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`.

```r
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:

```r
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 <- paste("https://api.census.gov/data/2021/pep/population?get=POP_2020,POP_2021,NAME&
request
```

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

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 <- request(request) |>
  req_perform()
resp_status(response)
```

```
[1] 200
```

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

```r
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 <- response |> resp_body_json(simplifyVector = TRUE)

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"
 [7,] "3114071"  "3143991"  "Nevada"               "32"
 [8,] "2117566"  "2115877"  "New Mexico"           "35"
 [9,] "3188669"  "3193079"  "Iowa"                 "19"
[10,] "2935880"  "2934582"  "Kansas"               "20"
[11,] "690093"   "670050"   "District of Columbia" "11"
[12,] "29217653" "29527941" "Texas"                "48"
[13,] "6154481"  "6168187"  "Missouri"             "29"
[14,] "3012232"  "3025891"  "Arkansas"             "05"
[15,] "10067664" "10050811" "Michigan"             "26"
[16,] "1377848"  "1388992"  "New Hampshire"        "33"
[17,] "10457177" "10551162" "North Carolina"       "37"
[18,] "11790587" "11780017" "Ohio"                 "39"
[19,] "5130729"  "5190705"  "South Carolina"       "45"
[20,] "577267"   "578803"   "Wyoming"              "56"
[21,] "39499738" "39237836" "California"           "06"
[22,] "778962"   "774948"   "North Dakota"         "38"
[23,] "4651203"  "4624047"  "Louisiana"            "22"
[24,] "6172679"  "6165129"  "Maryland"             "24"
[25,] "991886"   "1003384"  "Delaware"             "10"
[26,] "12989625" "12964056" "Pennsylvania"         "42"
[27,] "10725800" "10799566" "Georgia"              "13"
[28,] "4241544"  "4246155"  "Oregon"               "41"
[29,] "5707165"  "5707390"  "Minnesota"            "27"
[30,] "5784308"  "5812069"  "Colorado"             "08"
[31,] "9279743"  "9267130"  "New Jersey"           "34"
[32,] "4503958"  "4509394"  "Kentucky"             "21"
[33,] "7718785"  "7738692"  "Washington"           "53"
[34,] "1362280"  "1372247"  "Maine"                "23"
[35,] "642495"   "645570"   "Vermont"              "50"
[36,] "1847772"  "1900923"  "Idaho"                "16"
[37,] "6785644"  "6805985"  "Indiana"              "18"
[38,] "1086193"  "1104271"  "Montana"              "30"
[39,] "20154933" "19835913" "New York"             "36"
[40,] "3281538"  "3263584"  "Puerto Rico"          "72"
[41,] "3600260"  "3605597"  "Connecticut"          "09"
[42,] "21569932" "21781128" "Florida"              "12"
```

```
[43,] "8632044"  "8642274"  "Virginia"               "51"
[44,] "7022220"  "6984723"  "Massachusetts"          "25"
[45,] "12785245" "12671469" "Illinois"               "17"
[46,] "2956870"  "2949965"  "Mississippi"            "28"
[47,] "7177986"  "7276316"  "Arizona"                "04"
[48,] "3281684"  "3337975"  "Utah"                   "49"
[49,] "5892323"  "5895908"  "Wisconsin"              "55"
[50,] "5024803"  "5039877"  "Alabama"                "01"
[51,] "1789798"  "1782959"  "West Virginia"          "54"
[52,] "1096229"  "1095610"  "Rhode Island"           "44"
[53,] "732441"   "732673"   "Alaska"                 "02"
```

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.

