Problem set 4

Tiger Chaisutyakorn

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 2021 and 2022. 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.

```
library(httr2)
request <- request(url) |>
  req_headers(
    "Accept" = "application/json"
  ) |>
  req_url_query(
    get = "POP_2020,POP_2021,NAME",
    "for" = "state:*",
    key = census_key
  )
```

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.

```
response <- request |> req_perform()
cat(resp_status(response), resp_status_desc(response))
```

```
200 OK
```

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

```
# Your code here
print(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.

```
library(knitr)
population <- resp_body_json(response)
```

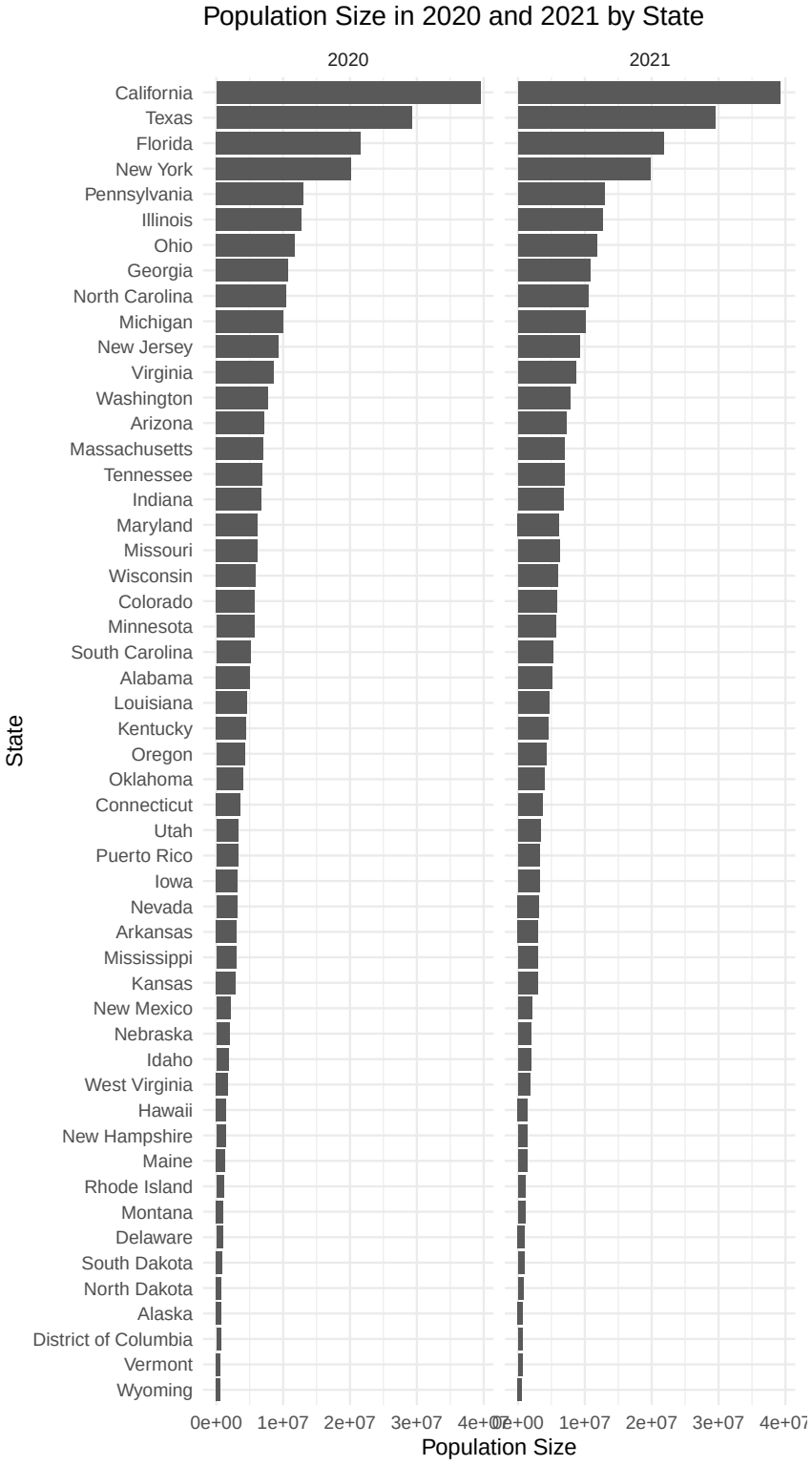
6. Examine the `population` matrix you just created. Notice that 1) it is not tidy, 2) the column types are not what we want, and 3) the first row is a header. Convert `population` to a tidy dataset. Remove the state ID column and change the name of the column with state names to `state_name`. Add a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

state_map <- data.frame(
  state_name = state.name,
  state_abb = state.abb
)

population <- do.call(rbind, population) |> ## Use janitor row to names function
  row_to_names(1) |>
  as_tibble() |> # convert to tibble
  select(-state) |> # remove stat column
  rename(state_name = "NAME") |> # rename state column to state_name
  mutate(state_name = as.character(state_name)) |>
  pivot_longer(!state_name, names_to = "year", values_to = "population") |> # use pivot_longe
  mutate(year = str_replace_all(year, "POP_", "")) |> # remove POP_ from year
  mutate(year = as.numeric(year), population = as.numeric(population)) |> # parese all releva
  left_join(state_map, by = "state_name") |> # add state abbreviations using state.abb varial
  mutate(
    state_abb = case_when(
      state_name == "District of Columbia" ~ "DC", # use case_when to add abbreviations for
      state_name == "Puerto Rico" ~ "PR",
      .default = state_abb
    )
  )
kable(head(population))
```

