Data Appendix to "Name of your paper"

Iris Zhong

Contents

1 Raw data

1.1 Gwinnett County 2019 Referendum Dataset

Citation: Results—Gwinnett—Election Night Reporting. (n.d.). Retrieved March 11, 2020, from https://results.enr.clarityelections.com/GA/Gwinnett/94961/Web02.225391/#/
DOI: N/A
Date Downloaded: Mar 11, 2020
Filename(s): raw_data/vote_result.xls
Unit of observation: Precinct
Dates covered: Mar 19, 2019

1.1.1 To obtain a copy

Users can visit the website that displays election results at Gwinnett County at https://results.enr.clarityelections.com/GA/Gwinnett/94961/Web02.225391/#/ and choose the **Detail XLS** link at the bottom right corner.

The xls file contains three spreadsheets. I will be using the second and third sheet.

1.1.2 Importable version

Filename(s): importable_data/vote_result_importable.xlsx

The raw dataset is hard to be imported to R directly because of the following reasons. First, it has three tabs. Second, the top two rows do not contain any useful information or should be incorporated to the next row. The file uses the extension .xls, which is imcompatible with R. Therefore, an importable version of the dataset was created.

Here are the steps:

1. The original file was opened in Microsoft Excel.

2. The turnout rate for each precinct which was displayed in the second spreadsheet was moved to the third spreadsheet.
3. The first and second spreadsheets were deleted.
4. The top two rows of the third spreadsheet were removed.
5. The columns were renamed to reflect whether the vote was for or against the proposal.
6. The extension was changed from .xls to .xlsx.

1.1.3 Variable descriptions

I cannot find any information that describes the variables in this dataset. Therefore, the description below is my understanding.

- **precinct:** The name of the precinct.
- **registered_voters:** The number of registered voters in the precinct.
- **total_votes:** The number of votes received in this referendum.
- **voter_turnout:** The percentage of voters who voted in this referendum. (*reigstered_voters/total_votes*)
- **election_day_yes:** The number of people who voted yes during the election day.
- **absentee_mail_yes:** The number of people who voted yes by mailing paper ballots prior to the election day.
- **advance_in_person_1_yes:** The number of people who voted yes prior to the election day.
- **advance_in_person_2_yes:** The number of people who voted yes prior to the election day. The difference between this variable from the previous one is not clear. My speculation is they record people voting on different days before election.
- **provisional_yes:** The number of people who voted yes but had questions in their eligibility.
- **votes_yes:** The number of people who voted yes in total. (the sum of the previous five variables)
- **election_day_no:** The number of people who voted no during the election day.
- **absentee_mail_no:** The number of people who voted no by mailing paper ballots prior to the election day.
- **advance_in_person_1_no:** The number of people who voted no prior to the election day.
- **advance_in_person_2_no:** The number of people who voted no prior to the election day. The difference between this variable from the previous one is not clear. My speculation is they record people voting on different days before election.
- **provisional_no:** The number of people who voted no but had questions in their eligibility.
- **votes_no:** The number of people who voted no in total. (the sum of the previous five variables)

1.1.4 Data import code and summary

Once you've described the variables, enter an R chunk by selecting Code -> Insert Chunk, or Ctrl+Alt+I, give it a name to describe the dataset you are importing. After importing, export a dataframe summary using the command.

```
vote_result <- read_excel("importable_data/vote_result_importable.xlsx",
    col_types = c("text", "numeric", "numeric",
        "numeric", "numeric", "numeric",
        "numeric", "numeric", "numeric", "numeric",
        "numeric", "numeric", "numeric", "numeric", "numeric",
        "numeric"))
View(vote_result)
export_summary_table(dfSummary(vote_result))
```

1.2 Gwinnett County Census Data

Citation: U.S. Census Bureau. (2018). American Community Survey. https://data.census.gov/cedsci/table?d=ACS%205-Year%20Estimates%20Data%20Profiles&table=DP05&tid=ACSDP5Y2018.DP05&g=0500000US13135,13135.140000&hidePreview=true&t=Age%20and%20Sex%3AHousing%3AHousing%20Units

%3ARace%20and%20Ethnicity
DOI: N/A
Date Downloaded: Mar 19, 2020
Filename(s): N/A
Unit of observation: Census tract
Dates covered: 2018 (5-year estimates)

1.2.1 To obtain a copy

Users can obtain a copy of the dataset from an R package `tidycensus`.
Citation: Walker, K., Eberwein, K., & Herman, M. (2020). tidycensus: Load US Census Boundary and Attribute Data as "tidyverse" and 'sf'-Ready Data Frames (Version 0.9.6) [Computer software]. https://CRAN.R-project.org/package=tidycensus

Below are the steps to use `tidycensus` to obtain the data:

