Assignment 5

Sungjoo Cho and Kate Lamoreaux

2023-12-04

GitHub link:

https://github.com/katelmrx/cho-lamoreaux-a1.git

Packages

```r
library(censusapi)
library(tidyverse)
library(magrittr)
library(factoextra)
library(base)
library(RSocrata)
library(ggmap)
library(lubridate)
library(corrplot)
library(dplyr)
library(knitr)
```

Exploring ACS Data

```r
cs_key <- readLines("cs_key.TeX")
```

```r
# Data from the American Community Survey (ACS)
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. Note 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.

```
# creating county_name variable in acs_il
acs_il_c$subregion <- gsub(", .*", "", acs_il_c$NAME)
acs_il_c$subregion <- gsub("County", "", acs_il_c$subregion)
acs_il_c$subregion <- trimws(acs_il_c$subregion, "right")
acs_il_c$subregion <- tolower(acs_il_c$subregion)

# join
acs_map <- inner_join(acs_il_c, il_map, by=join_by("subregion" == "subregion"))
head(acs_map)
```

```
##   state county                     NAME   pop hh_income income subregion
## 1    17    067 Hancock County, Illinois 18633     50077   25647   hancock
## 2    17    067 Hancock County, Illinois 18633     50077   25647   hancock
## 3    17    067 Hancock County, Illinois 18633     50077   25647   hancock
## 4    17    067 Hancock County, Illinois 18633     50077   25647   hancock
## 5    17    067 Hancock County, Illinois 18633     50077   25647   hancock
## 6    17    067 Hancock County, Illinois 18633     50077   25647   hancock
##        long      lat group order   region
## 1 -91.18623 40.63417    34   573 illinois
## 2 -90.89976 40.63417    34   574 illinois
## 3 -90.91121 40.27893    34   575 illinois
## 4 -90.91121 40.19299    34   576 illinois
## 5 -91.49563 40.21018    34   577 illinois
## 6 -91.48990 40.25029    34   578 illinois
```

```
# number of rows
dim(acs_il_c)
```

```
## [1] 102   7
```

```
dim(il_map)
```

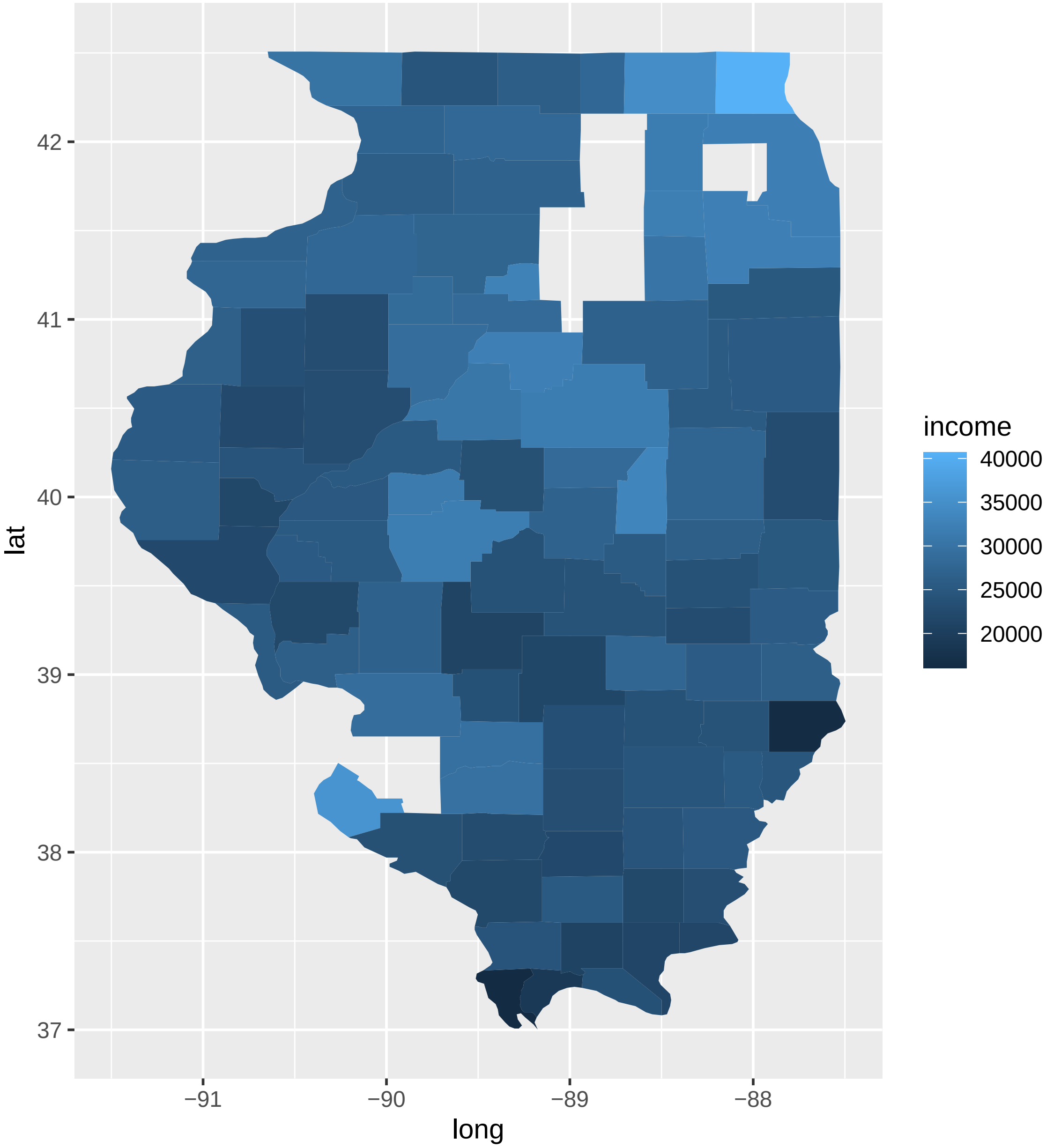
```
## [1] 1697    6
```

