Problem set 4

2025-10-05

In the next problem set, we plan to explore the relationship between COVID-19 death rates and vaccination rates across US states by visually examining their correlation. This analysis will involve gathering COVID-19 related data from the CDC's API and then extensively processing it to merge the various datasets. Since the population sizes of states vary significantly, we will focus on comparing rates rather than absolute numbers. To facilitate this, we will also source population data from the US Census to accurately calculate these rates.

In this problem set we will learn how to extract and wrangle data from the data US Census and CDC APIs.

1. Get an API key from the US Census at https://api.census.gov/data/key_signup.html. You can't share this public key. But your code has to run on a TFs computer. Assume the TF will have a file in their working directory named `census-key.R` with the following one line of code:

```
census_key <- "A_CENSUS_KEY_THAT_WORKS"
```

Write a first line of code for your problem set that defines `census_key` by running the code in the file `census-key.R`.

```
## Your code here
census_key <- source("census-key.R")
```

2. The US Census API User Guide provides details on how to leverage this valuable resource. We are interested in vintage population estimates for years 2021 and 2022. From the documentation we find that the *endpoint* is:

```
#url <- "https://api.census.gov/data/2021/pep/population"
url <- "https://api.census.gov/data/2021/pep/population?get=POP_2020,POP_2021,NAME&for=state
```

Use the **httr2** package to construct the following GET request.

`https://api.census.gov/data/2021/pep/population?get=POP_2020,POP_2021,NAME&for=state:*`

Create an object called `request` of class `httr2_request` with this URL as an endpoint. Hint: Print out `request` to check that the URL matches what we want.

```
library(httr2)
request <- request(url) |>
  req_url_query(
    key = census_key$value
  )
# print(request)
```

3. Make a request to the US Census API using the `request` object. Save the response to and object named `response`. Check the response status of your request and make sure it was successful. You can learn about *status codes* here.

```
library(readr)
response <- request |>
  req_perform()
```

4. Use a function from the **httr2** package to determine the content type of your response.

```
resp_content_type(response)
```

```
[1] "application/json"
```

5. Use just one line of code and one function to extract the data into a matrix. Hints: 1) Use the `resp_body_json` function. 2) The first row of the matrix will be the variable names and this OK as we will fix in the next exercise.

```
population <- resp_body_json(response, simplify = T)
```

6. Examine the `population` matrix you just created. Notice that 1) it is not tidy, 2) the column types are not what we want, and 3) the first row is a header. Convert `population` to a tidy dataset. Remove the state ID column and change the name of the column with state names to `state_name`. Add a column with state abbreviations called `state`. Make sure you assign the abbreviations for DC and PR correctly. Hint: Use the **janitor** package to make the first row the header.

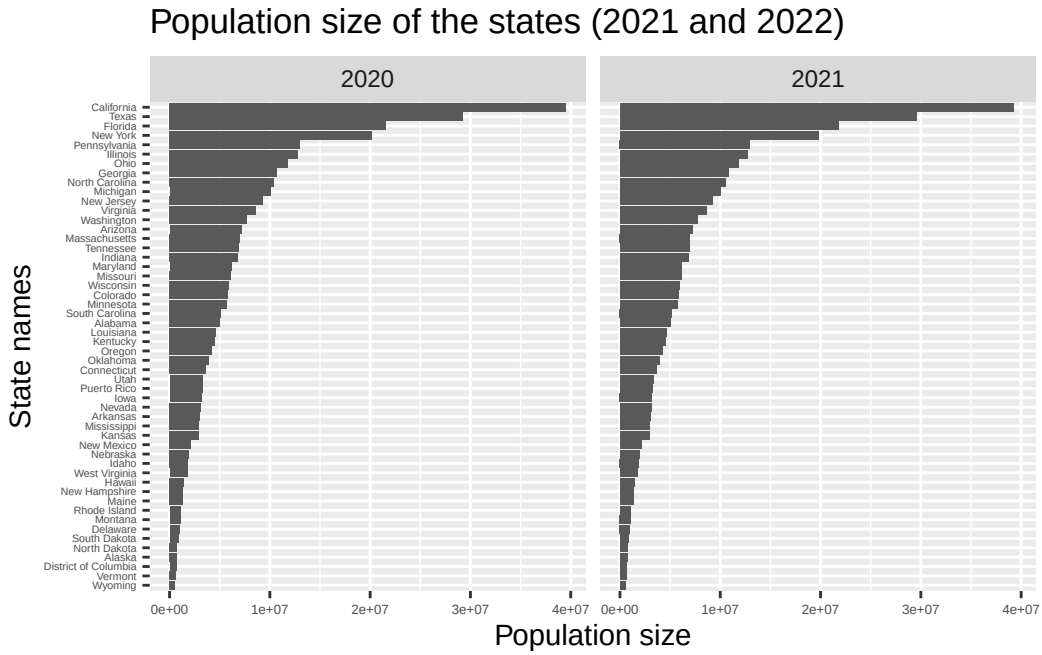
```r
library(tidyverse)
library(janitor)
#population <- population |> ## Use janitor row to names function
  # convert to tibble
  # remove stat column
  # rename state column to state_name
  # use pivot_longer to tidy
  # remove POP_ from year
  # parese all relevant columns to numeric
  # add state abbreviations using state.abb variable
  # use case_when to add abbreviations for DC and PR

population <- population |>
  row_to_names(1) |>
  as_tibble() |>
  select(-state) |>
  rename(state_name = NAME) |>
  pivot_longer(starts_with("POP_"),
               names_to = "year",
               values_to = "population") |>
  mutate(year = sub("POP_", "", year)) |>
  mutate(year = as.numeric(year),
         population = as.numeric(population)) |>
  mutate(state = case_when(state_name == "District of Columbia" ~ "DC",
                                state_name == "Puerto Rico" ~ "PR",
                                TRUE ~ state.abb[match(state_name, state.name)]
                                )
         )
head(population)
```

```
# A tibble: 6 x 4
  state_name  year population state
  <chr>      <dbl>      <dbl> <chr>
1 Oklahoma    2020    3962031 OK
2 Oklahoma    2021    3986639 OK
3 Nebraska    2020    1961455 NE
4 Nebraska    2021    1963692 NE
5 Hawaii      2020    1451911 HI
6 Hawaii      2021    1441553 HI
```

