Introducing tidycensus

ML

1/13/2020

Load packages and keys

We will be using the `tidycensus` package extensively this month. We'll also use the `pander` package for making publication-ready tables. Let's install them.

```
install.packages(c("tidycensus", "pander"))
```

Now let's load the `tidycensus` and `pander` packages as well as the `tidyverse` package we installed last class. Turn off the warnings, errors, and messages so they don't clog up the screen (or screw up the attempts to knit the file).

```
library(tidyverse)
library(tidycensus)
library(pander)
```

You each have a Census API key. Let's install them. We'll save our keys to R's memory so you don't have to install them again.

```
census_api_key("7b0d2b954c776e88fd1df8ff9cc4c18a2ea5cdf5",
               overwrite = TRUE, install = TRUE)
# First time, reload your environment so you can use the key without restarting R.
readRenviron("~/.Renviron")
```

You can check your API key with this code:

```
Sys.getenv("CENSUS_API_KEY")
```

```
## [1] "7b0d2b954c776e88fd1df8ff9cc4c18a2ea5cdf5"
```

Introducing tidycensus

There are three main functions in the `tidycensus` package:

- `get_estimates()` gets data from the US Census Bureau Population Estimates
- `get_acs()` gets data from the American Community Survey
- `get_decennial` gets data from decennial census

Let's start with the `get_estimates` function. In this chunk, we will create a new data frame called `race_df` that pulls estimates of the national population for each race category. The `breakdown_labels = TRUE` command says we want the actual names of each race category rather than the variable names.

```
race_df <- get_estimates(geography = "us",
                         year = 2018, # the most recent year available
                         product = "characteristics",
                         breakdown = c("RACE"),
                         breakdown_labels = TRUE)
```

What are the variables? What are the values?

Let's clean up the names of the variables by making them all lowercase.

```
names(race_df) <- tolower(names(race_df))
```

There are more race categories than we need. For example, we don't want the "All races" number and we don't want any of the "in combination" numbers. Let's filter them out. We can filter out "All races" by its value. We will use the `str_detect` function to filter out any observations where the *string* "in combination" is *detected* in the `race` column.

```
race_df <- race_df %>%
  filter(race != "All races") %>% # != means "does not equal"
  filter(!str_detect(race, "in combination"))
```

What are the race category names that are left in our dataframe?

REPLACE THIS LINE WITH YOUR CODE

```
table(race_df$race)
```

```
##
##          American Indian and Alaska Native alone
##                                                1
##                                      Asian alone
##                                                1
##                                      Black alone
##                                                1
## Native Hawaiian and Other Pacific Islander alone
##                                                1
##                                Two or more races
##                                                1
##                                      White alone
##                                                1
```

The last thing to clean up is some of the very long labels in the race variable. We do this by asserting that the variable is a factor variable and changing its labels. The labels have to be in the same order as the order in which the labels are currently stored (which is alphabetical by default).

The `mutate()` function creates new variables. In this case, we are creating a new factor variable called `race_abb` which takes abbreviations based on the existing character variable called `race`.

```
race_df <- race_df %>%
  mutate(race_abb = factor(race,
                           labels = c("AIAN alone",
                                      "Asian alone",
                                      "Black alone",
                                      "NHPI alone",
                                      "Two or more races",
                                      "White alone")))
```

Describing Race

Let's create a simple table with the estimated population and the proportion of the population in each race category. We will create a new variable (with `mutate()`) called `prop` that includes the proportions.

```
race_table <- race_df %>%
  select(race_abb, value) %>% # Use select to choose columns
```

```
  mutate(prop = round((value / sum(value)),3)) # Create proportions

race_table
```

