

COVID 19 Analysis

Tyler Kephart

2024-08-19

Required Packages

```
library(tidyverse)
library(lubridate)
library(usmap)
```

Part 1 - Basic Exploration of US Data

The New York Times (the Times) has aggregated reported COVID-19 data from state and local governments and health departments since 2020 and provides public access through a repository on GitHub. One of the data sets provided by the Times is county-level data for cumulative cases and deaths each day. This will be your primary data set for the first two parts of your analysis.

County-level COVID data from 2020, 2021, and 2022 has been imported below. Each row of data reports the cumulative number of cases and deaths for a specific county each day. A FIPS code, a standard geographic identifier, is also provided which you will use in Part 2 to construct a map visualization at the county level for a state.

Additionally, county-level population estimates reported by the US Census Bureau has been imported as well. You will use these estimates to calculate statistics per 100,000 people.

```
# Import New York Times COVID-19 data
# Import Population Estimates from US Census Bureau

us_counties_2020 <- read_csv(
  "https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-counties-2020.csv")
us_counties_2021 <- read_csv(
  "https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-counties-2021.csv")
us_counties_2022 <- read_csv(
  "https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-counties-2022.csv")

us_population_estimates <- read_csv("fips_population_estimates.csv")
```

Question 1

Your first task is to combine and tidy the 2020, 2021, and 2022 COVID data sets and find the total deaths and cases for each day since March 15, 2020 (2020-03-15). The data sets provided from the NY Times also includes statistics from Puerto Rico, a US territory. You may remove these observations from the data as they will not be needed for your analysis. Once you have tidied the data, find the total COVID-19 cases and deaths since March 15, 2020. Write a sentence or two after the code block communicating your results. Use inline code to include the `max_date`, `us_total_cases`, and `us_total_deaths` variables. To write inline code use `r`.

```
# combine the 2020, 2021, and 2022 COVID data sets
us_counties <- rbind(
  us_counties_2020,
```

```

    us_counties_2021,
    us_counties_2022,
    deparse.level = 1) %>%
  # exclude Puerto Rico, idk why other territories get to stay
  filter(state != "Puerto Rico",
    # filter from March 15, 2020 and after
    date >= as.Date("2020-03-15"))
# summarize total cases and deaths by date
daily_totals <- us_counties %>%
  group_by(date) %>%
  summarise(total_deaths = sum(deaths),
    total_cases = sum(cases))

daily_totals

```

```

## # A tibble: 1,022 x 3
##   date      total_deaths total_cases
##   <date>      <dbl>      <dbl>
## 1 2020-03-15         68        3595
## 2 2020-03-16         91        4502
## 3 2020-03-17        117        5901
## 4 2020-03-18        162        8345
## 5 2020-03-19        212       12387
## 6 2020-03-20        277       17998
## 7 2020-03-21        359       24507
## 8 2020-03-22        457       33050
## 9 2020-03-23        577       43474
## 10 2020-03-24        783       53899
## # i 1,012 more rows

```

```

# set variables for inline code usage
max_date <- max(daily_totals$date)
us_total_cases <- max(daily_totals$total_cases)
us_total_deaths <- max(daily_totals$total_deaths)

```

```

# Your output should look similar to the following tibble:
#
#   A tibble: 657 x 3
#     date      total_deaths total_cases
#     <date>      <dbl>      <dbl>
# 1 2020-03-15         68        3595
# 2 2020-03-16         91        4502
# 3 2020-03-17        117        5901
# 4 2020-03-18        162        8345
# 5 2020-03-19        212       12387
# 6 2020-03-20        277       17998
# 7 2020-03-21        359       24507
# 8 2020-03-22        457       33050
# 9 2020-03-23        577       43474
# 10 2020-03-24        783       53899
# ... with 647 more rows

```

As of December 31, 2022, there was more than 99.37 million cases and 1.09 million deaths from COVID-19 in the United States (excluding Puerto Rico).

