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`.

```
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_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()
```

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, simplifyVector = TRUE)
```

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 |> ## Use janitor row to names function
  # convert to tibble
  row_to_names(row_number = 1) |>
  as_tibble() |>
```

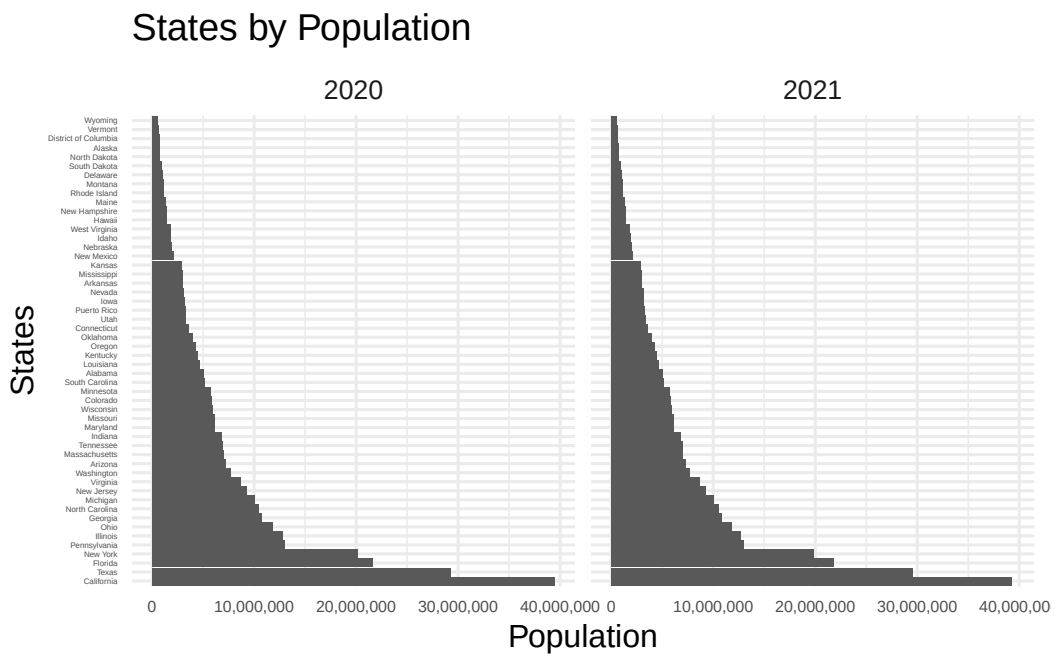
```
  # remove state column
  select(-state) |>
  # rename state column to state_name
  rename(state_name = NAME) |>
  # use pivot_longer to tidy
  # remove POP_ from year
  pivot_longer(cols = starts_with("POP_"),
               names_to = "year",
               values_to = "population") |>
  mutate(year = parse_number(year),
         population = parse_number(population), # parese all relevant columns to numeric
         state = case_when(state_name == "District of Columbia" ~ "DC",
                           state_name == "Puerto Rico"          ~ "PR",
                           # use case_when to add abbreviations for DC and PR
                           TRUE ~ state.abb[match(state_name, state.name)]))
# add state abbreviations using state.abb variable
population
```

```
# A tibble: 104 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 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`.

```
population |>
  filter(year %in% c(2020, 2021)) |>
  group_by(year) |>
  slice_max(population, n = 52, with_ties = FALSE) |>
  ungroup() |>
```

```r
# reorder state
ggplot(aes(x = reorder(state_name, -population), y = population)) +
# use geom_col to plot barplot
geom_col(width = 0.9) +
# flip coordinates
coord_flip() +
# facet by year
facet_wrap(~ year) +
scale_y_continuous(labels = scales::label_comma()) +
labs(x = "States", y = "Population", title = "States by Population") +
theme_minimal(base_size = 12) +
theme(axis.text.y = element_text(size = 3),
      axis.text.x = element_text(size = 6))
```



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 <- convert list to data frame. You can use map_df in purrr package
regions <- fromJSON(url) |> #use jsonlit JSON parser
  unnest(states) |>
  transmute(state_name = states,
            region = as.integer(region),
            region_name = region_name) |>
  mutate(region_name = if_else(
    region_name == "New York and New Jersey,
    Puerto Rico, Virgin Islands",
    "NY/NJ & Territories",
    region_name)) |>
  filter(state_name %in% c(state.name, "District of Columbia", "Puerto Rico"))
regions
```

```
# A tibble: 52 x 3
   state_name    region region_name
   <chr>          <int> <chr>
 1 Connecticut        1 New England
 2 Maine              1 New England
 3 Massachusetts      1 New England
 4 New Hampshire      1 New England
 5 Rhode Island       1 New England
 6 Vermont            1 New England
 7 New Jersey         2 New York and New Jersey, Puerto Rico, Virgin Islands
 8 New York           2 New York and New Jersey, Puerto Rico, Virgin Islands
 9 Puerto Rico        2 New York and New Jersey, Puerto Rico, Virgin Islands
10 Delaware           3 Mid-Atlantic
# i 42 more rows
```