```
dim(acs_map)
```

```
## [1] 1642   12
```

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.

```r
head(acs_map)
```

```
##   state county                     NAME   pop hh_income income subregion
## 1    17    067 Hancock County, Illinois 18633     50077  25647   hancock
## 2    17    067 Hancock County, Illinois 18633     50077  25647   hancock
## 3    17    067 Hancock County, Illinois 18633     50077  25647   hancock
## 4    17    067 Hancock County, Illinois 18633     50077  25647   hancock
## 5    17    067 Hancock County, Illinois 18633     50077  25647   hancock
## 6    17    067 Hancock County, Illinois 18633     50077  25647   hancock
##        long      lat group order   region
## 1 -91.18623 40.63417    34   573 illinois
## 2 -90.89976 40.63417    34   574 illinois
## 3 -90.91121 40.27893    34   575 illinois
## 4 -90.91121 40.19299    34   576 illinois
## 5 -91.49563 40.21018    34   577 illinois
## 6 -91.48990 40.25029    34   578 illinois
```

```r
# select pop, hh_income, and income variables
hclust_data <- acs_map %>%
  select(pop, hh_income, income)

# cleaned data
head(hclust_data)
```

```
##     pop hh_income income
## 1 18633     50077  25647
## 2 18633     50077  25647
## 3 18633     50077  25647
## 4 18633     50077  25647
## 5 18633     50077  25647
## 6 18633     50077  25647
```

Next, create the distance matrix of the cleaned data. This distance matrix can be used to cluster counties, e.g. using the ward method.

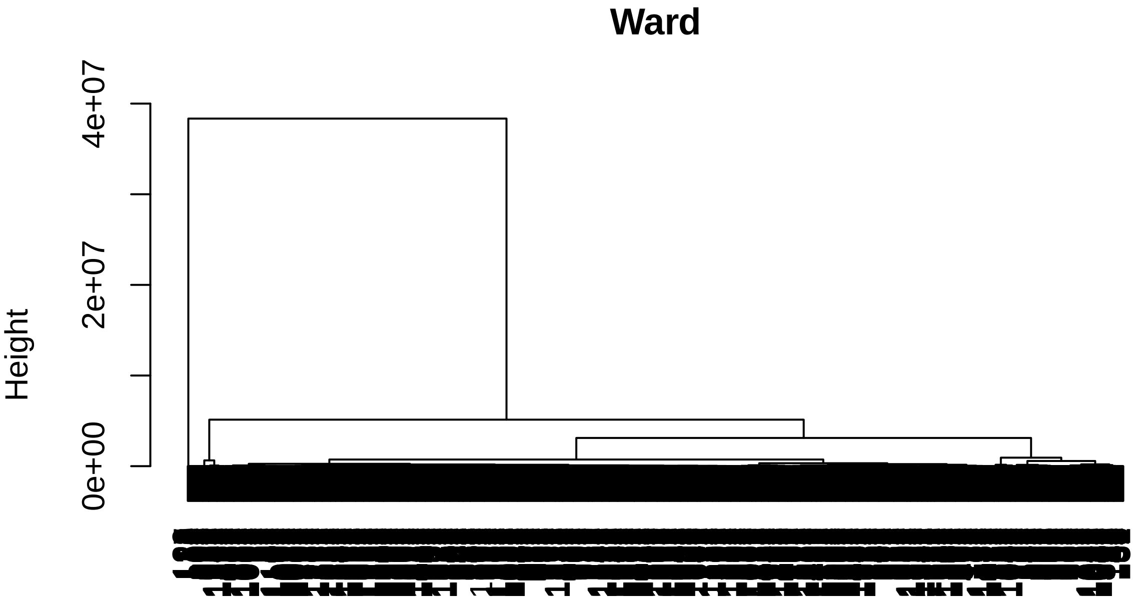
```r
# distance matrix of the cleaned data
hclust_d <- dist(hclust_data)
as.matrix(hclust_data)[1:10,]
```

```
##         pop hh_income income
##  [1,] 18633     50077  25647
##  [2,] 18633     50077  25647
##  [3,] 18633     50077  25647
##  [4,] 18633     50077  25647
##  [5,] 18633     50077  25647
##  [6,] 18633     50077  25647
##  [7,] 18633     50077  25647
##  [8,] 18633     50077  25647
##  [9,] 18633     50077  25647
## [10,] 18633     50077  25647
```

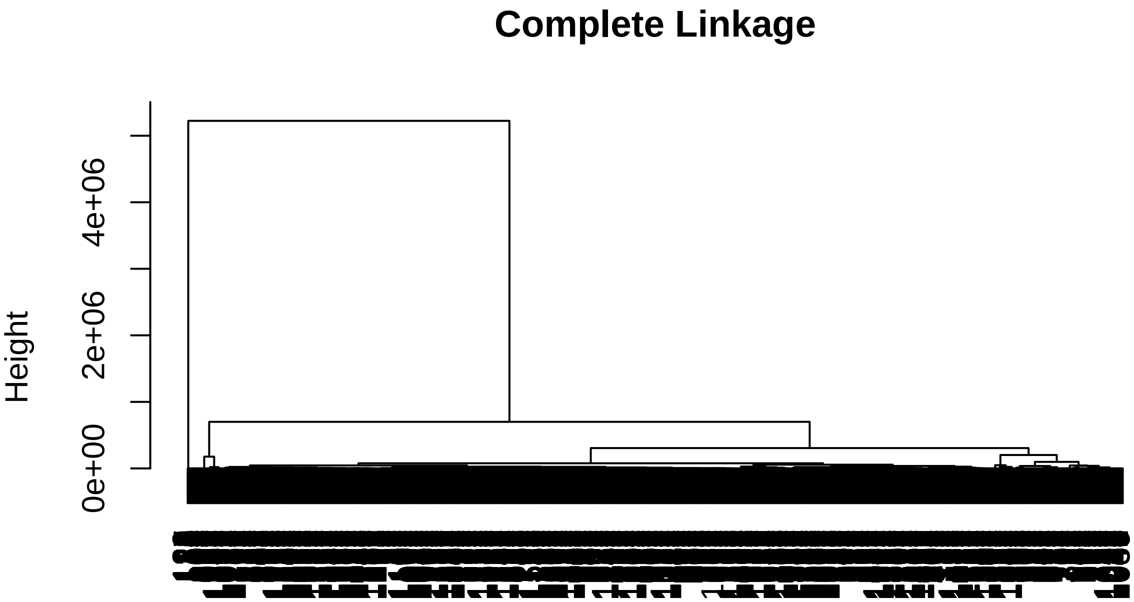
```
# clustering counties using the ward method
hc_ward <- hclust(hclust_d, method = "ward.D2")

