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
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 2020 and 2021. From the documentation we find that the *endpoint* is:

```
url <- "https://api.census.gov/data/2021/pep/population"
```

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:*&key=Y
```

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.

```r
library(httr2)
request <- request(url) |>
  req_url_query(
    get = "POP_2020,POP_2021,NAME",
    `for` = "state:*",
    key = census_key)
#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.

```r
response <- req_perform(request)
# check the response status to make sure it is successful
resp_status(response)
```

```
[1] 200
```

Since the result is 200, then it means that standard response for successful HTTP requests.

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

```r
# Your code here
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.

```r
#population <- resp_body_json(response) |> do.call(what = rbind)
population <- resp_body_json(response, simplifyVector = TRUE)
head(population)
```

```
     [,1]       [,2]       [,3]           [,4]   
[1,] "POP_2020" "POP_2021" "NAME"         "state"
[2,] "3962031"  "3986639"  "Oklahoma"     "40"   
[3,] "1961455"  "1963692"  "Nebraska"     "31"   
[4,] "1451911"  "1441553"  "Hawaii"       "15"   
[5,] "887099"   "895376"   "South Dakota" "46"   
[6,] "6920119"  "6975218"  "Tennessee"    "47"   
```

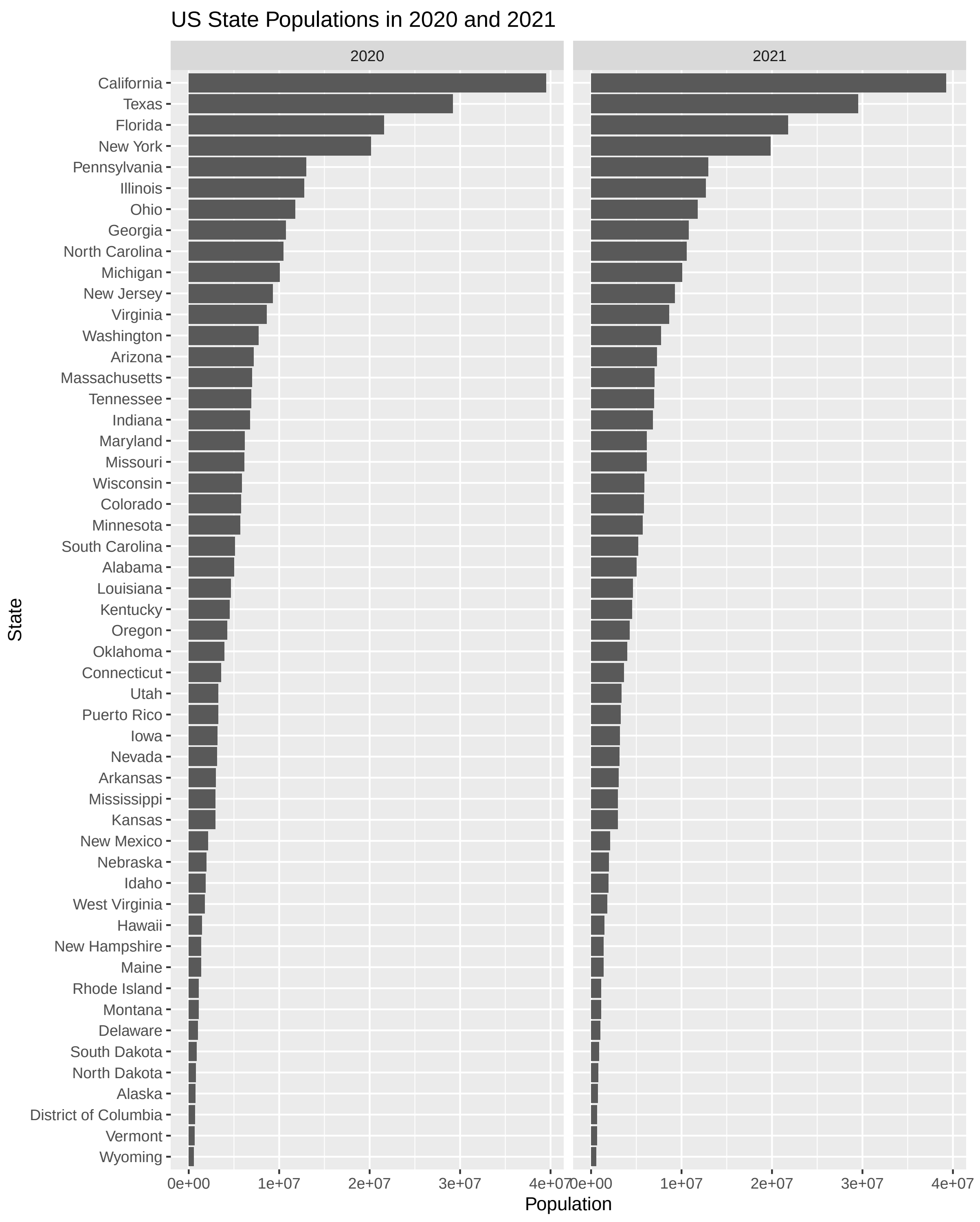
6. Examine the `population` matrix you just created. Notice that 1) it is not tidy, 2) the column types are not what we want, and 3) the first row is a header. Convert `population` to a tidy dataset. Remove the state ID column and change the name of the column with state names to `state_name`. Add a 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.