state_name	year	population	state_abb
Oklahoma	2020	3962031	OK
Oklahoma	2021	3986639	OK
Nebraska	2020	1961455	NE
Nebraska	2021	1963692	NE
Hawaii	2020	1451911	HI
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 |>
  mutate(state_name = fct_reorder(state_name, population)) |>
  ggplot(aes(x = population, y = state_name)) +
  geom_col() +
  facet_wrap("year") +
  labs(
    title = "Population Size in 2020 and 2021 by State",
    y = "State",
    x = "Population Size"
  ) +
  theme_minimal()
```


Population Size in 2020 and 2021 by State
2020
2021
California
Texas
Florida
New York
Pennsylvania
Illinois
Ohio
Georgia
North Carolina
Michigan
New Jersey
Virginia
Washington
Arizona
Massachusetts
Tennessee
Indiana
Maryland
Missouri
Wisconsin
Colorado
Minnesota
South Carolina
Alabama
Louisiana
Kentucky
Oregon
Oklahoma
Connecticut
Utah
Puerto Rico
Iowa
Nevada
Arkansas
Mississippi
Kansas
New Mexico
Nebraska
Idaho
West Virginia
Hawaii
New Hampshire
Maine
Rhode Island
Montana
Delaware
South Dakota
North Dakota
Alaska
District of Columbia
Vermont
Wyoming
State
0e+00
1e+07
2e+07
3e+07
4e+07
Population Size

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 two 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 <- fromJSON(url, simplifyVector = FALSE) |>
  map_df(~ as.data.frame(t(.x))) |>
  mutate(
    region = unlist(region),
    region_name = str_replace(region_name, "New York and New Jersey, Puerto Rico, Virgin Isla
  ) |>
  unnest(states) |>
  mutate(states = unlist(states))
kable(head(regions))
```

region	region_name	states
1	New England	Connecticut
1	New England	Maine
1	New England	Massachusetts
1	New England	New Hampshire
1	New England	Rhode Island
1	New England	Vermont