# clustering counties using the complete method
hc_complete <- hclust(hclust_d, method = "complete")

# clustering counties using the average method
hc_average <- hclust(hclust_d, method = "average")
```

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

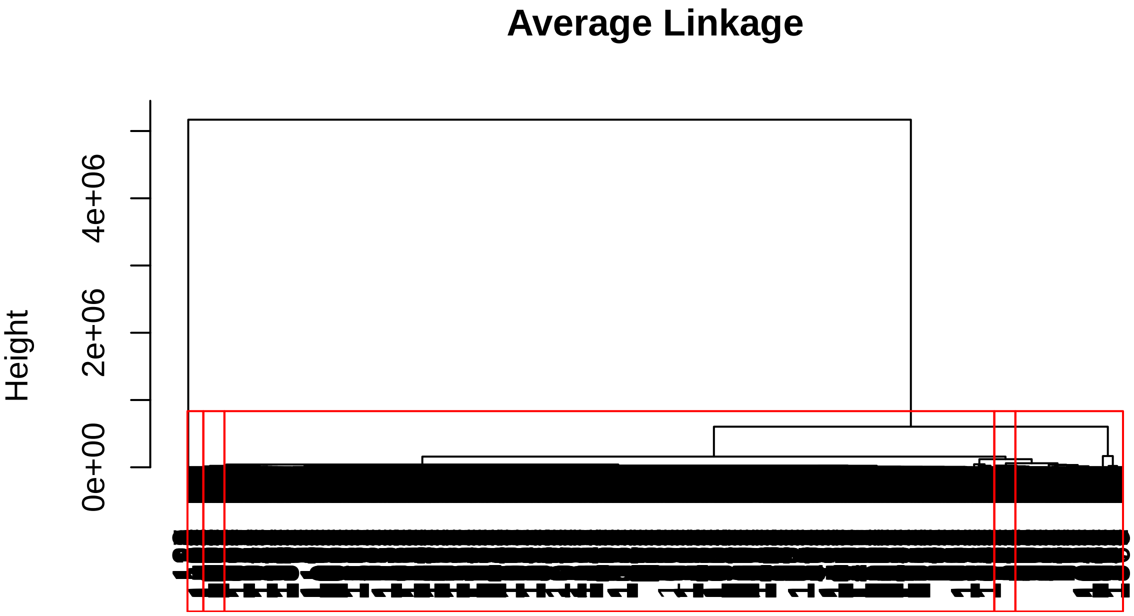
```
# ward method
plot(hc_ward, main = "Ward", xlab = "", sub = "")
```



```
# complete method
plot(hc_complete, main = "Complete Linkage", xlab = "", sub = "")
```



```
# average method
plot(hc_average, main = "Average Linkage", xlab = "", sub = "")

try <- rect.hclust(hc_ward,
            k = 5,
            border = "red")
```



We want to create five clusters based on the `hc_ward` object.