7. As a check, make a barplot of states' 2021 and 2022 populations. Show the state names in the y-axis ordered by population size. Hint: You will need to use `reorder` and use

facet_wrap.

```
# population |>
  # reorder state

  # assign aesthetic mapping

  # use geom_col to plot barplot

  # flip coordinates

  # facet by year

population |>
  ggplot(aes(x = reorder(state_name, population),
             y = population)) +
  geom_col() +
  coord_flip() +
  facet_wrap(~year) +
  labs(x = "State names",
       y = "Population size",
       title = "Population size of the states (2021 and 2022)") +
  theme(axis.text.y = element_text(size = 4),
        axis.text.x = element_text(size = 5))
```



8. The following URL:

```
url <- "https://github.com/datasciencelabs/2025/raw/refs/heads/main/data/regions.json"
```

points to a JSON file that lists the states in the 10 Public Health Service (PHS) defined by CDC. We want to add these regions to the `population` dataset. To facilitate this create a data frame called `regions` that has three columns `state_name`, `region`, `region_name`. One of the regions has a long name. Change it to something shorter.

```
library(jsonlite)
library(purrr)
url <- "https://github.com/datasciencelabs/2025/raw/refs/heads/main/data/regions.json"
# regions <- use jsonlit JSON parser
# regions <- convert list to data frame. You can use map_df in purrr package

regions_dat <- fromJSON(url)
regions <- regions_dat |>
  mutate(region = as.numeric(region),
         region_name = as.character(region_name),
         states = as.character(states)) |>
  separate_rows(states, sep = ",") |>
  rename(state_name = states) |>
```

```
  mutate(state_name = str_replace_all(state_name, "c\\(|\\)|\"", "")) |>
  mutate(state_name = str_trim(state_name)) |>
  mutate(
    region_name = case_when(
      region_name == "New York and New Jersey, Puerto Rico, Virgin Islands" ~
        "NY, NJ, PR, VI",
      TRUE ~ region_name
    ))
head(regions)
```

```
# A tibble: 6 x 3
  region region_name state_name
   <dbl> <chr>       <chr>
1      1 New England Connecticut
2      1 New England Maine
3      1 New England Massachusetts
4      1 New England New Hampshire
5      1 New England Rhode Island
6      1 New England Vermont
```

9. Add a region and region name columns to the `population` data frame.

```
population <- population |>
  left_join(regions, by = "state_name")
head(population)
```

```
# A tibble: 6 x 6
  state_name  year population state region region_name
  <chr>      <dbl>      <dbl> <chr>  <dbl> <chr>
1 Oklahoma    2020    3962031 OK         6 South Central
2 Oklahoma    2021    3986639 OK         6 South Central
3 Nebraska    2020    1961455 NE         7 Central Plains
4 Nebraska    2021    1963692 NE         7 Central Plains
5 Hawaii      2020    1451911 HI         9 Pacific
6 Hawaii      2021    1441553 HI         9 Pacific
```

10. From reading https://data.cdc.gov/ we learn the endpoint `https://data.cdc.gov/resource/pwn4-m3yp.` provides state level data from SARS-COV2 cases. Use the **httr2** tools you have learned to download this into a data frame. Is all the data there? If not, comment on why.

```
api <- "https://data.cdc.gov/resource/pwn4-m3yp.json"
cases_raw <- request(api) |>
  req_url_query() |>
  req_perform() |>
  resp_body_json(simplifyVector = T)
nrow(cases_raw)
```

```
[1] 1000
```

We see exactly 1,000 rows. We should be seeing over 52×3 rows per state.

