Data Ingest & Cleaning

Michael Lewis, Aarushi Sahejpal, Katherine Prince

2023-01-08

This file will ingest and clean two datasets the American Community Survery (ACS), and COVID data. We rely on the `tidyverse` packages and `tidycensus` to import and clean our data. For the COVID data, we call the API for `covidactnow.org` (a COVID data repository) directly.

Note: you will have to use your own API keys to run this file on your machine. We use the `keyring` package to manage our API keys. Below are the links to obtain the relevant API keys

- COVID Act Now https://apidocs.covidactnow.org
- Census https://api.census.gov/data/key_signup.html

Variables of interest from ACS:

- DP03_0018 Estimate!!COMMUTING TO WORK!!Workers 16 years and over
- DP03_0065 Estimate!!INCOME AND BENEFITS (IN 2021 INFLATION ADJUSTED DOLLARS)!!Total households!!With earnings!!Mean earnings (dollars)
- DP03_0095 Estimate!!HEALTH INSURANCE COVERAGE!!Civilian noninstitutionalized population
- DP03_0042P Percent!!INDUSTRY!!Civilian employed population 16 years and over!!Educational services, and health care and social assistance

Getting data from ACS via `tidycensus`

```r
#load in Census API key
tidycensus::census_api_key(key_get("API_KEY_CENSUS"))
acs_vars <- tidycensus::load_variables(year = "2021", dataset = "acs1/profile")

#get ACS data for 2021 At  the county level
ACS <- tidycensus::get_acs(
  geography = "county",
  year = 2021,
  variables = c("DP03_0018", "DP03_0065", "DP03_0095",
                "DP03_0042P")
)
```

```r
ACS <- ACS %>%
  #breakout counties and states
  separate(NAME, into = c("county", "state"), sep = ",") %>%
  janitor::clean_names() %>%
  mutate(
    state = str_trim(state),
    county = str_trim(county),

    state = as_factor(state),
    county = as_factor(county)
  )  %>%
  #focus on DC, MD, VA
  filter(state %in% c("Maryland", "Virginia", "District of Columbia")) %>%

  #variables to labels from code
  mutate(
    variable = str_replace_all(variable, "DP03_0018",
                               "Number Commuting to Work"),
    variable = str_replace_all(variable, "DP03_0065",
                               "Mean Earnings USD"),
    variable = str_replace_all(variable, "DP03_0095",
                               "Health Insurance Coverage"),
    variable = str_replace_all(variable, "DP03_0042P",
                               "Percent Essential Workers")
  ) %>%
  pivot_wider(names_from = variable, values_from = c(estimate, moe)) %>%
  clean_names()
```

Cleaning ACS data

Getting COVID data from API `covidactnow.org`

```r
#formatting API call for each state
urlMD <- 'https://api.covidactnow.org/v2/county/MD.timeseries.json?apiKey='
urlVA <- 'https://api.covidactnow.org/v2/county/VA.timeseries.json?apiKey='
urlDC <- 'https://api.covidactnow.org/v2/county/DC.timeseries.json?apiKey='

API_URL_MD <- paste0(urlMD, key_get("COVID"))
API_URL_VA <- paste0(urlVA, key_get("COVID"))
API_URL_DC <- paste0(urlDC, key_get("COVID"))

#function to call API, clean JSON, return tibble
api_to_df <- function(apiURL) {
  api_out <- GET(apiURL)
  api_out_cleaned <- fromJSON(rawToChar(api_out$content), flatten = TRUE)
  api_out_cleaned <- as_tibble(api_out_cleaned)
  return(api_out_cleaned)
}

#running function for each state and saving
maryland_covid <- api_to_df(API_URL_MD)
```

```r
virginia_covid <- api_to_df(API_URL_VA)
dc_covid <- api_to_df(API_URL_DC)
```

Cleaning COVID data The COVID data returns a series of nested data frames for each state. We need to extract the `actualsTimeseries` df for each state for further cleaning.

Extract MD data:

```r
maryland_covid <- maryland_covid %>%
  select(county, cols = c(actualsTimeseries)) %>%
  unnest() %>%
  mutate(state = "Maryland") #adding in State name
```

```
## Warning: `cols` is now required when using unnest().
## Please use `cols = c(cols)`
```

Extract VA data:

```r
virginia_covid <- virginia_covid %>%
  select(county, cols = c(actualsTimeseries)) %>%
  unnest() %>%
  mutate(state = "Virginia") #adding in State name
```

```
## Warning: `cols` is now required when using unnest().
## Please use `cols = c(cols)`
```

Extract DC data:

```r
dc_covid <- dc_covid %>%
  select(county, cols = c(actualsTimeseries)) %>%
  unnest() %>%
  mutate(state = "District of Columbia") #adding in State name
```

```
## Warning: `cols` is now required when using unnest().
## Please use `cols = c(cols)`
```

Now we can bind together DC, MD, VA data to create on df with all the COVID data. From here we need to filter down to 2021 (to match the ACS year), and create metrics which are at the same level of analysis. In this case, we take the average cases / deaths for each county during 2021.

```r
#binding together MD, VA, DC data
dmv_covid <- bind_rows(dc_covid, maryland_covid, virginia_covid)

#filtering down to 2021 to match ACS year
dmv_covid <- dmv_covid %>%
  mutate(date = ymd(date)) %>%
  filter(date < "2021-12-31") %>%
  filter(date > "2020-12-31")

#creating metrics at the year-level (average cases/deaths) for each county
```

```
dmv_covid <- dmv_covid %>%
  group_by(county) %>%
  mutate(average_cases = mean(cases, na.rm = T),
         average_deaths = mean(deaths, na.rm = T),
         average_vaccinations_administered = mean(vaccinesAdministered, na.rm = T)) %>%
  select(county, average_cases, average_deaths, average_vaccinations_administered) %>%
  unique()
```

We can now join together the COVID data and ACS data by county.

```
combined <- left_join(dmv_covid, ACS)
```

```
## Joining, by = "county"
```

```
glimpse(combined)
```

```
## Rows: 158
## Columns: 14
## Groups: county [153]
## $ county                            <chr> "District of Columbia", "Allegany C~
## $ average_cases                     <dbl> 51847.198, 7694.165, 44252.050, 649~
## $ average_deaths                    <dbl> 1104.82967, 217.65934, 642.59615, 1~
## $ average_vaccinations_administered  <dbl> NaN, NaN, NaN, NaN, NaN, 24154.82, ~
## $ geoid                             <chr> "11001", "24001", "24003", "24005",~
## $ state                             <fct> District of Columbia, Maryland, Mar~
## $ estimate_number_commuting_to_work  <dbl> 371014, 26127, 306178, 421882, 4838~
## $ estimate_percent_essential_workers <dbl> 17.5, 26.6, 19.9, 27.0, 21.1, 22.1,~
## $ estimate_mean_earnings_usd        <dbl> 145322, 69654, 132176, 110354, 1385~
## $ estimate_health_insurance_coverage <dbl> 673717, 62927, 560769, 843113, 9105~
## $ moe_number_commuting_to_work      <dbl> 2683, 698, 2597, 3204, 1037, 430, 5~
## $ moe_percent_essential_workers     <dbl> 0.5, 2.1, 0.8, 0.6, 1.6, 2.0, 3.6, ~
## $ moe_mean_earnings_usd             <dbl> 1933, 3012, 2285, 1532, 4911, 5253,~
## $ moe_health_insurance_coverage     <dbl> 341, 563, 1755, 826, 349, 130, 593,~
```

Exporting df to a CSV file

```
write_csv(combined, "covid_econ_data.csv")
```