 5 2 4 5 4 2 4 2 3 5 3 2 1 1 5 5 5 1 5 5 5 3 5 5 3 5 5
 [704] 4 3 5 5 2 1 5 1 5 1 1 5 4 4 4 4 4 1 2 1 2 5 5 2 5 5 5 5 5 5 5 5 4 5 2 4 5
 [741] 4 5 4 4 4 4 2 5 5 2 4 2 2 3 4 2 5 3 4 1 5 3 2 4 5 5 3 1 2 2 2 2 2 2 4 2 5
 [778] 2 2 5 2 2 2 2 2 5 2 2 4 2 2 5 5 1 1 1 5 1 1 1 3 5 5 3 2 1 5 4 2 3 5 4 2 2
 [815] 4 2 4 3 1 4 2 2 2 2 2 5 2 5 4 2 5 2 5 5 2 2 5 4 5 5 2 2 4 2 4 2 2 4 5 4 2
 [852] 2 2 2 2 5 5 2 5 3 5 3 3 3 4 4 1 4 4 5 1 1 1 5 4 5 5 1 5 2 2 5 1 1 3 5 5 5
 [889] 4 5 3 5 3 1 5 4 4 5 5 3 5 3 3 4 3 2 4 3 3 5 2 2 2 5 5 2 4 2 2 4 4 4 2 2 2
 [926] 5 4 1 4 5 4 4 2 2 2 2 2 2 4 4 4 4 5 5 5 5 2 4 3 2 2 4 4 2 5 5 5 3 5 5 5
 [963] 5 5 5 2 2 2 5 5 2 2 2 2 5 2 2 2 2 5 5 4 3 3 5 3 5 5 5 5 5 4 4 2 2 2 2 2 3
[1000] 5 4 1 1 2 2 4 4 2 2 2 2 2 2 4 2 1 5 5 4 1 1 4 4 4 1 1 1 1 5 3 4 4 5 2 1 1
[1037] 3 5 5 5 2 4 2 4 2 4 4 4 2 4 2 4 2 4 2 4 2 4 2 2 2 4 2 4 5 5 2 3 3 5 2 2 2
[1074] 2 2 4 2 4 5 2 1 3 3 3 5 3 3 3 5 5 4 5 4 4 2 5 5 1 4 3 4 5 3 2 5 4 5 4 4 5 2
[1111] 2 2 4 5 2 4 2 2 4 2 5 2 5 4 2 4 2 2 2 4 4 4 3 2 4 2 4 5 4 4 2 4 4 4 2 4 4
[1148] 2 2 2 5 5 1 1 4 4 2 4 5 2 2 2 2 2 2 4 2 2 2 2 4 5 4 4 2 2 2 2 2 5 3 5 5
[1185] 2 5 2 4 2 4 1 5 4 4 3 2 2 3 5 5 4 2 5 5 5 4 1 1 5 5 5 1 3 5 1 5 5 3 5 1 5
[1222] 5 3 5 1 1 4 2 4 2 3 3 4 5 4 1 3 1 5 4 5 5 4 3 5 5 2 1 1 4 5 4 5 1 4 2 3 5
[1259] 5 1 5 2 5 4 1 1 4 5 5 4 5 1 4 2 4 4 2 4 5 3 5 1 5 5 4 4 5 2 4 2 5 5 2 4 4
[1296] 2 4 4 4 2 2 4 4 4 4 2 4 4 4 2 4 2 4 2 2 5 4 4 2 2 2 4 2 4 1 1 5 1 1 5 1 2
[1333] 2 4 2 2 2 5 2 4 5 2 4 4 3 5 4 4 5 5 4 5 1 4 4 4 5 2 2 4 4 2 5 4 3 2 4 2 2
[1370] 2 4 2 2 2 2 2 2 5 5 4 2 4 2 4 4 2 2 5 2 4 4 5 5 4 4 5 5 4 4 2 4 4 5 4
[1407] 3 4 4 2 2 4 2 2 5 2 5 2 3 4 5 5 4 5 5 2 4 2 2 2 5 2 4 2 2 2 5 2 2 2 2 5 2
[1444] 2 2 2 5 5 5 4 5 3 1 4 4 4 5 4 5 4 2 4 5 2 2 1 4 4 4 4 4 3 2 5 5 4 5 2 2
[1481] 5 1 1 5 5 1 1 1 1 5 4 3 5 1 5 1 1 1 1 4 1 5 5 3 2 2 2 4 4 5 5 4 2 4 4 4 5
```

9