11. The reason you see exactly 1,000 rows is because CDC has a default limit. You can change this limit by adding `$limit=10000000000` to the request. Rewrite the previous request to ensure that you receive all the data.

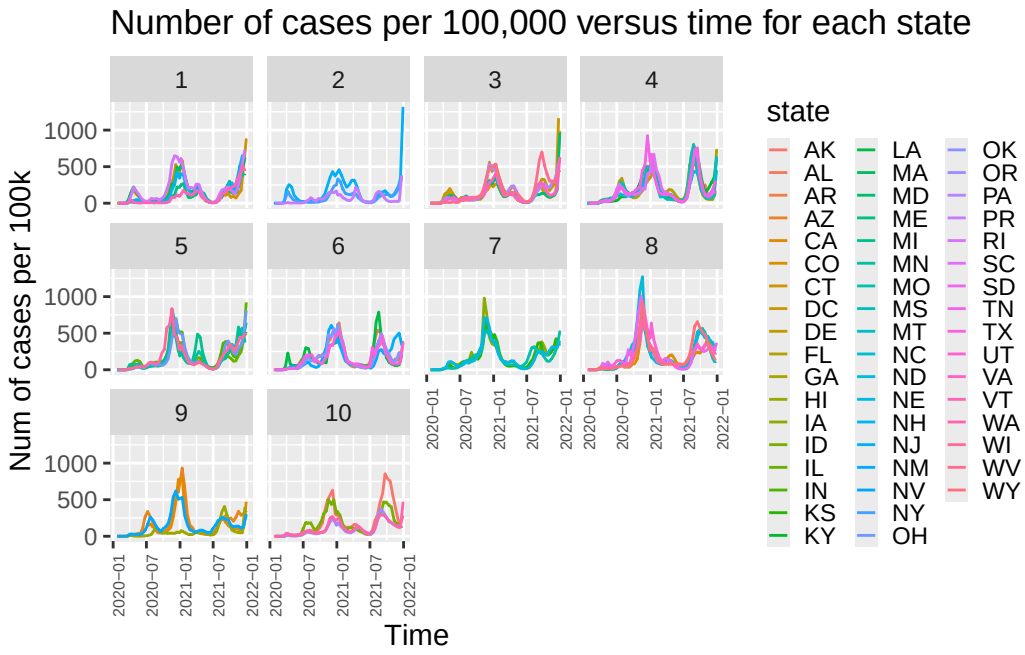
Then wrangle the resulting data frame to produce a data frame with columns `state`, `date` (should be the end date) and `cases`. Make sure the cases are numeric and the dates are in `Date` ISO-8601 format.

```
api <- "https://data.cdc.gov/resource/pwn4-m3yp.json"

cases_raw <- request(api) |>
  req_url_query(`$limit`=10000000000) |>
  req_perform() |>
  resp_body_json(simplifyVector = T) |>
  select(state, end_date, new_cases) |>
  rename(date = end_date,
         cases = new_cases) |>
  mutate(date = as.Date(date),
         cases = as.numeric(cases))
head(cases_raw)
```

```
  state       date cases
1    AZ 2023-02-22  3716
2    LA 2022-12-21  4041
3    GA 2023-02-22  5298
4    LA 2023-03-29  2203
5    LA 2023-02-01  5725
6    LA 2023-03-22  1961
```

12. For 2020 and 2021, make a time series plot of cases per 100,000 versus time for each state. Stratify the plot by region name. Make sure to label you graph appropriately.

```r
cases <- cases_raw
cases |>
  filter(year(date) == c(2020, 2021)) |>
  left_join(population, by = "state") |>
  na.omit() |>
  mutate(cases_per_100k = cases/population*100000) |>
  ggplot(aes(x= date, y = cases_per_100k, color = state)) +
  geom_line()+
  facet_wrap(~region) +
  labs(x = "Time",
       y = "Num of cases per 100k",
       title = "Number of cases per 100,000 versus time for each state") +
  theme(axis.text.x = element_text(angle = 90, size = 6),
        legend.key.size = unit(0.3, "cm"))
```