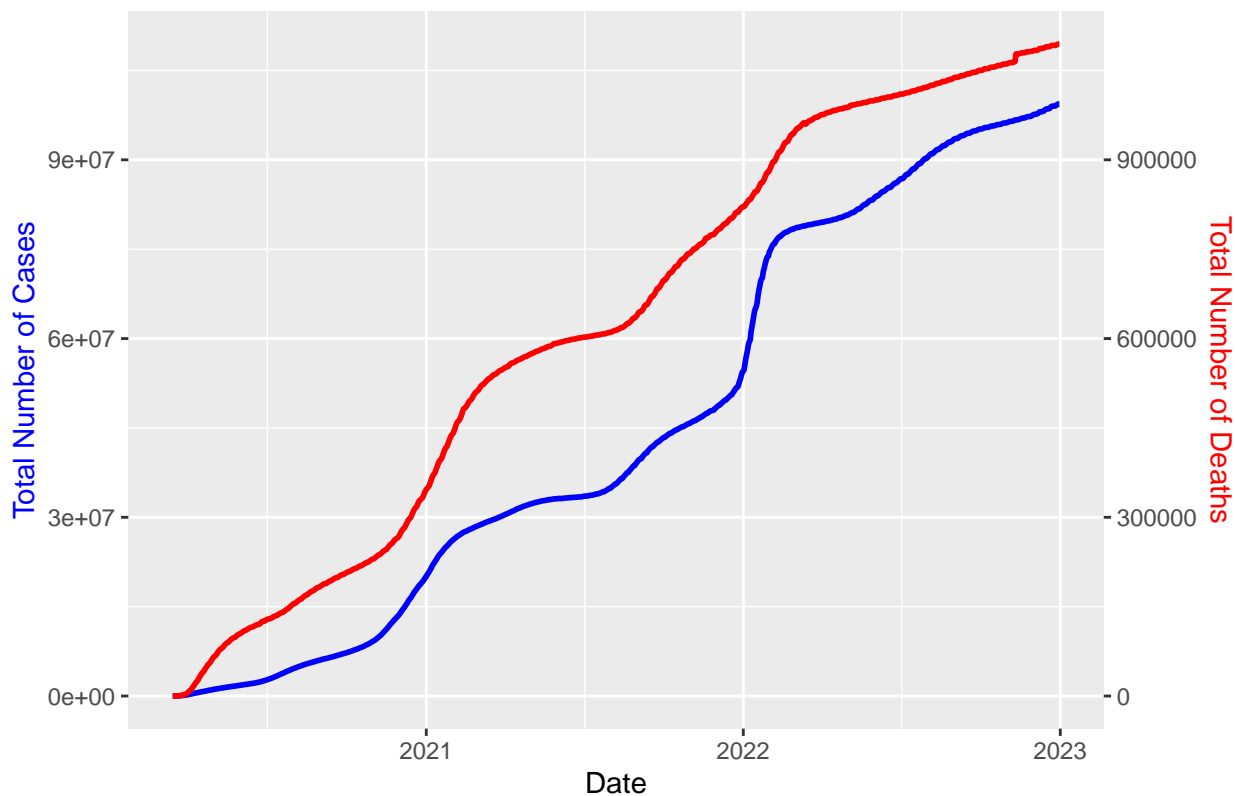
Question 2

Create a visualization for the total number of deaths and cases in the US since March 15, 2020. Before you create your visualization, review the types of plots you can create using the ggplot2 library and think about which plots would be

effective in communicating your results. After you have created your visualization, write a few sentences describing your visualization. How could the plot be interpreted? Could it be misleading?

```
# plot for the total number of US cases and deaths since March 15, 2020
ggplot(daily_totals, aes(x = date)) +
  geom_line(aes(y = total_cases, color = "Cases"), linewidth = 1) +
  # scaling deaths for visibility
  geom_line(aes(y = total_deaths * 100, color = "Deaths"), linewidth = 1) +
  # set up y axis for 2 variables
  scale_y_continuous(name = "Total Number of Cases",
                     # add second axis for deaths
                     sec.axis = sec_axis(~./100, name = "Total Number of Deaths")) +
  # color the y axis titles to match the plotted lines
  theme(axis.title.y.left = element_text(color = "blue"),
        axis.title.y.right = element_text(color = "red")) +
  labs(title = "Total COVID-19 Cases and Deaths in the US Since March 15, 2020",
        x = "Date") +
  # remove legend name
  theme(legend.position = "none") +
  scale_color_manual(values = c("Cases" = "blue", "Deaths" = "red"))
```

Total COVID-19 Cases and Deaths in the US Since March 15, 2020



There seems to have been a spike in cases early 2022 that was possibly due to increased travel during the holiday season in the United States.