```
[1518] 4 4 4 4 5 3 4 2 2 2 2 5 4 5 4 5 4 4 4 4 4 3 4 2 2 2 2 2 2 4 2 4 3 3 5
[1555] 3 5 1 1 3 4 3 2 1 2 4 2 2 2 2 2 4 2 2 2 2 4 5 5 4 4 4 2 5 4 4 2 4 3 4 1 5
[1592] 1 1 5 3 5 5 5 3 1 3 2 2 4 2 5 4 5 5 5 4 2 5 5 2 5 5 2 2 2 5 4 4 3 3 2 4 3
[1629] 3 1 3 4 4 4 4 2 5 5 5 4 4 4 5 5 2 4 2 2 4 2 2 2 4 5 2 4 2 2 5 4 2 2 2 4 2
[1666] 2 2 4 2 2 2 2 2 2 2 5 1 3 5 4 4 3 2 2 3 4 5 3 3 4 5 4 4 3 1 2 3 3 5 5 4 5
[1703] 4 2 5 4 2 2 2 2 2 4 2 2 5 1 1 5 5 5 5 2 2 5 2 2 2 2 2 4 4 2 2 5 4 2 5 5 4
[1740] 4 2 5 4 5 4 2 2 2 2 4 2 2 2 2 2 4 4 4 4 4 4 5 4 4 2 2 5 2 2 2 2 2 2 2 2
[1777] 2 2 2 2 2 3 5 4 5 2 5 1 2 3 5 5 5 5 3 3 3 4 2 2 2 1 4 2 2 5 3 5 2 2 5 4 5
[1814] 3 5 2 5 4 2 2 4 2 5 4 3 4 2 2 2 4 2 4 3 3 5 4 2 4 5 1 4 5 5 5 1 4 2 2 2 2
[1851] 4 3 5 5 5 1 2 4 2 4 2 2 2 4 2 2 2 4 2 4 1 3 3 5 2 2 4 5 4 5 2 2 2 2 4 4 5
[1888] 4 4 4 5 5 2 2 2 3 4 2 5 2 5 2 2 3 1 2 2 5 2 5 2 2 2 5 2 2 2 5 5 5 2 2 5 4
[1925] 2 5 5 5 2 1 1 2 4 4 2 2 2 2 2 2 2 2 2 2 3 5 5 4 2 2 2 1 3 1 4 5 5 2 2 3 4
[1962] 2 5 4 2 4 1 1 5 4 1 3 3 4 4 2 2 2 5 5 2 2 2 2 2 2 5 2 2 5 4 2 5 2 5 2 2 2
[1999] 2 2 4 5 5 4 2 4 4 4 2 2 4 2 2 2 4 2 2 2 3 4 5 5 4 4 3 4 5 4 5 2 4 3 5 4 4
[2036] 2 5 5 5 2 5 5 2 2 5 4 2 2 5 4 5 5 2 2 5 4 2 5 5 2 2 4 5 2 5 2 2 5 5 3 2 5
[2073] 2 2 2 5 4 5 2 2 2 2 2 3 3 4 2 5 3 5 4 4 2 5 3 5 5 5 5 2 5 5 4 4 4 5 1 5 5
[2110] 2 2 2 2 2 4 4 2 4 5 5 2 3 5 1 5 1 1 5 2 2 5 4 5 4 4 3 4 5 5 2 2 4 4 2 2 5
[2147] 3 1 3 4 4 4 5 5 2 2 4 5 4 4 4 5 4 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 4 4 5 5 3
[2184] 2 2 5 5 2 4 2 2 2 5 2 2 2 4 5 2 4 4 2 1 4 4 2 5 2 5 2 2 2 4 2 2 2 4 5 2 2
[2221] 2 2 4 4 3 5 2 5 5 4 5 2 3 5 5 5 5 2 4 4 2 5 3 4 4 5 4 2 2 2 2 5 5 1 4 2 4
[2258] 2 2 2 1 4 3 4 4 4 4 4 5 4 1 5 5 5 5 4 2 5 2 3 4 5 5 5 5 5 2 5 3 5 4 3 4 2
[2295] 2 2 2 2 4 2 2 3 4 2 3 5 2 2 2 5 2 4 4 4 5 5 5 3 1 1 4 3 1 5 1 5 1 1 4 2 2
[2332] 2 2 2 2 2 2 4 2 2 4 5 2 3 4 5 5 5 3 4 5 5 4 3 3 5 4 5 3 5 2 2 3 4 2 4 5 1
[2369] 3 2 5 1 1 1 5 5 5 4 1 5 5 4 3 5 5 5 1 5 2 5 2 4 2 2 2 5 2 4 2 2 4 4 2 5 5
[2406] 3 5 1 1 1 1 4 5 5 5 1 2 3 2 2 3 2 5 5 2 2 2 2 5 2 2 4 4 2 1 4 5 1 3 3 1 1
[2443] 3 5 3 4 5 5 2 2 2 2 1 1 2 5 4 3 1 1 2 1 3 1 4 3 5 2 2 4 4 5 3 1 3 5 5 5 5
[2480] 4 5 2 4 4 5 4 4 5 4 2 2 1 1 1 1 5 5 4 4 5 1 2 5 2 2 2 5 5 1 2 2 2 2 2 4 4
[2517] 4 3 5 5 4 5 5 3 5 2 4 4 4 4 2 4 4 4 3 3 4 2 2 2 4 2 5 5 2 5 2 2 5 2 2 5 2
[2554] 4 2 5 2 2 4 2 4 3 3 4 5 2 2 4 4 5 4 5 5 2 5 1 1 5 4 4 4 4 4 4 4 5 5 5 5 5
[2591] 5 5 5 5 5 4 5 5 5 1 5 5 3 5 2 3 4 4 3 2 4 2 4 5 4 2 4 4 4 4 4 5 4 5 4 3 4
[2628] 5 3 3 3 5 4 3 3 3 3 5 5 5 4 4 4 5 4 2 4 4 4 4 5 4 2 2 2 2 2 2 2 4 2 4 2
[2665] 5 5 5 4 2 5 1 1 1 2 4 5 4 5 2 1 1 3 4 5 5 3 5 5 4 4 3 3 5 4 3 1 3 4 5 5 3
[2702] 5 5 5 4 2 3 3 5 4 5 2 4 5 3 5 5 4 3 3 2 2 2 1 1 1 5 2 4 5 2 4 2 2 5 2 4 2
[2739] 2 2 1 2 2 4 1 1 5 5 4 2 5 5 4 2 4 4 2 4 4 4 2 4 2 2 2 5 2 1 5 5 5 3 3 3 3
[2776] 5 4 5 5 5 5 5 5 3 1 1 5 5 4 4 2 3 3 3 4 5 2 2 2 4 2 2 5 3 4 4 4 5 3 5 5 3
[2813] 3 2 5 4 5 5 2 1 1 3 2 4 2 4 2 2 2 4 4 2 5 3 5 5 5 5 3 4 1 4 5 3 1 1 1 2 2
[2850] 4 2 4 2 2 2 2 2 2 2 2 2 2 5 2 4 2 4 3 5 5 5 3 2 5 2 3 5 4 2 2 4 2 5 2 2 1
[2887] 3 1 1 1 2 2 2 5 5 2 1 2 2 2 4 2 4 2 4 2 2 5 5 2 2 1 4 5 3 5 5 1 5 1 1 1 1
[2924] 1 1 1 1 5 2 5 4 2 2 2 2 4 2 2 2 2 2 2 2 5 5 5 4 5 4 5 3 5 5 5 4 2 5 4 4 4 4
[2961] 4 4 2 2 2 1 2 3 3 5 3 3 4 3 5 3 3 2 5 3 3 5 5 2 2 2 2 2 4 4 2 2 2 2 2 2 4
[2998] 4 3 2 4 5 4 2 2 2 2 2 5 5 2 2 2 2 2 4 4 5 2 4 4 5 5 5 2 2 2 2 4 4 4 3 5 4
[3035] 2 5 4 3 5 4 4 4 5 5 4 4 5 5 5 5 4 4 2 2 2 5 2 2 2 2 4 4 4 2 4 2 2 2 2 3 4
[3072] 5 5 5 2 5 2 4 4 4 4 4 4 5 5 4 4 4 4 2 4 4 4 4 2 2 2 4 5 4 1 4 2 3 5 4 2 2 2
```

10