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

```
# Unnest the regions data.frame
regions <-
population <- left_join(population, regions, by = join_by(state_name == states))
kable(head(population))
```

state_name	year	population	state_abb	region	region_name
Oklahoma	2020	3962031	OK	6	South Central
Oklahoma	2021	3986639	OK	6	South Central
Nebraska	2020	1961455	NE	7	Central Plains
Nebraska	2021	1963692	NE	7	Central Plains
Hawaii	2020	1451911	HI	9	Pacific
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.

```
library(lubridate)
api <- "https://data.cdc.gov/resource/pwn4-m3yp.json"
response <- request(api) |>
  req_perform()
print(resp_status(response))
```

```
[1] 200
```

```
cases_raw <- response |>
  resp_body_json() |>
  bind_rows() |>
  mutate(
    across(c("tot_cases", "new_cases", "tot_deaths"), parse_number),
    across(c("date_updated", "start_date", "end_date"), parse_datetime)
  ) |>
  mutate(
    across(c("date_updated", "start_date", "end_date"), as.Date)
  )
print(dim(cases_raw))
```

```
[1] 1000   10
```

```
# We only see 1000 rows, but we expect weekly data from each state for whole 3 year periods
kable(head(cases_raw))
```

date_updated	state	start_date	end_date	tot_cases	new_cases	tot_deaths	new_deaths	new_historic_cases	new_historic_deaths
2023-02-23	AZ	2023-02-16	2023-02-22	2434631	3716	33042	39.0	23150	0
2022-12-22	LA	2022-12-15	2022-12-21	1507707	4041	18345	21.0	21397	0
2023-02-23	GA	2023-02-16	2023-02-22	3061141	5298	42324	88.0	6800	0
2023-03-30	LA	2023-03-23	2023-03-29	1588259	2203	18858	23.0	5347	0
2023-02-02	LA	2023-01-26	2023-02-01	1548508	5725	18572	47.0	4507	0
2023-03-23	LA	2023-03-16	2023-03-22	1580709	1961	18835	35.0	2239	0

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

```
[1] 200
```

```
cases_raw <- response |>
  resp_body_json() |>
  bind_rows() |>
  mutate(
    across(c("tot_cases", "new_cases", "tot_deaths"), parse_number),
    across(c("date_updated", "start_date", "end_date"), parse_datetime)
  ) |>
  mutate(
```

```
    across(c("date_updated", "start_date", "end_date"), as.Date)
  )
print(dim(cases_raw))
```

```
[1] 10380    10
```

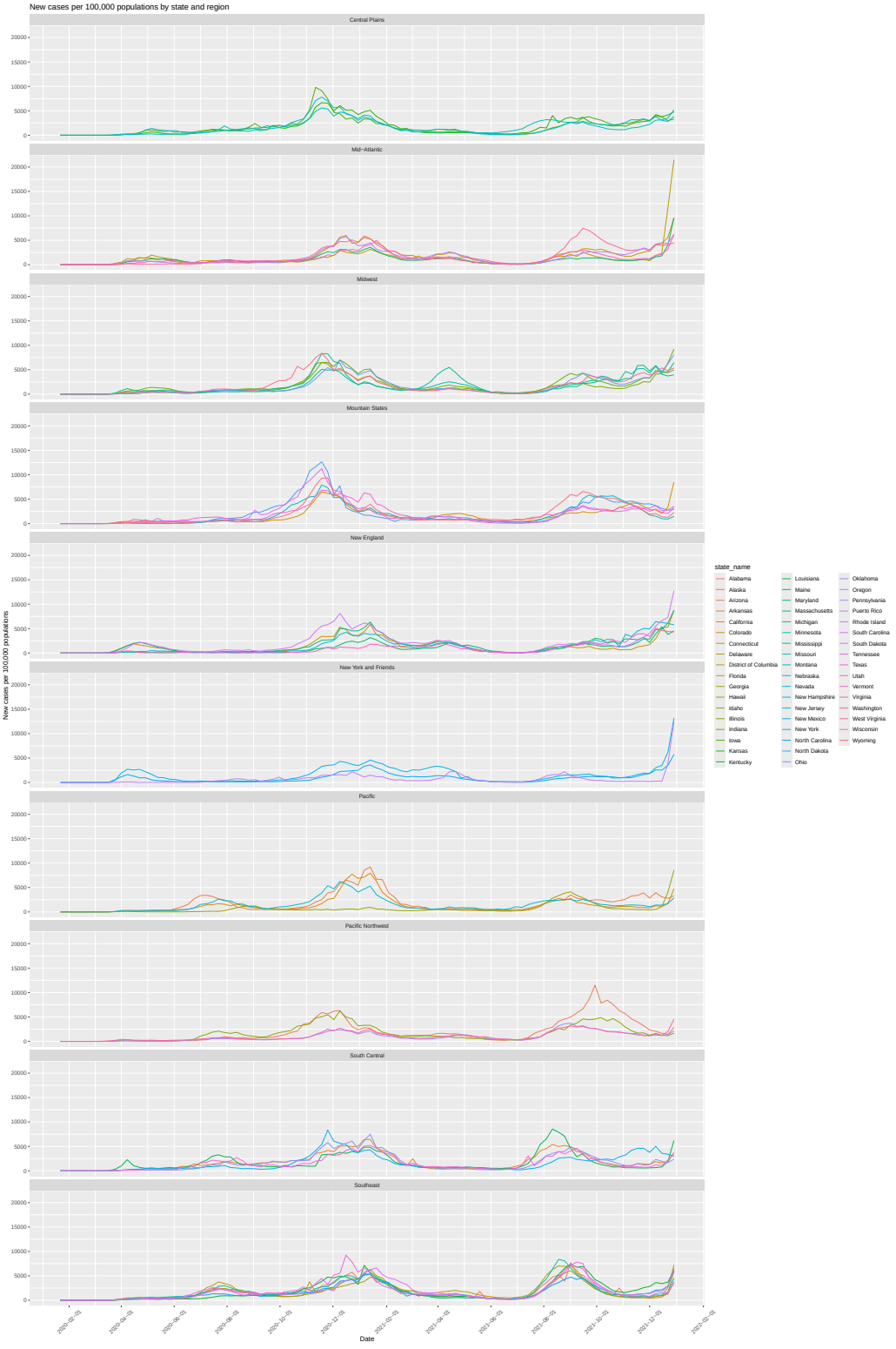
```
# Clean the dataframe
cases <- cases_raw |>
  select(end_date, new_cases, state) |>
  rename(date = end_date, cases = new_cases)
kable(head(cases))
```