Question 3

While it is important to know the total deaths and cases throughout the COVID-19 pandemic, it is also important for local and state health officials to know the the number of new cases and deaths each day to understand how rapidly the virus is spreading. Using the table you created in Question 1, calculate the number of new deaths and cases each day and a seven-day average of new deaths and cases. Once you have organized your data, find the days that saw the largest number of new cases and deaths. Write a sentence or two after the code block communicating your results.

```
# calculate number of new deaths and cases each day
daily_7day_avg <- daily_totals %>%
  mutate(
    delta_deaths_1 = total_deaths - lag(total_deaths),
    delta_cases_1 = total_cases - lag(total_cases))
# calculate rolling averages
calculate_rolling_average <- function(x, window_size) {
  sapply(seq_along(x), function(i) {
    if (i < window_size) return(NA)
    mean(x[(i - window_size + 1):i])
  })
}
# add 7-day rolling averages
daily_7day_avg <- daily_7day_avg %>%
  mutate(
    delta_deaths_7 = calculate_rolling_average(delta_deaths_1, 7),
    delta_cases_7 = calculate_rolling_average(delta_cases_1, 7)
  )
daily_7day_avg
```

```
## # A tibble: 1,022 x 7
##   date      total_deaths total_cases delta_deaths_1 delta_cases_1
##   <date>      <dbl>      <dbl>      <dbl>      <dbl>
## 1 2020-03-15         68        3595          NA          NA
## 2 2020-03-16         91        4502          23          907
## 3 2020-03-17        117        5901          26        1399
## 4 2020-03-18        162        8345          45        2444
## 5 2020-03-19        212       12387          50        4042
## 6 2020-03-20        277       17998          65        5611
## 7 2020-03-21        359       24507          82        6509
## 8 2020-03-22        457       33050          98        8543
## 9 2020-03-23        577       43474         120       10424
## 10 2020-03-24        783       53899         206       10425
## # i 1,012 more rows
## # i 2 more variables: delta_deaths_7 <dbl>, delta_cases_7 <dbl>
```

```
# find the days with the largest number of new cases and deaths
max_new_cases_date <- daily_7day_avg %>%
  slice_max(order_by = delta_cases_1, n = 1)
max_new_deaths_date <- daily_7day_avg %>%
  slice_max(order_by = delta_deaths_1, n = 1)
```

Your output should look similar to the following tibble:

```
#
# date
# total_deaths > the cumulative number of deaths up to and including the associated date
# total_cases > the cumulative number of cases up to and including the associated date
# delta_deaths_1 > the number of new deaths since the previous day
# delta_cases_1 > the number of new cases since the previous day
# delta_deaths_7 > the average number of deaths in a seven-day period
# delta_cases_7 > the average number of cases in a seven-day period
#==
# A tibble: 813 x 7
#   date      total_deaths total_cases delta_deaths_1 delta_cases_1 delta_deaths_7 delta_cases_7
#   <date>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
# 1 2020-03-15         68        3600          0          0          NA          NA
# 2 2020-03-16         91        4507          23          907          NA          NA
# 3 2020-03-17        117        5906          26        1399          NA          NA
```

```
# 4 2020-03-18      162      8350      45      2444      NA      NA
# 5 2020-03-19      212     12393      50     4043      NA      NA
# 6 2020-03-20      277     18012      65     5619      NA      NA
# 7 2020-03-21      360     24528      83     6516      NA      NA
# 8 2020-03-22      458     33073      98     8545     55.7    4210.
# 9 2020-03-23      579     43505     121    10432     69.7    5571.
# 10 2020-03-24     785     53938     206    10433     95.4    6862.
# ... with 803 more rows
```

As of December 31, 2022, the most amount of new cases in a single day happened on January 10, 2022 with 1.427097×10^6 cases. The most amount of new deaths in a single day happened on November 11, 2022 with 1.0037×10^4 deaths.