```
library(tidyverse)
library(janitor)
population <- population |> row_to_names(row_number = 1) |>  # Use janitor row to names func
  as_tibble() |>  # convert to tibble
  select(-state) |>  # remove state column
  rename(state_name = NAME) |>  # rename NAME column to state_name
  pivot_longer(
    cols = -state_name,
    names_to = "year",
    values_to = "population"
  ) |>  # use pivot_longer to tidy
  mutate(year = str_sub(year, -4), # remove POP_ from year
         # parse all relevant columns to numeric
         year = as.integer(year),
         population = as.numeric(population),
         # add state abbreviations using state.abb variable
         # use case_when to add abbreviations for DC and PR
         state = case_when(state_name == "District of Columbia" ~ "DC",
                           state_name == "Puerto Rico" ~ "PR",
                           TRUE ~ state.abb[match(state_name, state.name)]))
population
```

```
# A tibble: 104 x 4
   state_name    year population state
   <chr>        <int>      <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 South Dakota  2020     887099 SD
 8 South Dakota  2021     895376 SD
 9 Tennessee     2020    6920119 TN
10 Tennessee     2021    6975218 TN
# i 94 more rows
```

7. As a check, make a barplot of states' 2020 and 2021 populations. Show the state names in the y-axis ordered by population size. Hint: You will need to use `reorder` and use `facet_wrap`.

```r
population |>
  # reorder state by the average population across the two years
  # assign aesthetic mapping
  ggplot(aes(x = population,
             y = reorder(state_name, population))) +
  geom_col() +  # use geom_col to plot barplot
  facet_wrap(~ year) +  # facet by year
  labs(
    x = "Population",
    y = "State",
    title = "US State Populations in 2020 and 2021"
  )
```



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_raw <- request(url) |>
  req_perform() |>
  resp_check_status() |>  # check the status; stop if not 200 (error)
  resp_body_string() |>  # extract the response body as a JSON string
  fromJSON(simplifyDataFrame = FALSE)  # use jsonlit JSON parser

#class(regions)
# convert list to data frame. You can use map_df in purrr package
regions <- regions_raw |>
  map_df(function(x){
    tibble(
      state_name  = x$states,       # expand each state as a row
      region      = x$region,
      region_name = x$region_name
   )
  }) |>
  # change the long name of region to short name
  mutate(
    region_name = case_when(
      region_name == "New York and New Jersey, Puerto Rico, Virgin Islands" ~ "NY/NJ/PR/VI",
      TRUE ~ region_name
    )
  ) |>
  # keep rows where state_name is one of the 50 U.S. states, District of Columbia, or Puerto
  filter(state_name %in% c(state.name, "District of Columbia", "Puerto Rico"))
#nrow(regions)
```

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

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

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_perform() |>
  resp_check_status() |>  # check the API response status
  resp_body_json(simplifyDataFrame = TRUE)
#cases_raw
nrow(cases_raw)
```