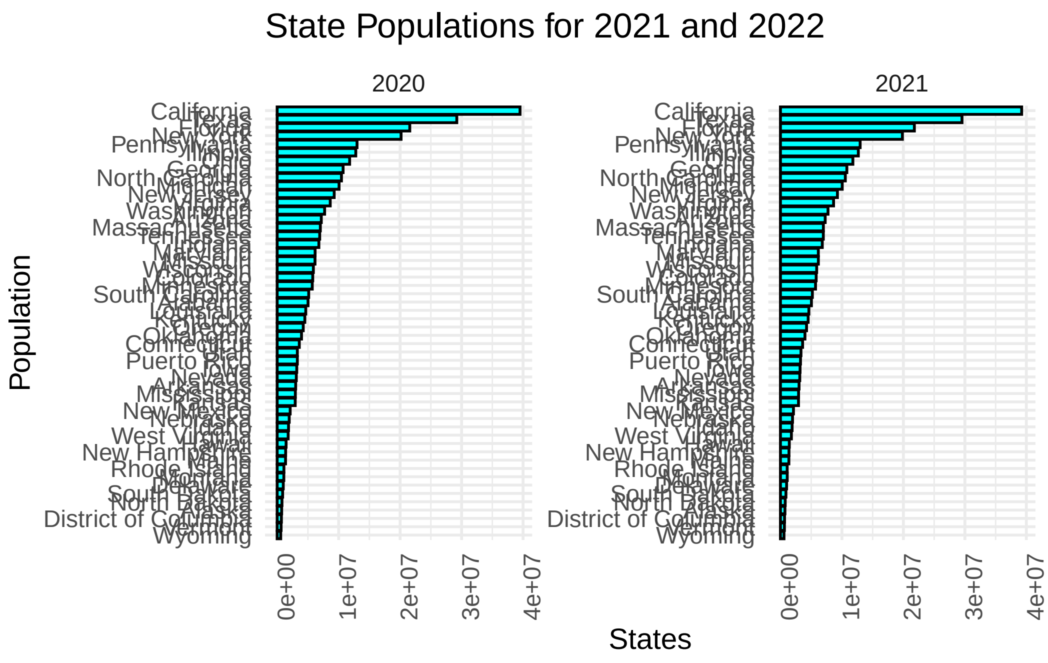
```r
library(tidyverse)
library(janitor)
#population <- population |> ## Use janitor row to names function
  # convert to tibble
  # remove stat column
  # rename state column to state_name
  # use pivot_longer to tidy
  # remove POP_ from year
  # parese all relevant colunns to numeric
  # add state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


population <- population |>
  row_to_names(row_number=1) |>
  as.tibble(population) |>
  select(-state) |> rename(state_name = NAME) |>
  pivot_longer(cols = starts_with("POP_"), names_to = "year", values_to = "population") |>
  mutate(year = str_remove(year, "POP_"), 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)]
             ))
```

```r
print(population)
```

```
# A tibble: 104 x 4
   state_name   year  population state
   <chr>        <chr>      <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' 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`.

```r
# population |>
  # reorder state
  # assign aesthetic mapping
  # use geom_col to plot barplot
  # flip coordinates
  # facet by year

population.7 <- population |>
  mutate(state_name = reorder(state_name, population)) |>
  ggplot(aes(x = state_name, y = population)) +
  geom_col(fill = "cyan", color = "black") +
  coord_flip() +
  facet_wrap(~year, scales = "free_y") +
  labs(title = "State Populations for 2021 and 2022",
       x = "Population",
       y = "States") +
  theme_minimal()  +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

population.7
```



8. The following URL:

```r
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 two columns `state_name`, `region`, `region_name`. One of the regions has a long name. Change it to something shorter.

```r
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 <- fromJSON(url, simplifyDataFrame = FALSE)

regions <- map_df(regions, function(x)
  data.frame(region=x$region, region_name = x$region_name, state_name = x$states))

regions$region_name <- gsub("New York and New Jersey, Puerto Rico, Virgin Islands", "NY & NJ

regions
```

```
   region       region_name                                   state_name
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
7       2 NY & NJ & PR & VI                                   New Jersey
8       2 NY & NJ & PR & VI                                     New York
9       2 NY & NJ & PR & VI                                  Puerto Rico
10      2 NY & NJ & PR & VI                               Virgin Islands
11      3      Mid-Atlantic                                     Delaware
12      3      Mid-Atlantic                         District of Columbia
13      3      Mid-Atlantic                                     Maryland
14      3      Mid-Atlantic                                 Pennsylvania
15      3      Mid-Atlantic                                     Virginia
16      3      Mid-Atlantic                                West Virginia
17      4         Southeast                                      Alabama
18      4         Southeast                                      Florida
19      4         Southeast                                      Georgia
20      4         Southeast                                     Kentucky
21      4         Southeast                                  Mississippi
22      4         Southeast                               North Carolina
23      4         Southeast                               South Carolina
24      4         Southeast                                    Tennessee
25      5           Midwest                                     Illinois
26      5           Midwest                                      Indiana
27      5           Midwest                                     Michigan
28      5           Midwest                                    Minnesota
29      5           Midwest                                         Ohio
30      5           Midwest                                    Wisconsin
31      6     South Central                                     Arkansas
32      6     South Central                                    Louisiana
33      6     South Central                                   New Mexico
34      6     South Central                                     Oklahoma
35      6     South Central                                        Texas
36      7    Central Plains                                         Iowa
37      7    Central Plains                                       Kansas
38      7    Central Plains                                     Missouri
39      7    Central Plains                                     Nebraska
40      8   Mountain States                                     Colorado
41      8   Mountain States                                      Montana
42      8   Mountain States                                 North Dakota
```

```
43       8   Mountain States                                      South Dakota
44       8   Mountain States                                              Utah
45       8   Mountain States                                           Wyoming
46       9           Pacific                                           Arizona
47       9           Pacific                                        California
48       9           Pacific                                            Hawaii
49       9           Pacific                                            Nevada
50       9           Pacific                                    American Samoa
51       9           Pacific Commonwealth of the Northern Mariana Islands
52       9           Pacific               Federated States of Micronesia
53       9           Pacific                                              Guam
54       9           Pacific                                  Marshall Islands
55       9           Pacific                                 Republic of Palau
56      10 Pacific Northwest                                            Alaska
57      10 Pacific Northwest                                             Idaho
58      10 Pacific Northwest                                            Oregon
59      10 Pacific Northwest                                        Washington
```