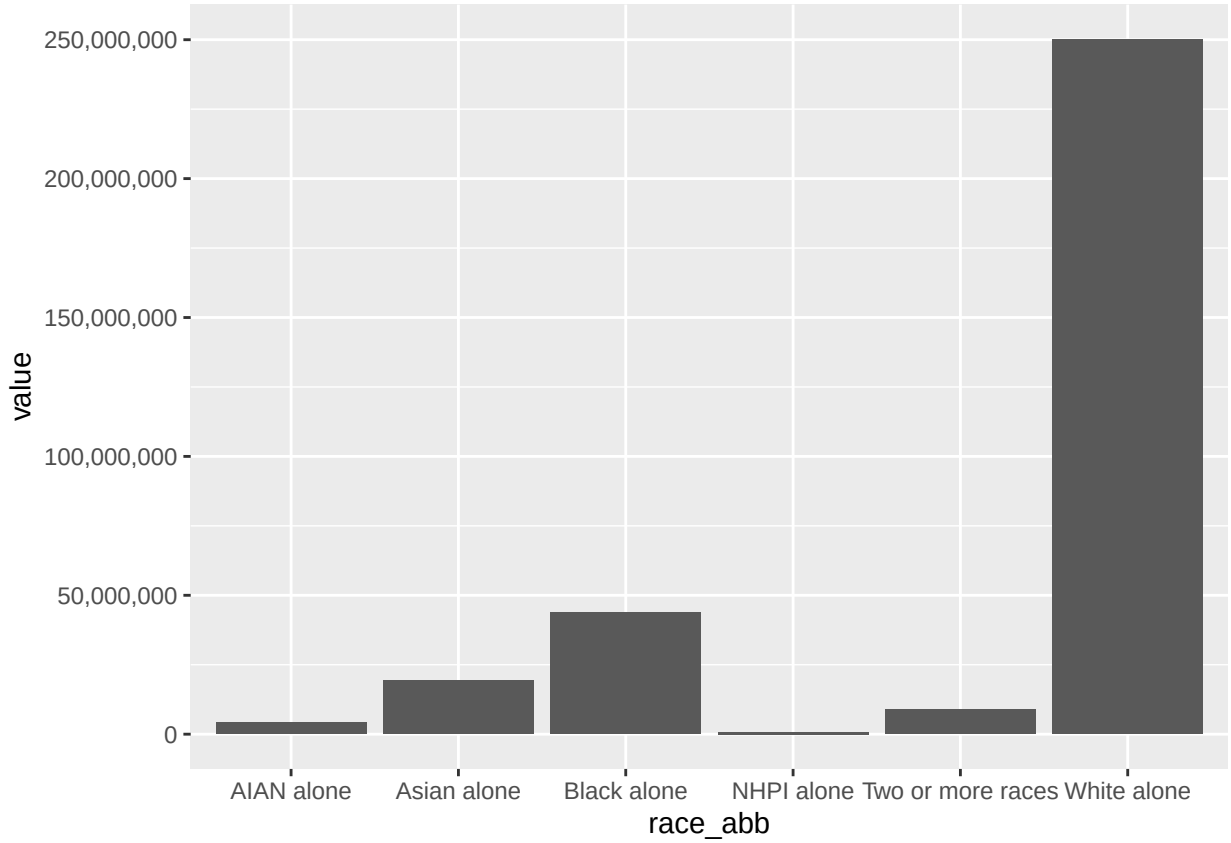
```
## # A tibble: 6 x 3
##   race_abb              value  prop
##   <fct>                 <dbl> <dbl>
## 1 White alone       250139096 0.765
## 2 Black alone        43804319 0.134
## 3 AIAN alone          4147521 0.013
## 4 Asian alone        19330600 0.059
## 5 NHPI alone           799418 0.002
## 6 Two or more races   8946480 0.027
```

Use the table we just made to create a plot showing the estimated population in each race category.

REPLACE THIS LINE WITH YOUR CODE

```
race_plot <- ggplot(race_table, aes(x = race_abb, y = value))

race_plot + geom_col() +
  scale_y_continuous(labels = scales::comma)
```



A simple geom_col works here but it is not that informative. A table is probably preferable to a figure to summarize this data. We'll clean up the table by adding commas to the population counts (with `format`) and by changing the column names (with `colnames`; there is also a `rownames` function) before feeding the

cleaned up table into `pander()`.

```
race_table$value <- format(race_table$value,
                           big.mark = ",",
                           trim = TRUE)

colnames(race_table) <- c("Race", "Estimated Count",
                          "Proportion of Population")

pander(race_table,
       caption = "Counts And Proportions By Race") # adds table title
```