```
[1] 1000
```

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

Comments:

No, it doesn't show all the data there. Because API often limit how much we can download. If we want to get more rows, we can use `$limit` parameters.

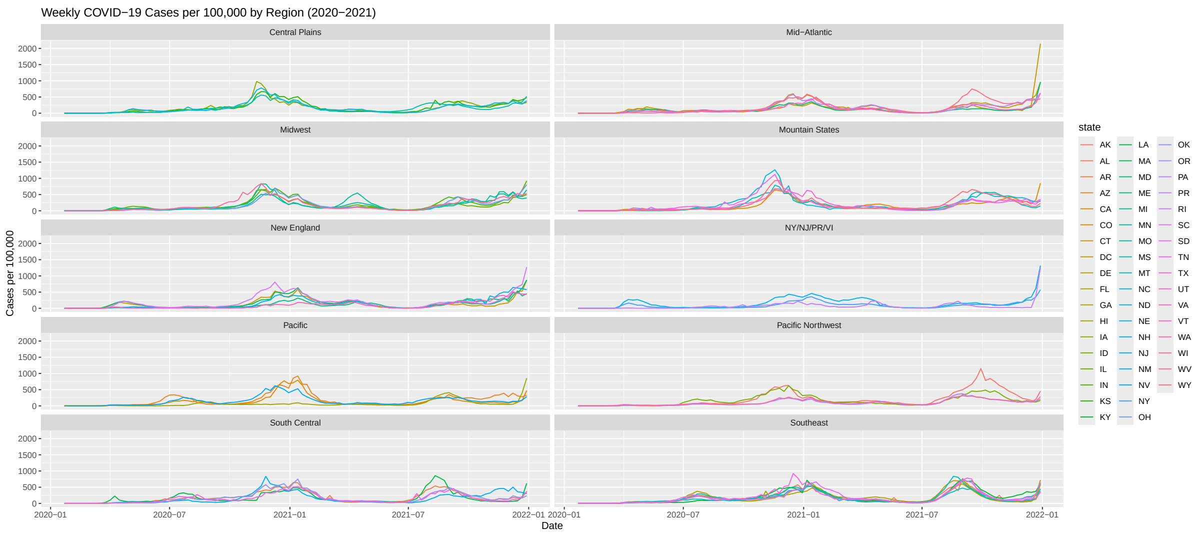
11. The reason you see exactly 1,000 rows is because CDC has a default limit. You can change this limit by adding `$limit=10000000000` to the request. Rewrite the previous request to ensure that you receive all the data. Then wrangle the resulting data frame to produce a data frame with columns `state`, `date` (should be the end date) and `cases` (Use new cases (daily new cases), not cumulative totals). 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 <- request(api) |>
  req_url_query(`$limit` = "10000000000") |>
  req_perform() |>
  resp_check_status() |>  # check the API response status
  resp_body_json(simplifyDataFrame = TRUE) |>
  select(state, end_date, new_cases) |>  # select columns `state`, `end_date`, and `new_case
  rename(date = end_date,
         cases = new_cases) |>  # rename column names
  mutate(cases = as.numeric(cases),
         date = as.Date(date))  # make sure the cases are numeric and the dates are in the `I
```

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 |>
  filter(year(date) %in% c(2020, 2021)) |>  # keep only 2020 & 2021
  mutate(year = year(date)) |>  # create a new column `year` by extracting year from `date`
  left_join(population, by = c("state", "year")) |>  # join with `population` on both `state
  mutate(cases_per100k = (cases / population) * 100000) |>  # compute cases per 100,000 popul
  select(state, region_name, date, cases_per100k) |>  # select columns `state`, `region_name
  filter(!is.na(state), !is.na(region_name), !is.na(date), !is.na(cases_per100k)) |>  # remov

  ggplot(aes(x = date, y = cases_per100k, color = state)) +  # x-axis=date, y-axis=cases_per1
  geom_line() +  # draw lines for each state's time series
  facet_wrap(~ region_name, ncol=2) +  # create separate panels for each region
  labs(
    title = "Weekly COVID-19 Cases per 100,000 by Region (2020-2021)",
    x = "Date",
    y = "Cases per 100,000"
  )
```



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(lubridate)
library(knitr)
```

```r
cases |>
  mutate(date = ymd(date)) |>  # properly parse the `date` column
  filter(year(date) %in% c(2020, 2021)) |>  # keep only 2020 & 2021
  mutate(year = year(date), month = month(date, label=TRUE, abbr=FALSE)) |>  # full month nam
  select(year, month, cases) |>  # select columns `year`, `month`, and `cases`
  group_by(year, month) |>  # group by `year` and `month`
  # calculate the total number of cases for each year-month group (ignoring NAs)
  summarise(total_cases = sum(cases, na.rm = TRUE),
            .groups = "drop") |>  # drop all grouping
  arrange(year, month) |>  # order by `year` and `month`
  kable(caption = "Total COVID-19 Cases by Year and Month (2020-2021)")
```