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

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

```
# A tibble: 104 x 6
   state_name    year population state region region_name
   <chr>        <dbl>      <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.` 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_body_json(simplifyDataFrame = TRUE)
```

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

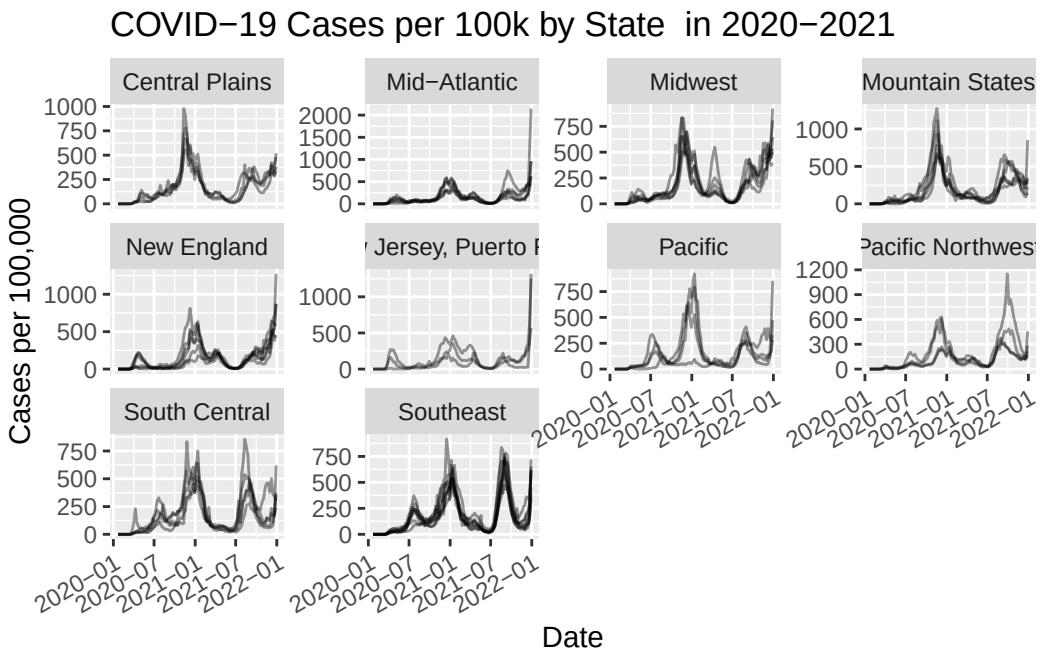
All the data is not there because CDC has a default limit.

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(simplifyDataFrame = TRUE)
cases <- cases_raw |>
  transmute(state_name = state,
            date  = ymd_hms(end_date, quiet = TRUE) |> as_date(),
            cases = parse_number(new_cases)) |>
  filter(!is.na(date), !is.na(cases))
```

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 |> transmute(state = state_name,
                   date  = as.Date(date),
                   cases = cases) |>
  left_join(population |>
              filter(year == 2021) |>
              select(state, region_name, population), by = "state") |>
  filter(state %in% c(state.abb, "DC", "PR")) |>
  filter(format(date, "%Y") %in% c("2020", "2021")) |>
  mutate(cases_per_100k = cases * 1e5 / population) |>
  ggplot(aes(x = date, y = cases_per_100k, group = state)) +
  geom_line(alpha = 0.4) +
  facet_wrap(~ region_name, scales = "free_y") +
  labs(x = "Date", y = "Cases per 100,000",
       title = "COVID-19 Cases per 100k by State  in 2020-2021") +
  theme(axis.text.x = element_text(angle = 30, hjust = 1))
```



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

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:

```
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.