```
cutree(hc_ward, 5)
```

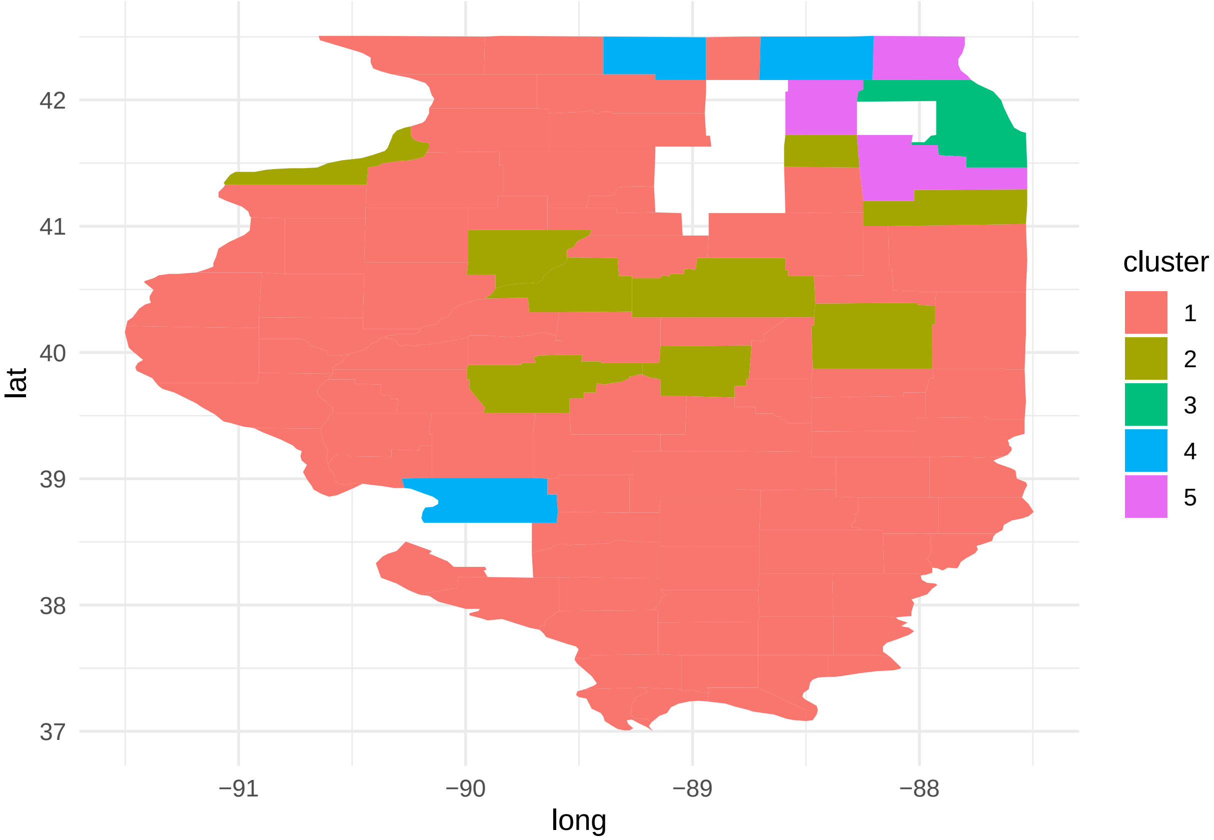
```
##    [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
##   [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##   [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3
##  [112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1
##  [149] 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
##  [186] 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 4 4 4 4 4
##  [223] 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 1 1 1 1 1 1 1 1 1
##  [260] 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
##  [297] 1 1 1 1 1 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 1 1 1 1
##  [334] 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
##  [371] 1 1 1 1 1 1 2 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
##  [408] 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
##  [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 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] 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 4 4 4
##  [556] 4 4 4 4 4 4 4 4 4 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
##  [593] 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 1 1 1
##  [630] 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
##  [667] 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 2 2 2
##  [704] 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 1 1 1 1 1 1
##  [741] 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 2 2 2 2 2 2
##  [778] 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 1 1 1 1 1 1 1 1 1 1 1
##  [815] 1 1 1 1 5 5 5 5 5 5 5 5 5 5 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 1 1 1 1 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 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
##  [926] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 5 5 5 5 5 5 5 5 5 5 5 5 5 5
##  [963] 5 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 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 1 1 1
## [1074] 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
## [1111] 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
## [1148] 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
## [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 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 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 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
## [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 1 1 1 1 1 1 1 1 1 1 1 1
## [1333] 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
## [1370] 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
## [1407] 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
## [1444] 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
## [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 1 1 1 1 1 1 1 1 1 1 1 1
## [1518] 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
## [1555] 1 1 1 1 1 1 1 1 1 1 1 1 1 5 5 5 5 5 5 5 5 5 5 5 5 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1629] 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