Table 1: Counts And Proportions By Race

Race	Estimated Count	Proportion of Population
White alone	250,139,096	0.765
Black alone	43,804,319	0.134
AIAN alone	4,147,521	0.013
Asian alone	19,330,600	0.059
NHPI alone	799,418	0.002
Two or more races	8,946,480	0.027

Plots By Race And Age

Now we will get national estimates for each race/age combination. We ask for multiple breakdowns by listing both of them. Use `AGEGROUP` for age. Other options are `SEX` and `HISP` (for Hispanic).

```
race_age_df <- get_estimates(geography = "us",
                           product = "characteristics",
                           breakdown = c("AGEGROUP", "RACE"),
                           breakdown_labels = TRUE)
```

Clean up the data. Make the variable names lowercase. Filter out the "in combination" race categories and the "All races" category. And change the long race labels.

REPLACE THIS LINE WITH YOUR CODE

```
names(race_age_df) <- tolower(names(race_age_df))

race_age_df <- race_age_df %>%
  filter(!str_detect(race, "in combination")) %>%
  filter(race != "All races") %>%
  mutate(race_abb = factor(race,
                       labels = c("AIAN alone",
                                  "Asian alone",
                                  "Black alone",
                                  "NHPI alone",
                                  "Two or more races",
                                  "White alone")))
```

Take a look at all the available age groups.

REPLACE THIS LINE WITH YOUR CODE

```
table(race_age_df$agegroup)
```

```
##
##               All ages       Age 0 to 4 years       Age 5 to 9 years
##                      6                      6                      6
##     Age 10 to 14 years     Age 15 to 19 years     Age 20 to 24 years
##                      6                      6                      6
##     Age 25 to 29 years     Age 30 to 34 years     Age 35 to 39 years
##                      6                      6                      6
##     Age 40 to 44 years     Age 45 to 49 years     Age 50 to 54 years
##                      6                      6                      6
##     Age 55 to 59 years     Age 60 to 64 years     Age 65 to 69 years
##                      6                      6                      6
##     Age 70 to 74 years     Age 75 to 79 years     Age 80 to 84 years
##                      6                      6                      6
## Age 85 years and older         Under 18 years          5 to 13 years
##                      6                      6                      6
##         14 to 17 years         18 to 64 years         18 to 24 years
##                      6                      6                      6
##         25 to 44 years         45 to 64 years      65 years and over
##                      6                      6                      6
##      85 years and over      16 years and over      18 years and over
##                      6                      6                      6
##         15 to 44 years             Median age
##                      6                      6
```

There are a few different breakouts here. There are five year age groups, more aggregated categories (Under 18 years, 18-24, 25-44, 45-64, 65 years and over), and summary categories (all ages and median age).

Create a new dataframe called `median_df` that only has the `Median age` categories.