```
deaths_raw <- request(deaths_url) |>
  req_url_query("$limit" = 10000000000) |>
  req_perform() |>
  resp_body_json(simplifyDataFrame = TRUE)
deaths <- deaths_raw |>
  filter(group == "By Year",
         sex == "All Sexes",
         age_group == "All Ages",
         state != "United States") |>
  transmute(state  = state,
            date   = ymd_hms(end_date) |> as_date(),
            deaths = parse_number(covid_19_deaths)) |>
  filter(!is.na(state), !is.na(date), !is.na(deaths))
deaths
```

```
                  state       date deaths
1               Alabama 2020-12-31   6706
2               Alabama 2021-12-31   9719
3               Alabama 2022-12-31   4226
4               Alabama 2023-09-23    869
5                Alaska 2020-12-31    254
6                Alaska 2021-12-31    839
7                Alaska 2022-12-31    330
8                Alaska 2023-09-23     69
9               Arizona 2020-12-31   9321
10              Arizona 2021-12-31  14060
11              Arizona 2022-12-31   5906
12              Arizona 2023-09-23   1020
13             Arkansas 2020-12-31   4027
14             Arkansas 2021-12-31   5336
15             Arkansas 2022-12-31   2773
16             Arkansas 2023-09-23    527
17           California 2020-12-31  33578
18           California 2021-12-31  48330
```

19	California	2022-12-31	22426
20	California	2023-09-23	4914
21	Colorado	2020-12-31	5073
22	Colorado	2021-12-31	6207
23	Colorado	2022-12-31	3413
24	Colorado	2023-09-23	685
25	Connecticut	2020-12-31	6298
26	Connecticut	2021-12-31	3153
27	Connecticut	2022-12-31	2579
28	Connecticut	2023-09-23	541
29	Delaware	2020-12-31	1096
30	Delaware	2021-12-31	1266
31	Delaware	2022-12-31	867
32	Delaware	2023-09-23	216
33	District of Columbia	2020-12-31	1004
34	District of Columbia	2021-12-31	707
35	District of Columbia	2022-12-31	444
36	District of Columbia	2023-09-23	76
37	Florida	2020-12-31	21827
38	Florida	2021-12-31	38873
39	Florida	2022-12-31	16898
40	Florida	2023-09-23	4296
41	Georgia	2020-12-31	10454
42	Georgia	2021-12-31	17310
43	Georgia	2022-12-31	7229
44	Georgia	2023-09-23	1506
45	Hawaii	2020-12-31	365
46	Hawaii	2021-12-31	744
47	Hawaii	2022-12-31	639
48	Hawaii	2023-09-23	195
49	Idaho	2020-12-31	1541
50	Idaho	2021-12-31	2785
51	Idaho	2022-12-31	1084
52	Idaho	2023-09-23	200
53	Illinois	2020-12-31	16721
54	Illinois	2021-12-31	11787
55	Illinois	2022-12-31	8720
56	Illinois	2023-09-23	1496
57	Indiana	2020-12-31	9775
58	Indiana	2021-12-31	10054
59	Indiana	2022-12-31	6032
60	Indiana	2023-09-23	1024
61	Iowa	2020-12-31	4769

62	Iowa	2021-12-31	3526
63	Iowa	2022-12-31	2242
64	Iowa	2023-09-23	489
65	Kansas	2020-12-31	3510
66	Kansas	2021-12-31	3806
67	Kansas	2022-12-31	2408
68	Kansas	2023-09-23	428
69	Kentucky	2020-12-31	4619
70	Kentucky	2021-12-31	8695
71	Kentucky	2022-12-31	5756
72	Kentucky	2023-09-23	1204
73	Louisiana	2020-12-31	7066
74	Louisiana	2021-12-31	6919
75	Louisiana	2022-12-31	3113
76	Louisiana	2023-09-23	624
77	Maine	2020-12-31	465
78	Maine	2021-12-31	1433
79	Maine	2022-12-31	1048
80	Maine	2023-09-23	289
81	Maryland	2020-12-31	6729
82	Maryland	2021-12-31	6101
83	Maryland	2022-12-31	4315
84	Maryland	2023-09-23	998
85	Massachusetts	2020-12-31	10221