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`.

```
# adding cluster membership variable
acs_map <- acs_map %>%
  mutate(cluster = cutree(hc_ward, 5))

# new acs_map
head(acs_map)
```

```
##   state county                    NAME   pop hh_income income subregion
## 1    17    067 Hancock County, Illinois 18633     50077  25647   hancock
## 2    17    067 Hancock County, Illinois 18633     50077  25647   hancock
## 3    17    067 Hancock County, Illinois 18633     50077  25647   hancock
## 4    17    067 Hancock County, Illinois 18633     50077  25647   hancock
## 5    17    067 Hancock County, Illinois 18633     50077  25647   hancock
## 6    17    067 Hancock County, Illinois 18633     50077  25647   hancock
##        long      lat group order   region cluster
## 1 -91.18623 40.63417    34   573 illinois       1
## 2 -90.89976 40.63417    34   574 illinois       1
## 3 -90.91121 40.27893    34   575 illinois       1
## 4 -90.91121 40.19299    34   576 illinois       1
## 5 -91.49563 40.21018    34   577 illinois       1
## 6 -91.48990 40.25029    34   578 illinois       1
```

```
# visualize the county clusters on map
ggplot(acs_map) +geom_polygon(aes(x=long, y=lat, group=group, fill=factor(cluster))) +
  theme_minimal( ) + labs(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_all(list(~ ifelse(.==-666666666, NA, .))) %>%
  rename(pop = B01003_001E,
         hh_income = B19013_001E,
         income = B19301_001E)

acs_il_t <- na.omit(acs_il_t)
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.

```
# select pop, hh_income, income
acs_il_t_cleaned <- acs_il_t %>%
  select(pop, hh_income, income) %>%
  na.omit()

# new dataset
head(acs_il_t_cleaned)
```

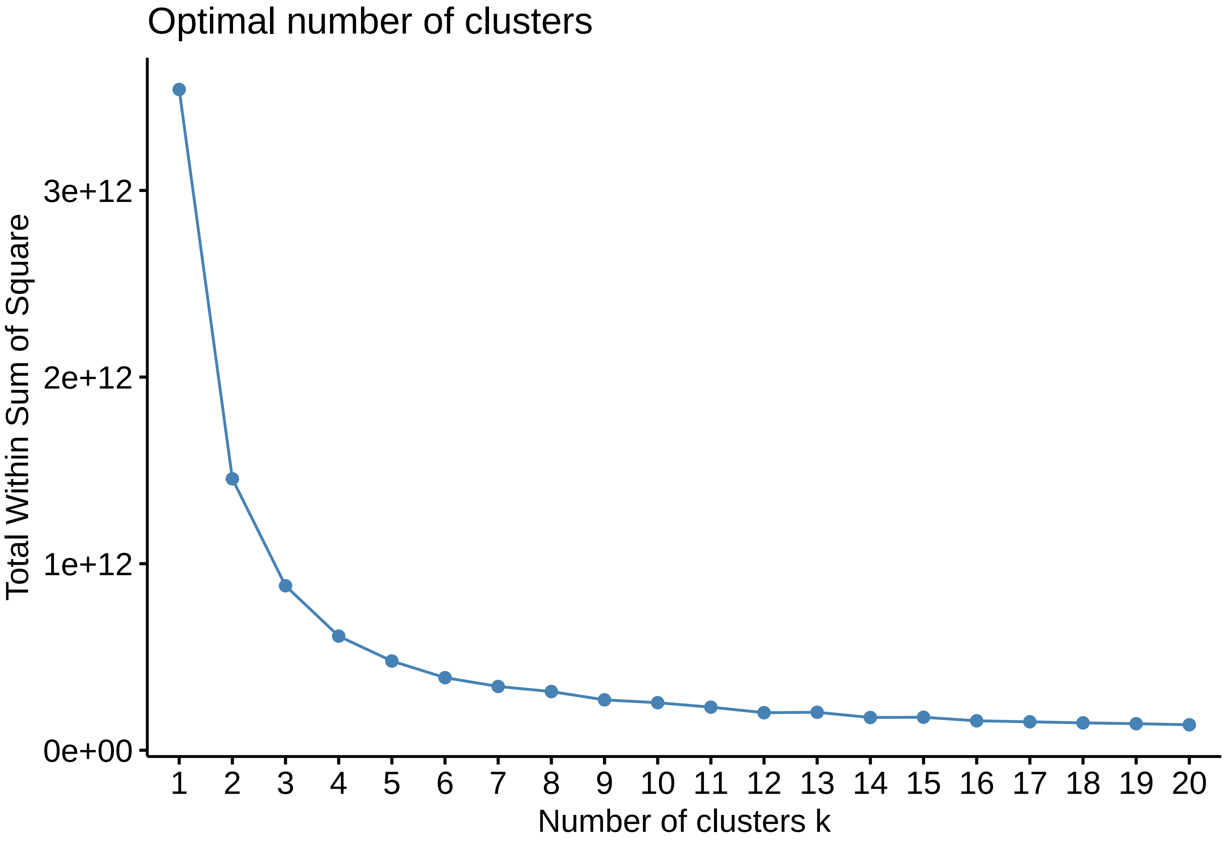
```
##    pop hh_income income
## 1 7304     56975  23750
## 2 7577     53769  25016
## 3 2684     62750  30154
## 4 2590     53583  20282
## 5 3594     40125  18347
## 6 1521     63250  31403
```

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).

- We choose k = 6 as the optimal number of clusters.