REPLACE THIS LINE WITH YOUR CODE

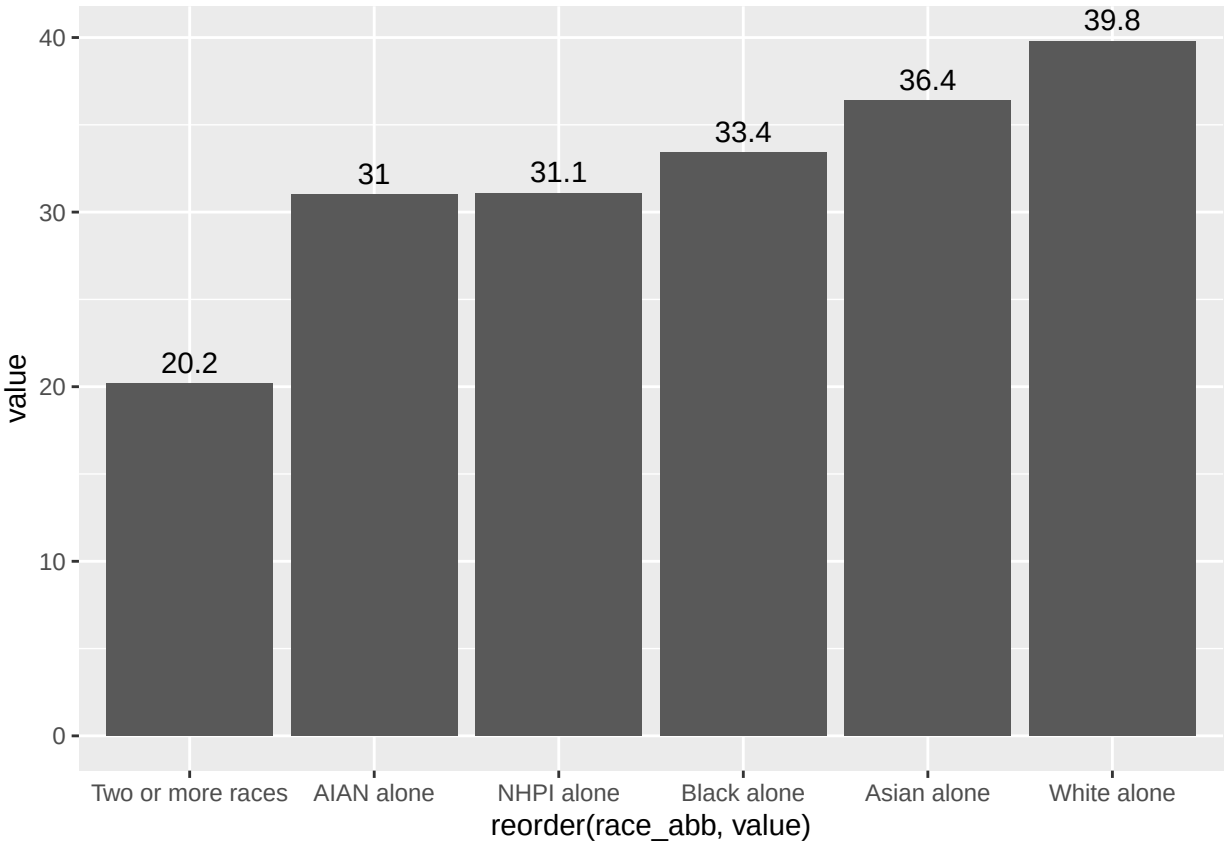
```
median_df <- race_age_df %>%
  filter(agegroup == "Median age")
```

Make a column plot showing the median age by race category. Order the races by lowest to highest median age.

REPLACE THIS LINE WITH YOUR CODE

```
median_plot <- ggplot(median_df, aes(x = reorder(race_abb, value),
                                     y = value,
                                     label = value))

median_plot + geom_col() + geom_text(vjust = -.5)
```



Using Age Categories

Let's return to the `race_age_df` data frame. We still have all the age categories in this data frame but we only want the big categories. Use filter to keep the ones we want.

```
race_agecat_df <- race_age_df %>%
  filter(race_abb != "All races") %>%
  filter(agegroup == "Under 18 years" |
           agegroup == "18 to 24 years" |
           agegroup == "25 to 44 years" |
           agegroup == "45 to 64 years" |
           agegroup == "65 years and over") %>%
  droplevels() # deletes the labels for the age categories we are not using
```