86	Massachusetts	2021-12-31	5579
87	Massachusetts	2022-12-31	4335
88	Massachusetts	2023-09-23	1151
89	Michigan	2020-12-31	12342
90	Michigan	2021-12-31	14945
91	Michigan	2022-12-31	8112
92	Michigan	2023-09-23	1703
93	Minnesota	2020-12-31	5834
94	Minnesota	2021-12-31	5181
95	Minnesota	2022-12-31	3360
96	Minnesota	2023-09-23	923
97	Mississippi	2020-12-31	5162
98	Mississippi	2021-12-31	5907
99	Mississippi	2022-12-31	3190
100	Mississippi	2023-09-23	696
101	Missouri	2020-12-31	8059
102	Missouri	2021-12-31	8946
103	Missouri	2022-12-31	5433
104	Missouri	2023-09-23	925

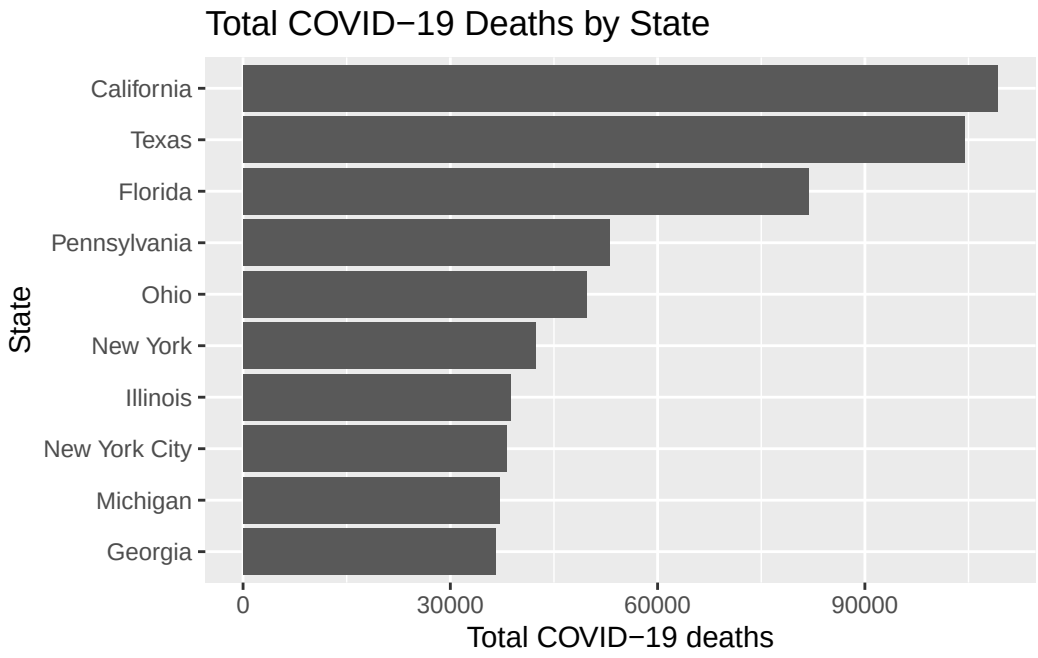
105	Montana	2020-12-31	1269
106	Montana	2021-12-31	1816
107	Montana	2022-12-31	729
108	Montana	2023-09-23	147
109	Nebraska	2020-12-31	2253
110	Nebraska	2021-12-31	1939
111	Nebraska	2022-12-31	1296
112	Nebraska	2023-09-23	327
113	Nevada	2020-12-31	3527
114	Nevada	2021-12-31	5645
115	Nevada	2022-12-31	2713
116	Nevada	2023-09-23	374
117	New Hampshire	2020-12-31	828
118	New Hampshire	2021-12-31	1204
119	New Hampshire	2022-12-31	821
120	New Hampshire	2023-09-23	259
121	New Jersey	2020-12-31	18164
122	New Jersey	2021-12-31	9605
123	New Jersey	2022-12-31	6571
124	New Jersey	2023-09-23	1351
125	New Mexico	2020-12-31	2886
126	New Mexico	2021-12-31	3704
127	New Mexico	2022-12-31	2008
128	New Mexico	2023-09-23	365
129	New York	2020-12-31	16195
130	New York	2021-12-31	14801
131	New York	2022-12-31	9091
132	New York	2023-09-23	2186
133	New York City	2020-12-31	22282
134	New York City	2021-12-31	9059
135	New York City	2022-12-31	5838
136	New York City	2023-09-23	988
137	North Carolina	2020-12-31	8789
138	North Carolina	2021-12-31	15235
139	North Carolina	2022-12-31	8385
140	North Carolina	2023-09-23	1974
141	North Dakota	2020-12-31	1513
142	North Dakota	2021-12-31	939
143	North Dakota	2022-12-31	538
144	North Dakota	2023-09-23	111
145	Ohio	2020-12-31	15097
146	Ohio	2021-12-31	20536
147	Ohio	2022-12-31	11972

148	Ohio	2023-09-23	2124
149	Oklahoma	2020-12-31	5248
150	Oklahoma	2021-12-31	7850
151	Oklahoma	2022-12-31	4562
152	Oklahoma	2023-09-23	860
153	Oregon	2020-12-31	1612
154	Oregon	2021-12-31	4190
155	Oregon	2022-12-31	2801
156	Oregon	2023-09-23	629
157	Pennsylvania	2020-12-31	18407
158	Pennsylvania	2021-12-31	20546
159	Pennsylvania	2022-12-31	11642
160	Pennsylvania	2023-09-23	2454
161	Rhode Island	2020-12-31	1918
162	Rhode Island	2021-12-31	1217
163	Rhode Island	2022-12-31	769
164	Rhode Island	2023-09-23	179
165	South Carolina	2020-12-31	5705
166	South Carolina	2021-12-31	9497
167	South Carolina	2022-12-31	4572
168	South Carolina	2023-09-23	1068
169	South Dakota	2020-12-31	1715
170	South Dakota	2021-12-31	987
171	South Dakota	2022-12-31	634
172	South Dakota	2023-09-23	136
173	Tennessee	2020-12-31	7927
174	Tennessee	2021-12-31	13705
175	Tennessee	2022-12-31	7223
176	Tennessee	2023-09-23	1553
177	Texas	2020-12-31	33542
178	Texas	2021-12-31	48787
179	Texas	2022-12-31	18614
180	Texas	2023-09-23	3478
181	Utah	2020-12-31	1657
182	Utah	2021-12-31	2671
183	Utah	2022-12-31	1333
184	Utah	2023-09-23	206
185	Vermont	2020-12-31	146
186	Vermont	2021-12-31	312
187	Vermont	2022-12-31	387
188	Vermont	2023-09-23	131
189	Virginia	2020-12-31	6156
190	Virginia	2021-12-31	9668