date	cases	state
2023-02-22	3716	AZ
2022-12-21	4041	LA
2023-02-22	5298	GA
2023-03-29	2203	LA
2023-02-01	5725	LA
2023-03-22	1961	LA

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.

```
# Joining the region name
cases |>
  filter(date >= as.Date("2020-01-01") & date <= as.Date("2021-12-31")) |>
  mutate("year" = year(date)) |>
  left_join(population, by = join_by(state == state_abb, year == year)) |>
  drop_na(population) |>
  mutate(case_rate = cases / population * 10e5) |>
  ggplot(aes(x = date, y = case_rate, color = state_name)) +
  geom_line() +
  facet_wrap("region_name", ncol = 1) +
  labs(
    title = "New cases per 100,000 populations by state and region",
    x = "Date",
    y = "New cases per 100,000 populations"
  ) +
  scale_x_date(breaks = "2 month") +
  theme(
```

```
    legend.position = "right",
    axis.text.x = element_text(angle = 45, vjust = 0.5)
  )
```


New cases per 100,000 populations by state and region
Central Plains
Mid−Atlantic
Midwest
Mountain States
New England
New York and Friends
Pacific
Pacific Northwest
South Central
Southeast
New cases per 100,000 populations
Date
state_name
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Puerto Rico
Rhode Island
South Carolina
South Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Washington
West Virginia
Wisconsin
Wyoming

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.

```
cases |> mutate(
  date = parse_date_time(date, "ymd"),
  "year" = year(date),
  "month" = month(date)
  ) |>
  filter(year %in% c(2020, 2021)) |>
  group_by(year, month) |>
  summarize(
    total_cases = sum(cases)
  ) |>
  arrange(year, month) |>
  kable()
```

```
`summarise()` has grouped output by 'year'. You can override using the
`.groups` argument.
```

year	month	total_cases
2020	1	11
2020	2	68
2020	3	68245
2020	4	974032
2020	5	650943
2020	6	654904
2020	7	1989512
2020	8	1461283
2020	9	1415438
2020	10	1628598
2020	11	3932646
2020	12	7027128
2021	1	5808063
2021	2	2667511
2021	3	2068441
2021	4	1773591

year	month	total_cases
2021	5	972915
2021	6	493635
2021	7	1137440
2021	8	3572562
2021	9	5027537
2021	10	2356302
2021	11	2322814
2021	12	5615644

