Data Science

Mapping, Scraping, and Factors, Oh My!

Carley Dziewicki

March 6, 2019

1. Mapping County Census Data

Knowing how to access and work with census data is an important skill for any data scientist. For this question, you will need to import census data and access the information you need to construct chropleth county-level maps reflecting the proportion of the population who are older adults.

The `data` directory contains files you will need to carry out this analysis:

- `AGE03.csv` : census file containing the data you will need (and much more that you do not need!)
- `Mastdata.xls` : spreadsheet containing a list of more than 6000 census items (with descriptions) that are available at the county level.

a. Open `Mastdata.xls` outside of RStudio (in Excel or Google Sheets) and find the name of the one item which contains this very specific information: “Resident population 65 years and over, percent (July 1 - estimate) 2009”. Write down the variable name (known in Census jargon as `Item_Id`).

Answer: `AGE775209D`

b. Now use an appropriate `readr()` statement to read the entire `AGE03.csv` data file into a tibble called `census_data` .

Answer:

```
census_data <- read_csv(file = "data/AGE03.csv")
```

c. Select only the columns and rows you will need to graph the percent of residents 65 and over for each county in the mainland US, excluding DC, Hawaii, and Alaska. Use appropriate commands to select the columns and rows you need and/or exclude those you do not need. (The column `STCOU` gives state and county codes and may be useful in selecting/filtering rows.) Save the dataset under the name `county_age_data`.

Answer:

```
 county_age_data <-
  census_data %>%
  select(Areaname, STCOU, AGE775209D) %>%
  rename("percent_ge_65" = AGE775209D) %>%
  filter(!str_sub(STCOU, -3) == "000", !(str_sub(S
TCOU, end = 2) %in% c("15", "02")), !Areaname == "
District of Columbia")
```

d. To combine the census data with the mapping data will require a couple of steps. First split the `Areaname` variable into a variable called `subregion` which contains the county name and `region` which contains the state abbreviation. Call this new file `county_age_data2`.

Answer:

```
# The trick here is do be careful about the white
space
# An alternative is to make the sep = ", " which m
ight avoid the need for trimming but we'd
# be unlikely to see this the first time through.
county_age_data2 <-
  county_age_data %>%
  separate(Areaname, into = c("subregion", "region
"), sep = ",") %>%
   mutate(region = str_trim(region, side = "both")
)
```

e. In order to use the county level mapping data, we need the state name rather than just the abbreviation. Create a "crosswalk" or look-up table using the built-in variables **state.abb** and **state.name**. Your new tibble should

contain fifty rows, one for each state and two columns, one for the state 2-letter abbreviation and one for the state name. Use this `crosswalk` table to add the column of state names to your county data. Call the new data set `county_age_data3`. (Note: If we did this all in one long pipe, we would not need intermediate names but it might also get more confusing the first time through.)

Answer:

```
crosswalk <- tibble(state.abb, state.name)
county_age_data3 <-
  left_join(county_age_data2, crosswalk, by = c("r
egion"= "state.abb")) %>%
  mutate(state.name = str_to_lower(state.name),
         subregion = str_to_lower(subregion))
county_age_data3
```

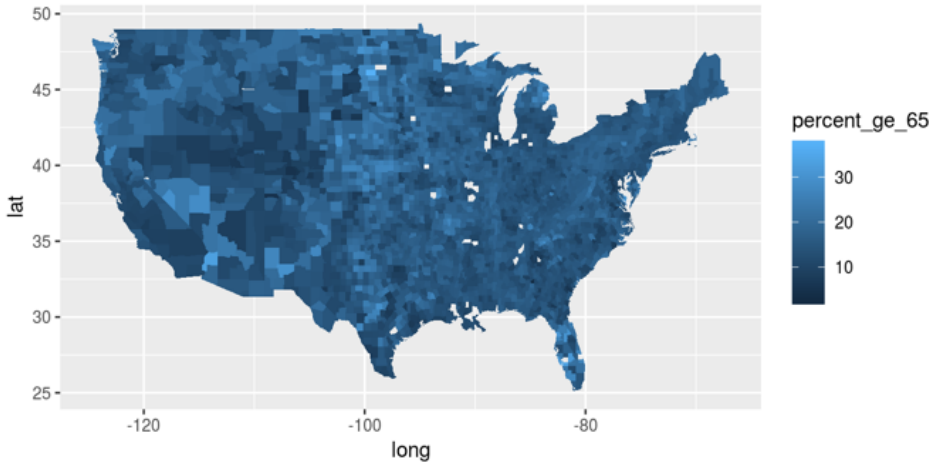
```
## # A tibble: 3,111 x 5
##    subregion region STCOU percent_ge_65 state.n
ame
##    <chr>     <chr>  <chr>         <dbl> <chr>
##  1 autauga   AL     01001          11.6 alabama
##  2 baldwin   AL     01003          17   alabama
##  3 barbour   AL     01005          13.8 alabama
##  4 bibb      AL     01007          13.5 alabama
##  5 blount    AL     01009          14.7 alabama
##  6 bullock   AL     01011          10.8 alabama
##  7 butler    AL     01013          16.2 alabama
##  8 calhoun   AL     01015          15   alabama
##  9 chambers  AL     01017          16.8 alabama
## 10 cherokee  AL     01019          18.6 alabama
## # ... with 3,101 more rows
```