```
191            Virginia 2022-12-31   6003
192            Virginia 2023-09-23   1321
193          Washington 2020-12-31   3703
194          Washington 2021-12-31   6165
195          Washington 2022-12-31   4156
196          Washington 2023-09-23   1054
197       West Virginia 2020-12-31   1564
198       West Virginia 2021-12-31   3938
199       West Virginia 2022-12-31   2360
200       West Virginia 2023-09-23    477
201           Wisconsin 2020-12-31   6286
202           Wisconsin 2021-12-31   6076
203           Wisconsin 2022-12-31   3912
204           Wisconsin 2023-09-23    790
205             Wyoming 2020-12-31    461
206             Wyoming 2021-12-31    963
207             Wyoming 2022-12-31    353
208             Wyoming 2023-09-23     78
209         Puerto Rico 2020-12-31   1629
210         Puerto Rico 2021-12-31   1644
211         Puerto Rico 2022-12-31   2366
212         Puerto Rico 2023-09-23    768
```

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 |>
  group_by(state) |>
  summarise(total_deaths = sum(deaths, na.rm = TRUE), .groups = "drop") |>
  slice_max(total_deaths, n = 10, with_ties = FALSE) |>
  ggplot(aes(x = reorder(state, total_deaths), y = total_deaths)) +
  geom_col() +
  coord_flip() +
  labs(x = "State",
       y = "Total COVID-19 deaths",
       title = "Total COVID-19 Deaths by State")
```


Total COVID−19 Deaths by State
California
Texas
Florida
Pennsylvania
Ohio
New York
Illinois
New York City
Michigan
Georgia
State
0
30000
60000
90000
Total COVID−19 deaths