13. The dates in the `cases` dataset are stored as character strings. Use the **lubridate** package to properly parse the `date` column, then create a summary table showing the total COVID-19 cases by month and year for 2020 and 2021. The table should have columns for year, month (as month name), and total cases across all states. Order by year and month. Use the **knitr** package and `kable()` function to display the results as a formatted table.

```r
library(knitr)
cases |>
  filter(year(date) == c(2020, 2021)) |>
  mutate (year = year(date),
        month = month(date)) |>
  group_by(year, month) |>
  summarize(
    total_cases = sum(cases, na.rm = T)
  ) |>
  arrange(year, month) |>
  kable()
```

year	month	total_cases
2020	1	7
2020	2	29
2020	3	27380
2020	4	514723
2020	5	316647
2020	6	341861
2020	7	935211
2020	8	718112
2020	9	756981
2020	10	841671
2020	11	2052808
2020	12	3329321
2021	1	2863313
2021	2	1272798
2021	3	960030
2021	4	896363
2021	5	496619
2021	6	264568
2021	7	618449
2021	8	1800164
2021	9	2390091
2021	10	1193450
2021	11	1140242
2021	12	3076783

14. The following URL provides additional COVID-19 data from the CDC in JSON format:

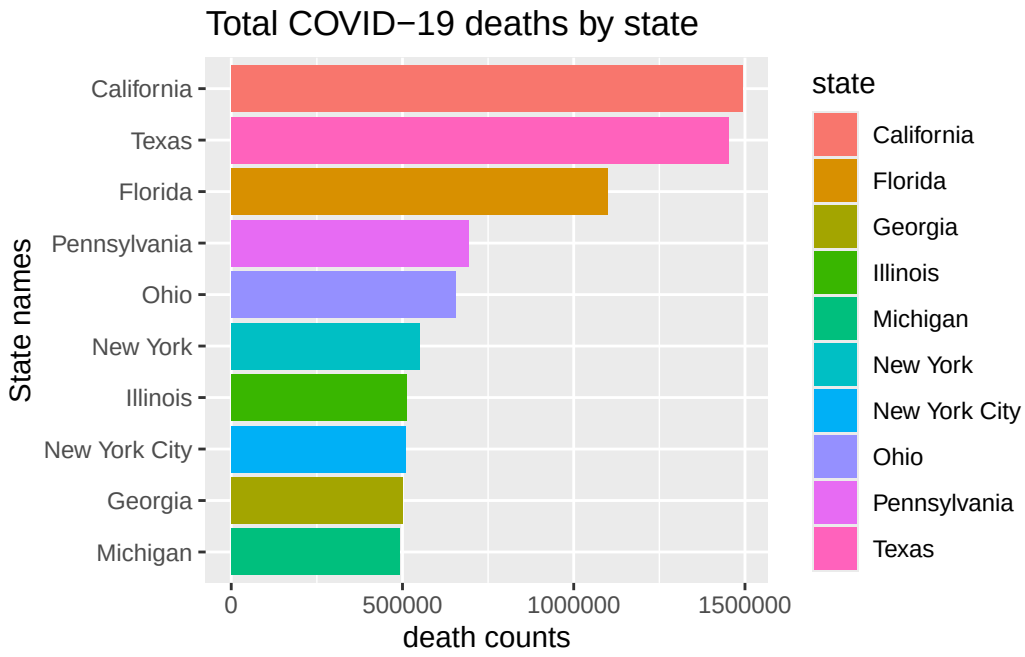
```
deaths_url <- "https://data.cdc.gov/resource/9bhg-hcku.json"
```

Use **httr2** to download COVID-19 death data from this endpoint. Make sure to remove the default limit to get all available data. Create a clean dataset called `deaths` with columns `state`, `date`, and `deaths` (renamed from the original column name). Ensure dates are in proper Date format and deaths are numeric.

```
request <- request(deaths_url) |>
  req_url_query(`$limit` = 10000000000)
deaths <- request |>
  req_perform() |>
  resp_body_json(simplifyVector = T) |>
  select(state, end_date, covid_19_deaths) |>
  rename(date = end_date,
         deaths = covid_19_deaths) |>
  mutate(date = as.Date(date),
         deaths = as.numeric(deaths)) |>
  na.omit()
```

15. Using the `deaths` dataset you created, make a bar plot showing the total COVID-19 deaths by state. Show only the top 10 states with the highest death counts. Order the bars from highest to lowest and use appropriate labels and title.

```
deaths |>
  filter(!state == "United States") |>
  group_by(state) |>
  summarize(
    total_deaths = sum(deaths)
  ) |>
  slice_max(total_deaths, n = 10) |>
  ggplot(aes(x = reorder(state, total_deaths),
             y = total_deaths, fill = state)) +
  geom_col() +
  coord_flip() +
  labs(x = "State names",
       y = "death counts",
       title = "Total COVID-19 deaths by state")
```


Total COVID−19 deaths by state
State names
California
Texas
Florida
Pennsylvania
Ohio
New York
Illinois
New York City
Georgia
Michigan
0
500000
1000000
1500000
death counts
state
California
Florida
Georgia
Illinois
Michigan
New York
New York City
Ohio
Pennsylvania
Texas