1. Install the package `tidycensus` in R.
2. If haven't, register to get an API key in order to download data from the pacakage. The key can be acquired at http://api.census.gov/data/key_signup.html.
3. Load the key into R with the following code:
   `census_api_key(key, install = TRUE)`
4. Load the library in R. Execute the code chunck below to get the dataframe *acs18*. *acs18* shows all of the variables present in ACS-5 2018 survey and their IDs.

```
library(tidycensus)
acs18 <- load_variables(2018, "acs5", cache = TRUE)
```

5. Search for desirable variables in *acs18* and record their IDs. The selected variables are: population, median income, median age, white population, the number of people who work, the number of people who commute by car, the number of people who commute by public transportation, the number of people who commute by subway, the number of people who go to work by bike, the number of people who walk to work, the number of people who use other transportation means, and the number of people who work at home. Subway is a subcategory of public transportation, but since it is particularly important in this project, it is also selected.

```
cbdata <- get_acs(geography = "tract",
                  variables = c(total = "B01001_001",
                                medincome = "B19013_001",
                                medage = "B01002_001",
                                white = "B01001A_001",
                                transportation_total = "B08006_001",
                                car = "B08006_002",
                                public = "B08006_008",
                                subway = "B08006_011",
                                bike = "B08006_014",
                                walk = "B08006_015",
                                other_transport = "B08006_016",
                                no_transport = "B08006_017"),
                  state = "GA",
                  county = "Gwinnett",
                  year = 2018)
View(cbdata)
```

The code above constructs a dataframe called *cbdata* by calling the function `get_acs()`, which pulls the data from the American Community Survey. Inside the function, the unit of measurement is specified by the `geography` argument. In this case, select "tract" for census tract. Put all the chosen variables in the `variables` argument. Finally, address state (GA), county (Gwinnett), and year of survey (2018).

1.2.2 Importable version (Data Wrangling)

The current dataframe requires modifications. First, each row displays one variable from one tract. However, to make tract as the unit of measurement, each row should include all the variables of one tract. Second, instead of the actual number of people who are white, the percentage of white population is more informative. Similarly, the percentage of people who go to work by certain transportation should also be calculated.

Here are the steps of data wrangling:

1. Remove the margin of error for each measurement (*moe*), because it is not useful in later analyses.

```
cbdata_moe <- cbdata %>%
  select (-moe)
```

2. Use `pivot_wider()` to transpose the data.

```
cbdata_wider <- cbdata_moe %>%
  pivot_wider(names_from = variable,
              values_from = estimate)
```

3. Calculate the percentage of white population (*white*/*total*)

```
cbdata_white <- cbdata_wider %>%
  mutate(white_pct = white / total) %>%
  select (-white)
```

4. Calculate the percentage of people using each transportation method.

```
cbdata_tidy <- cbdata_white %>%
  mutate(car_pct = car / transportation_total,
         public_pct = public / transportation_total,
         subway_pct = subway / transportation_total,
         bike_pct = bike / transportation_total,
         walk_pct = walk / transportation_total,
         other_pct = other_transport / transportation_total,
         no_pct = no_transport / transportation_total) %>%
  select(-c(car, public, bike, walk, other_transport, no_transport))
View(cbdata_tidy)
```

1.2.3 Variable descriptions

- **GEOID:** The geographic identifier of the census tract.
- **NAME:** The name of the census tract.
- **total:** The total population of the tract.
- **medage:** The median age of the population in the tract.
- **transportation_total:** The number of people who work in the tract.
- **medincome:** The median income of the population in the tract.
- **white_pct:** The percentage of white population in the tract.
- **car_pct:** The percentage of people who go to work by car, truck or van.

- **public_pct:** The percentage of people who go to work by public transportation (excluding taxi or cab).

- **subway_pct:** The percentage of people who go to work by subway or elevated.
- **bike_pct:** The percentage of people who go to work by bike.

- **walk_pct:** The percentage of people who go to work on foot.
- **other_pct:** The percentage of people who go to work by other transportation means such as taxi, cab and motorcycle.

- **no_pct:** The percentage of people who work at home (i.e. no transportation needed).

1.2.4 Data import code and summary

Once you've described the variables, enter an R chunk by selecting Code -> Insert Chunk, or Ctrl+Alt+I, give it a name to describe the dataset you are importing. After importing, export a dataframe summary using the command.

```
export_summary_table(dfSummary(dataset_name))
```

2 Data Processing and Combination

This section should include a discussion of the processing and merging steps needed to create your basic data. The code to implement these steps should be included in chunks in this section. Once the final merged data has been created, you should use the dfSummary function again to summarize the data you will be using. You should also save a file containing all the objects you will use in your final analysis to the processed_data folder.

3 Analysis Variables

This section should include a description of all the variables that are used in your final analysis. At the end of the section, you should save all of these variables in the processed_data folder of your repository.

4 Discussion of Data

This section should include a discussion of any data patterns you notice based on the summaries created in the code above.