```r
# finding the optimal number of K
fviz_nbclust(acs_il_t_cleaned, kmeans, method = "wss", k.max = 20)
```

```
## 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
```



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

```r
# kmeans
km <- kmeans(acs_il_t_cleaned, 6, nstart = 20)
```

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
# adding cluster variable
acs_il_t_cluster <- acs_il_t %>%
  mutate(cluster = km$cluster)

# mean population, household income and per capita income grouped by clusters
stat <- acs_il_t_cluster %>%
  group_by(cluster) %>%
  summarize(mean_pop = mean(pop),
            mean_hh_income = mean(hh_income),
            mean_income = mean(income),
            most_frequent_county = names(which.max(table(NAME))))

# table
kable(stat, caption = "Statistics")
```

Table 1: Statistics

cluster	mean_pop	mean_hh_income	mean_income	most_frequent_county
1	4126.795	163369.92	78493.00	Census Tract 3204, Cook County, Illinois
2	5107.751	87549.29	41192.31	Census Tract 1.05, McLean County, Illinois
3	3023.996	27265.57	15527.79	Census Tract 1, Peoria County, Illinois
4	4535.430	112380.92	59990.76	Census Tract 12.04, Champaign County, Illinois
5	4357.746	64282.81	31434.40	Census Tract 1, Adams County, Illinois
6	4029.991	46117.11	23254.56	Census Tract 1, Coles County, Illinois

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.

```
# Specifying the dataset we want to use when calling the function
cluster_data <- acs_il_t %>%
    select(pop, hh_income, income)

# Writing the function to automate K Means clustering
function_kmeans <- function(data, K_clusters) {
  # Looping over each value in K, which could be a vector
  for (k in K_clusters) {
    # Performing K means clustering
  kmeans_result <- kmeans(cluster_data, centers = k, nstart = 10)
  # Adding resulting cluster membership to the original data frame
    cluster_col_name <- paste0("Cluster_K", k)
    data[[cluster_col_name]] <- kmeans_result$cluster
  }
  return(data)
}
```

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

```
# K values
Ks <- 2:10
head(function_kmeans(acs_il_t, Ks))
```

```
##   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_K2 Cluster_K3 Cluster_K4 Cluster_K5 Cluster_K6
## 1     56975  23750          1          3          1          3          3
## 2     53769  25016          1          3          1          3          5
## 3     62750  30154          1          2          1          3          3
## 4     53583  20282          1          3          1          3          5
## 5     40125  18347          1          3          3          4          5
## 6     63250  31403          1          2          1          3          3
##   Cluster_K7 Cluster_K8 Cluster_K9 Cluster_K10
## 1          1          7          1           1
## 2          6          7          1          10
## 3          1          7          6           1
## 4          6          7          1          10
## 5          6          4          7           8
## 6          1          7          6           1
```