```
[3109] 2

Within cluster sum of squares by cluster:
[1] 738.5933 490.3378 689.9021 390.6437 548.2085
 (between_SS / total_SS =  69.4 %)

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.

```r
acs_il_t <- acs_il_t %>%
  mutate(
    county_name = str_extract(NAME, "[^,]+ County") %>%
      str_replace(" County", "") %>%
      str_trim()
  )

acs_il_t$cluster <- km_tract$cluster


cluster_summary <- acs_il_t %>%
  group_by(cluster) %>%
  summarise(
    mean_pop = mean(pop, na.rm = TRUE),
    mean_hh_income = mean(hh_income, na.rm = TRUE),
    mean_income = mean(income, na.rm = TRUE),
    most_frequent_county = county_name %>%
      as.factor() %>%
      fct_count(sort = TRUE) %>%
      slice(1) %>%
      pull(f)
  )

cluster_summary
```

```
# A tibble: 5 x 5
  cluster mean_pop mean_hh_income mean_income most_frequent_county
    <int>    <dbl>          <dbl>       <dbl> <fct>
```

```
1      1    3896.       122368.       67665. Cook
2      2    2686.        37123.       19778. Cook
3      3    7838.        86010.       38154. Cook
4      4    5381.        49260.       23275. Cook
5      5    3610.        73195.       35913. Cook
```

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.

```r
run_kmeans <- function(k) {
  set.seed(123)
  kmeans(tract_clust, centers = k, nstart = 20)
}
```

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.

```r
for (k in 2:10) {
  km <- run_kmeans(k)
  acs_il_t[[paste0("cluster_k", k)]] <- km$cluster
}
```

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

```r
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 county_name cluster
1     56975  23750        Cook       4
2     53769  25016        Cook       4
3     62750  30154        Cook       5
```

```
4       53583   20282           Cook            2
5       40125   18347           Cook            2
6       63250   31403           Cook            5
```