Table 1: Total COVID-19 Cases by Year and Month (2020–2021)

year	month	total_cases
2020	January	11
2020	February	68
2020	March	68245
2020	April	974032
2020	May	650943
2020	June	654904
2020	July	1989512
2020	August	1461283
2020	September	1415438
2020	October	1628598
2020	November	3932646
2020	December	7027128
2021	January	5808063
2021	February	2667511
2021	March	2068441
2021	April	1773591
2021	May	972915
2021	June	493635
2021	July	1137440
2021	August	3572562
2021	September	5027537
2021	October	2356302
2021	November	2322814
2021	December	5615644

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

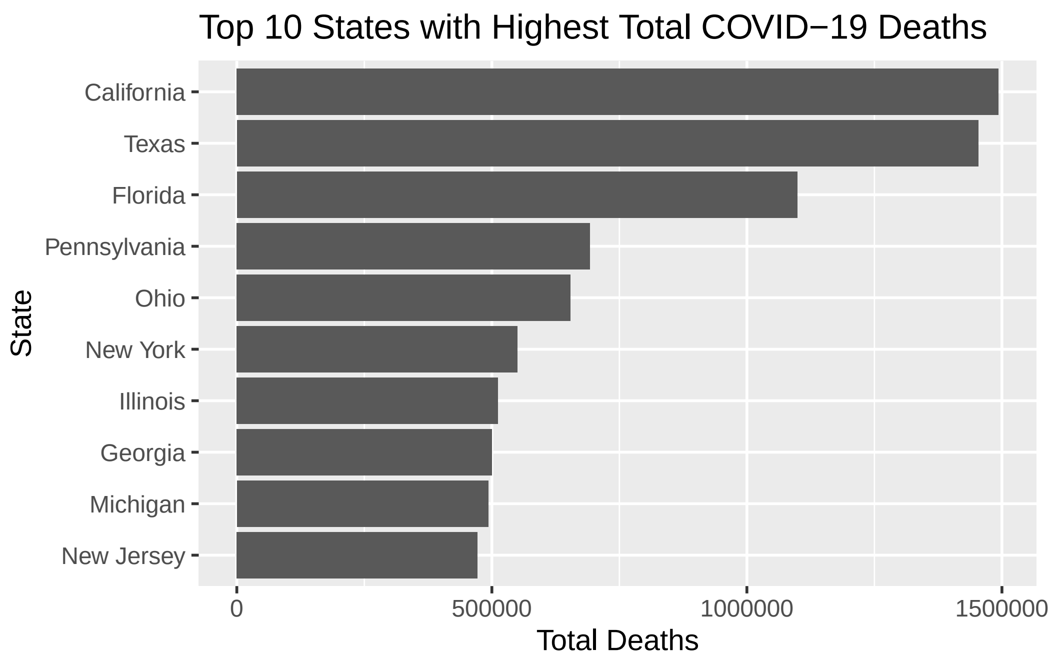
```r
deaths_url <- "https://data.cdc.gov/resource/9bhg-hcku.json"
```

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.

```r
# Your code here
deaths <- request(deaths_url) |>
  req_url_query(`$limit` = "1000000000") |>
  req_perform() |>
  resp_check_status() |>  # check the API response status
  resp_body_json(simplifyDataFrame = TRUE) |>
  select(state, end_date, covid_19_deaths) |>  # select columns state, end_date, covid_19_dea
  rename(date = end_date, deaths = covid_19_deaths) |>  # rename columns
  # ensure dates are in proper Date format and deaths are numeric
  mutate(date = as.Date(date), deaths = as.numeric(deaths)) |>
  # keep rows where state is one of the 50 U.S. states, District of Columbia, or Puerto Rico
  filter(state %in% c(state.name, "District of Columbia", "Puerto Rico")) |>
  filter(!is.na(date), !is.na(state), !is.na(deaths))
#deaths
```

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.

```r
# Your code here
deaths |>
  group_by(state) |>  # group by state
  summarise(total_deaths = sum(deaths, na.rm = TRUE),  # compute the total number of cases fo
            .groups = "drop") |>  # drop all groupings
  slice_max(total_deaths, n = 10) |>  # select the top 10 states with the highest death coun
  # reorder states so bars are sorted by death counts
  ggplot(aes(x = total_deaths, y = reorder(state, total_deaths))) +
  geom_col() +  # draw a bar chart (bars represent death counts)
  labs(
    title = "Top 10 States with Highest Total COVID-19 Deaths",
    y = "State",
    x = "Total Deaths"
  )
```


Top 10 States with Highest Total COVID−19 Deaths
State
California
Texas
Florida
Pennsylvania
Ohio
New York
Illinois
Georgia
Michigan
New Jersey
0
500000
1000000
1500000
Total Deaths