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

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

population
```

```
# A tibble: 104 x 6
   state_name   year  population state region region_name
   <chr>        <chr>      <dbl> <chr>  <int> <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
 7 South Dakota 2020      887099 SD         8 Mountain States
 8 South Dakota 2021      895376 SD         8 Mountain States
 9 Tennessee    2020     6920119 TN         4 Southeast
10 Tennessee    2021     6975218 TN         4 Southeast
# i 94 more rows
```

10. From reading https://data.cdc.gov/ we learn the endpoint `https://data.cdc.gov/resource/pwn4-m3yp.json` 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.

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

response <- request(api) |>
  req_perform()

cases_raw <- response |>
  resp_body_json(simplifyVector = TRUE) |>
  as_tibble()

cases_raw
```

```
# A tibble: 1,000 x 10
   date_updated         state start_date end_date tot_cases new_cases tot_deaths
   <chr>                <chr> <chr>      <chr>    <chr>     <chr>     <chr>
 1 2023-02-23T00:00:00~ AZ    2023-02-1~ 2023-02~ 2434631.0 3716.0    33042.0
 2 2022-12-22T00:00:00~ LA    2022-12-1~ 2022-12~ 1507707.0 4041.0    18345.0
 3 2023-02-23T00:00:00~ GA    2023-02-1~ 2023-02~ 3061141.0 5298.0    42324.0
 4 2023-03-30T00:00:00~ LA    2023-03-2~ 2023-03~ 1588259.0 2203.0    18858.0
 5 2023-02-02T00:00:00~ LA    2023-01-2~ 2023-02~ 1548508.0 5725.0    18572.0
 6 2023-03-23T00:00:00~ LA    2023-03-1~ 2023-03~ 1580709.0 1961.0    18835.0
 7 2023-04-27T00:00:00~ LA    2023-04-2~ 2023-04~ 1597070.0 1884.0    18937.0
 8 2023-03-16T00:00:00~ NV    2023-03-0~ 2023-03~ 891702.0  1233.0    11937.0
 9 2023-05-11T00:00:00~ FL    2023-05-0~ 2023-05~ 7572282.0 6937.0    88248.0
10 2022-10-27T00:00:00~ NYC   2022-10-2~ 2022-10~ 2928439.0 14590.0   42863.0
# i 990 more rows
# i 3 more variables: new_deaths <chr>, new_historic_cases <chr>,
#   new_historic_deaths <chr>
```

No, the data are not all there. Since the default limit is 1000 rows, we need to change the limit to a larger number.

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.

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

response <- request(api) |>
  req_url_query(`$limit` = 10000000000) |>
  req_perform()

cases_raw <- response |>
  resp_body_json(simplifyVector = TRUE) |>
  as_tibble()

#cases_raw

cases <- cases_raw |>
  select(state, end_date, new_cases) |>
  rename(date = end_date, cases = new_cases) |>
  mutate(
    cases = as.numeric(cases),
    date = as.Date(date))