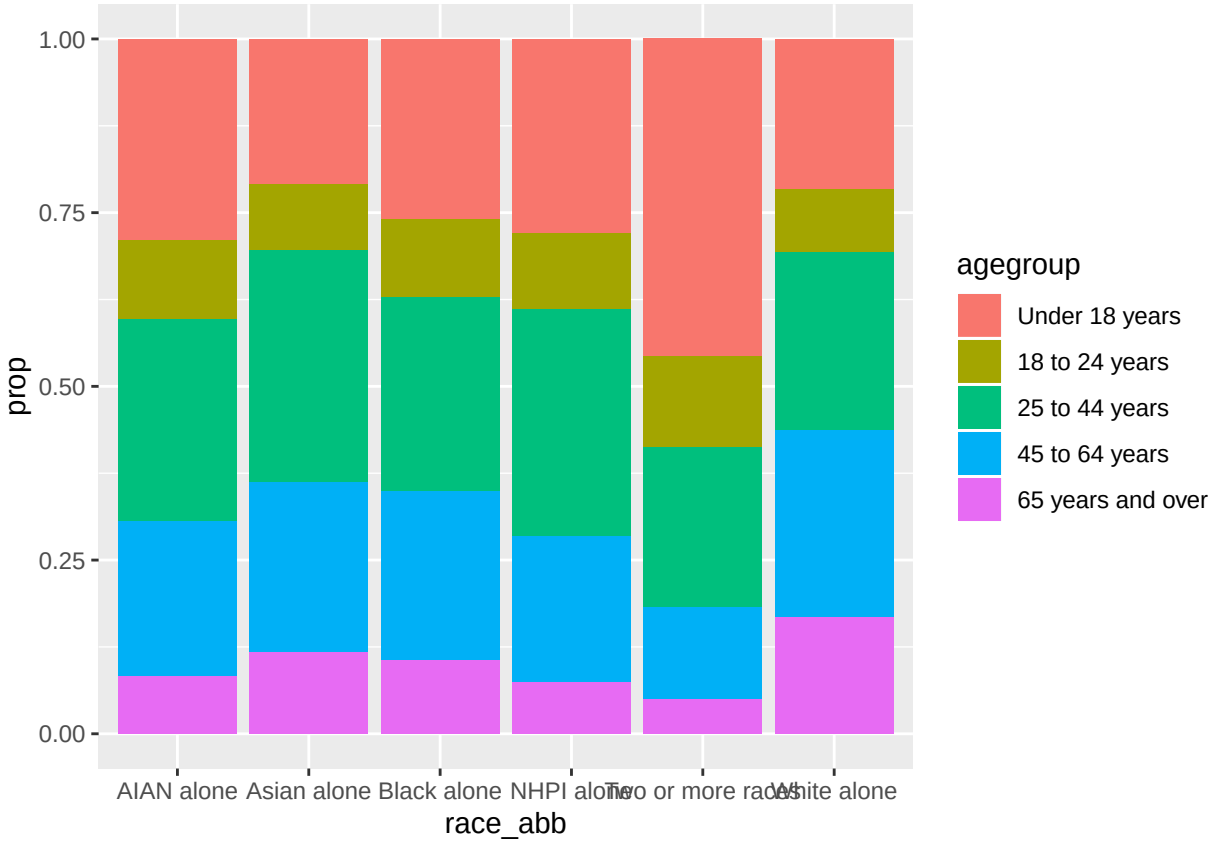
We want the proportion in each age group for each race category. Since we'll use those proportions more than once, it makes sense to create a new variable (using `mutate()`) that takes the value of the proportion. We want this for each race category so we will `group_by()` the race variable before creating the new variable.

```
race_agecat_df <- race_agecat_df %>%
  group_by(race_abb) %>%
  mutate(prop = value / sum(value))
```

Finally, create a figure that includes a column for each race category showing the proportion in each age category. We'll add the `fill =` option to the aesthetic map, and add `position = "fill"` to the geom_col() layer to get a stacked bar plot.

```
age_categories_plot <- ggplot(race_agecat_df,
                              aes(x = race_abb,
                                  y = prop,
                                  fill = agegroup))

age_categories_plot + geom_col(position = "fill")
```




```
age_categories_plot <- ggplot(race_agecat_df,
                              aes(x = race_abb,
                                  y = prop,
                                  fill = agegroup,
                                  label = round(prop, 2)))

age_categories_plot + geom_col(position = "fill") +
  geom_text(position = position_stack(vjust = 0.5))
```


1.00
0.75
0.50
0.25
0.00
prop
0.29
0.11
0.29
0.22
0.08
0.21
0.1
0.33
0.25
0.12
0.26
0.11
0.28
0.24
0.11
0.28
0.11
0.33
0.21
0.08
0.46
0.13
0.23
0.13
0.05
0.22
0.09
0.26
0.27
0.17
AIAN alone
Asian alone
Black alone
NHPI alone
Two or more races
White alone
race_abb
agegroup
Under 18 years
18 to 24 years
25 to 44 years
45 to 64 years
65 years and over