f. Now join the county level data to the mapping data from the package **maps** we used in class. Call the new data set `merged_county()`. Create a county-level map of the U.S that maps the proportion of the population 65 and over to the “fill” aesthetic. You can use whatever color scheme you like. (Hint: If you get everything to work but see some holes in your map, don’t worry. Continue to the next part.)

```
main_states <- map_data("state")
all_county <- map_data("county")
merged_county <- inner_join(county_age_data3, all_
county, by = c("state.name" = "region",  "subregio
n" = "subregion"))
merged_county
```

```
## # A tibble: 86,734 x 9
##    subregion region STCOU percent_ge_65 state.n
ame  long   lat group order
##    <chr>     <chr>  <chr>         <dbl> <chr>
<dbl> <dbl> <dbl> <int>
##  1 autauga   AL     01001          11.6 alabama
-86.5  32.3     1     1
##  2 autauga   AL     01001          11.6 alabama
-86.5  32.4     1     2
##  3 autauga   AL     01001          11.6 alabama
-86.5  32.4     1     3
##  4 autauga   AL     01001          11.6 alabama
-86.6  32.4     1     4
##  5 autauga   AL     01001          11.6 alabama
-86.6  32.4     1     5
##  6 autauga   AL     01001          11.6 alabama
-86.6  32.4     1     6
##  7 autauga   AL     01001          11.6 alabama
-86.6  32.4     1     7
##  8 autauga   AL     01001          11.6 alabama
-86.6  32.4     1     8
##  9 autauga   AL     01001          11.6 alabama
-86.6  32.4     1     9
## 10 autauga   AL     01001          11.6 alabama
-86.6  32.4     1    10
## # ... with 86,724 more rows
```

```
ggplot() +
  geom_polygon(data = merged_county , aes(x = long
, y = lat, group = group, fill = percent_ge_65))
```




```
  # geom_polygon(data = main_states, aes(x = long,
y = lat, group = group),
  #              color = "black", fill = "white",
size = .05)
```

g. Unless you have been very thorough, your map likely has some holes in it because not all county names matched correctly. Use `anti_join()` to identify the counties with no matches. By looking at the original data, identify at least 2 reasons why matches failed.

Answer:

See below. Some reasons for failures include names with prefixes that have spaces between them in one file but not in the other (e.g. “dekalb” versus “de kalb”.) Another reason is differing use of punctuation so “st. clair” does not match “st clair”).

```
bad <- anti_join(county_age_data3, all_county, by
= c("state.name" = "region",  "subregion" = "subre
gion"))
bad
```

```
## # A tibble: 76 x 5
##    subregion   region STCOU percent_ge_65 state
.name
##    <chr>       <chr>  <chr>         <dbl> <chr>
##  1 dekalb      AL     01049          14.3 alaba
ma
##  2 st. clair   AL     01115          13   alaba
ma
##  3 st. francis AR     05123          12.6 arkan
sas
##  4 desoto      FL     12027          18   flori
da
##  5 st. johns   FL     12109          15.7 flori
da
##  6 st. lucie   FL     12111          20.4 flori
da
##  7 dekalb      GA     13089           8.5 georg
ia
##  8 dekalb      IL     17037           9.5 illin
ois
##  9 dupage      IL     17043          11.4 illin
ois
## 10 lasalle     IL     17099          16   illin
ois
## # ... with 66 more rows
```

h. Create a subset of the census data for only counties in Florida. Use string functions to fix the unmatched county names in Florida so that each Florida county has complete data. Use this data set to create a Florida county map with counties shaded by percent of the population 65 and over.

Answer:

```
# More to come...
str_replace(merged_county$subregion,"^(st\\.)(.*)"
, "st\\2") %>%
  str_subset("clair")
```

```
## [1] "eau claire" "eau claire" "eau claire" "eau
claire" "eau claire"
## [6] "eau claire" "eau claire" "eau claire"
```

i. Within Florida, where are high concentrations of seniors? Look at your map and comment on any patterns you see.