cases
```

```
# A tibble: 10,380 x 3
   state date       cases
   <chr> <date>     <dbl>
 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
 7 LA    2023-04-26  1884
 8 NV    2023-03-15  1233
 9 FL    2023-05-10  6937
10 NYC   2022-10-26 14590
# i 10,370 more rows
```

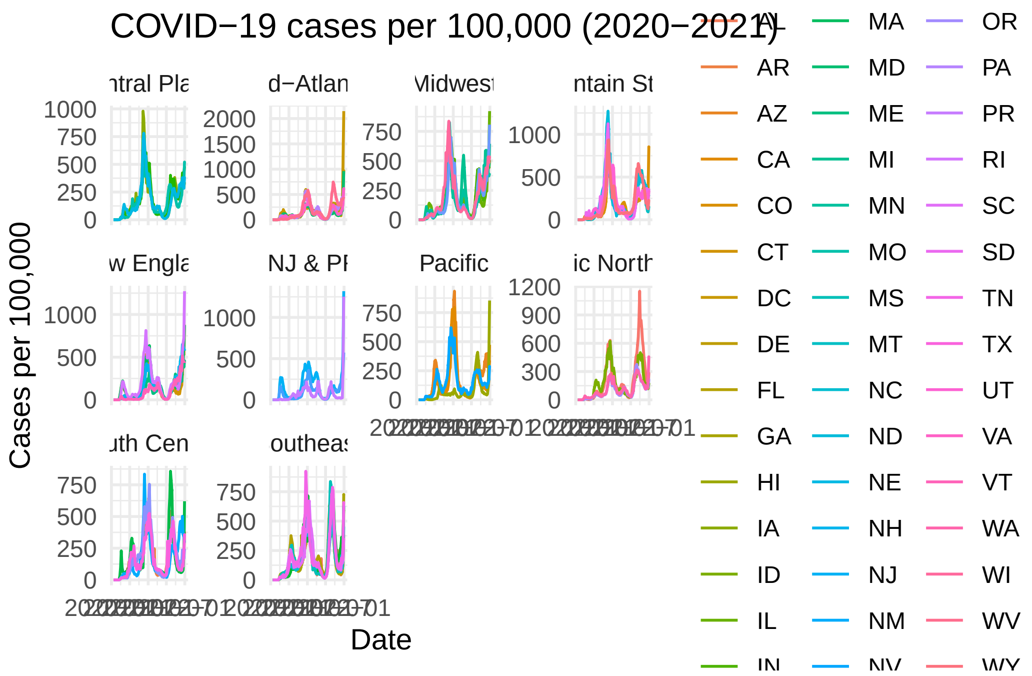
12. For 2020 and 2021, make a time series plot of cases per 100,000 versus time for each state. Stratify the plot by region name. Make sure to label you graph appropriately.

```r
library(tidyverse)
library(lubridate)
```

```r
cases <- cases |>
  filter(year(date) %in% c(2020, 2021)) |>
  left_join(population, by = "state") |>
  mutate(cases_per_100k = (cases / population) * 100000) |>
  filter(!is.na(cases_per_100k))
```

```
Warning in left_join(filter(cases, year(date) %in% c(2020, 2021)), population, : Detected an
to-many relationship between `x` and `y`.
i Row 1 of `x` matches multiple rows in `y`.
i Row 103 of `y` matches multiple rows in `x`.
i If a many-to-many relationship is expected, set `relationship =
  "many-to-many"` to silence this warning.
```

```r
cases |>
  ggplot(aes(x = date, y = cases_per_100k, color = state)) +
  geom_line() +
  facet_wrap(~ region_name, scales = "free_y") +
  labs(
    title = "COVID-19 cases per 100,000 (2020-2021)",
    x = "Date",
    y = "Cases per 100,000",
    color = "State"
  ) +
  theme_minimal()
```



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)

cases |>
  mutate(
    date  = ymd(date),
    year  = year(date),
    m_num = month(date),
    month = month(date, label = TRUE, abbr = FALSE)
  ) |>
  filter(year %in% c(2020, 2021)) |>
  group_by(year, month, m_num) |>
  summarise(total_cases = sum(cases, na.rm = TRUE), .groups = "drop") |>
  arrange(year, m_num) |>
  select(year, month, total_cases) |>
  kable()
```

```
Warning in attr(x, "align"): 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")
```

```
Warning in attr(x, "format"): 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")
```

year	month	total_cases
2020	January	22
2020	February	136
2020	March	100670
2020	April	1645296
2020	May	1233382
2020	June	1285104
2020	July	3954032
2020	August	2904786
2020	September	2803834
2020	October	3217864
2020	November	7774444
2020	December	13815080
2021	January	11298230
2021	February	5087928
2021	March	3857498
2021	April	3388378
2021	May	1897906
2021	June	969634
2021	July	2241878
2021	August	7038814
2021	September	9921614
2021	October	4635708
2021	November	4578236
2021	December	10586782