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.

```
# Your code here
response <- request(deaths_url) |>
  req_url_query(
    "$limit" = 100000000
  ) |>
  req_perform()
print(resp_status(response))
```

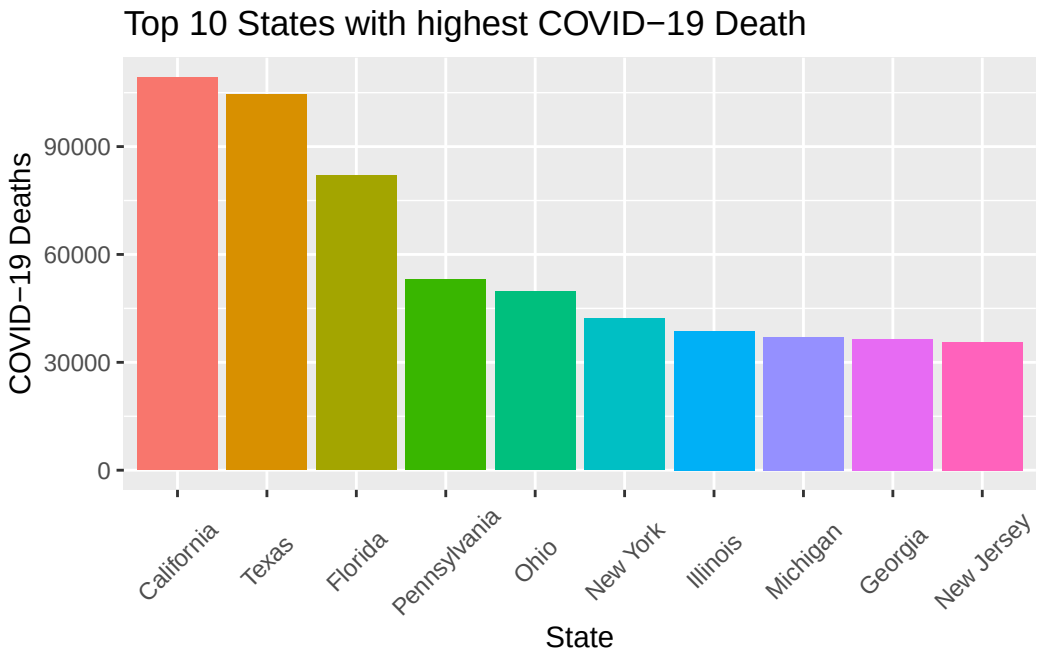
```
[1] 200
```

```
deaths <- response |>
  resp_body_json() |>
  bind_rows() |>
  filter(
    sex == "All Sexes",
    age_group == "All Ages",
    group == "By Total"
  ) |>
  select(end_date, covid_19_deaths, state) |>
  mutate(
    end_date = parse_date_time(end_date, "ymdHMS"),
```

```
    covid_19_deaths = as.numeric(covid_19_deaths)
  ) |>
  rename(
    date = end_date,
    deaths = covid_19_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.

```
# Your code here
deaths |>
  # filter to have only valid State
  semi_join(population, by = join_by(state == state_name)) |>
  arrange(desc(deaths)) |>
  head(10) |>
  mutate(state = fct_reorder(state, desc(deaths))) |>
  ggplot(aes(state, deaths)) +
  geom_col(aes(fill = state)) +
  labs(
    title = "Top 10 States with highest COVID-19 Death",
    x = "State",
    y = "COVID-19 Deaths"
  ) +
  theme(
    axis.text.x = element_text(angle = 45, vjust = 0.5),
    legend.position = "none"
  )
```


Top 10 States with highest COVID−19 Death
COVID−19 Deaths
90000
60000
30000
0
California
Texas
Florida
Pennsylvania
Ohio
New York
Illinois
Michigan
Georgia
New Jersey
State