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"
```

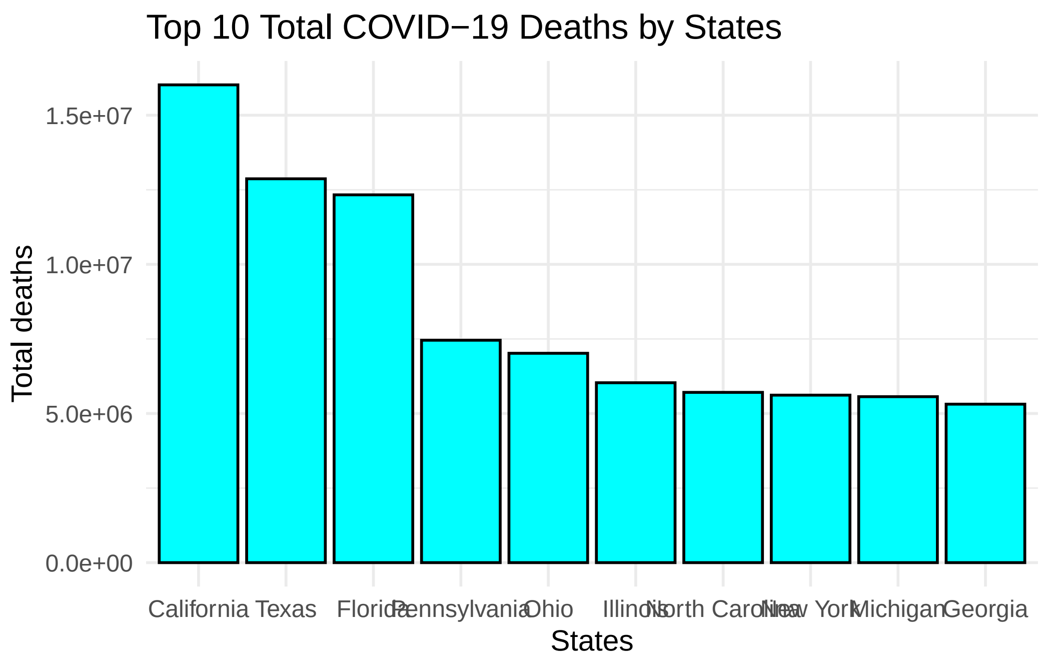
Use **httr2** to download COVID-19 death data from this endpoint. Make sure to remove the default limit to get all available data. Create a clean dataset called `deaths` with columns `state`, `date`, and `deaths` (renamed from the original column name). Ensure dates are in proper Date format and deaths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

```r
library(httr2)
library(tidyverse)
library(janitor)
library(lubridate)
api <- "https://data.cdc.gov/resource/9bhg-hcku.json"

response <- request(api) |>
  req_url_query(`$limit` = 10000000000) |>
  req_perform()

death_raw <- response |>
  resp_body_json(simplifyVector = TRUE) |>
  as_tibble()

#death_raw

deaths <- death_raw |>
  select(state, end_date, total_deaths) |>
  rename(date = end_date, deaths = total_deaths) |>
  mutate(
    deaths = as.numeric(deaths),
    date = as.Date(date)) |>
  filter(!state %in% c("United States", "Puerto Rico", "New York City", "District of Columbi

deaths
```

```
# A tibble: 127,500 x 3
   state   date       deaths
   <chr>   <date>      <dbl>
 1 Alabama 2023-09-23 231602
 2 Alabama 2023-09-23   1491
 3 Alabama 2023-09-23   2691
 4 Alabama 2023-09-23    344
 5 Alabama 2023-09-23    453
 6 Alabama 2023-09-23   2672
 7 Alabama 2023-09-23   4549
 8 Alabama 2023-09-23   5388
 9 Alabama 2023-09-23   6827
10 Alabama 2023-09-23   8639
# i 127,490 more rows
```

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
deaths_top10 <- deaths |>
  group_by(state) |>
  summarise(total_deaths = sum(deaths, na.rm = TRUE), .groups = "drop") |>
  arrange(desc(total_deaths)) |>
  slice_head(n = 10) |>
  mutate(state = factor(state, levels = state))

deaths_top10 |>
  ggplot(aes(x = state, y = total_deaths)) +
  geom_col(fill = "cyan", color = "black") +
  labs(
    title = "Top 10 Total COVID-19 Deaths by States",
    x = "States",
    y = "Total deaths"
  ) +
  theme_minimal()
```


Top 10 Total COVID−19 Deaths by States
Total deaths
1.5e+07
1.0e+07
5.0e+06
0.0e+00
California
Texas
Florida
Pennsylvania
Ohio
Illinois
North Carolina
New York
Michigan
Georgia
States