Question 4

Create a new table, based on the table from Question 3, and calculate the number of new deaths and cases per 100,000 people each day and a seven day average of new deaths and cases per 100,000 people.

```
# aggregate US population by year
us_population_totals <- us_population_estimates %>%
  group_by(Year) %>%
  summarize(total_population = sum(Estimate))

# add year to daily_totals
daily_7day_avg_yr <- daily_7day_avg %>%
  mutate(year = year(date))

# join with population totals
daily_7day_avg_yr_pop <- daily_7day_avg_yr %>%
  left_join(us_population_totals, by = c("year" = "Year"))

# calculate metrics per 100,000 people
daily_7day_avg_per_100k <- daily_7day_avg_yr_pop %>%
  mutate(
    total_deaths = (total_deaths / total_population) * 100000,
    total_cases = (total_cases / total_population) * 100000,
    delta_deaths_1 = (delta_deaths_1 / total_population) * 100000,
    delta_cases_1 = (delta_cases_1 / total_population) * 100000,
    delta_deaths_7 = calculate_rolling_average(delta_deaths_1, 7),
    delta_cases_7 = calculate_rolling_average(delta_cases_1, 7)
  ) %>%
  select(date,
    total_deaths,
    total_cases,
    delta_deaths_1,
    delta_cases_1,
    delta_deaths_7,
    delta_cases_7)
daily_7day_avg_per_100k
```

```
## # A tibble: 1,022 x 7
##   date      total_deaths total_cases delta_deaths_1 delta_cases_1
##   <date>      <dbl>      <dbl>      <dbl>      <dbl>
## 1 2020-03-15    0.0205      1.08      NA          NA
## 2 2020-03-16    0.0275      1.36    0.00694    0.274
## 3 2020-03-17    0.0353      1.78    0.00784    0.422
## 4 2020-03-18    0.0489      2.52    0.0136    0.737
## 5 2020-03-19    0.0640      3.74    0.0151    1.22
```

```
## 6 2020-03-20      0.0836      5.43      0.0196      1.69
## 7 2020-03-21      0.108      7.39      0.0247      1.96
## 8 2020-03-22      0.138      9.97      0.0296      2.58
## 9 2020-03-23      0.174     13.1      0.0362      3.14
## 10 2020-03-24     0.236     16.3      0.0621      3.14
## # i 1,012 more rows
## # i 2 more variables: delta_deaths_7 <dbl>, delta_cases_7 <dbl>
```

```
# Your output should look similar to the following tibble:
#
# date
# total_deaths    > the cumulative number of deaths up to and including the associated date
# total_cases     > the cumulative number of cases up to and including the associated date
# delta_deaths_1  > the number of new deaths since the previous day
# delta_cases_1   > the number of new cases since the previous day
# delta_deaths_7  > the average number of deaths in a seven-day period
# delta_cases_7   > the average number of cases in a seven-day period
#==
# A tibble: 657 x 7
#   date          total_deaths total_cases delta_deaths_1 delta_cases_1 delta_deaths_7 delta_cases_7
#   <date>          <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
# 1 2020-03-15      0.0205      1.08          0          0          NA          NA
# 2 2020-03-16      0.0275      1.36      0.00694    0.274      NA          NA
# 3 2020-03-17      0.0353      1.78      0.00784    0.422      NA          NA
# 4 2020-03-18      0.0489      2.52      0.0136    0.737      NA          NA
# 5 2020-03-19      0.0640      3.74      0.0151    1.22      NA          NA
# 6 2020-03-20      0.0836      5.43      0.0196    1.69      NA          NA
# 7 2020-03-21      0.108      7.39      0.0247    1.96      NA          NA
# 8 2020-03-22      0.138      9.97      0.0296    2.58      0.0168      1.27
# 9 2020-03-23      0.174     13.1      0.0362    3.14      0.0209      1.68
# 10 2020-03-24     0.236     16.3      0.0621    3.14      0.0287      2.07
```

The increasing values for `delta_deaths_1` and `delta_cases_1` suggest a rising trend in both metrics, which is further confirmed by the increasing `delta_deaths_7` and `delta_cases_7`. This type of data can be useful for understanding the progression of the pandemic and the effectiveness of interventions over time.

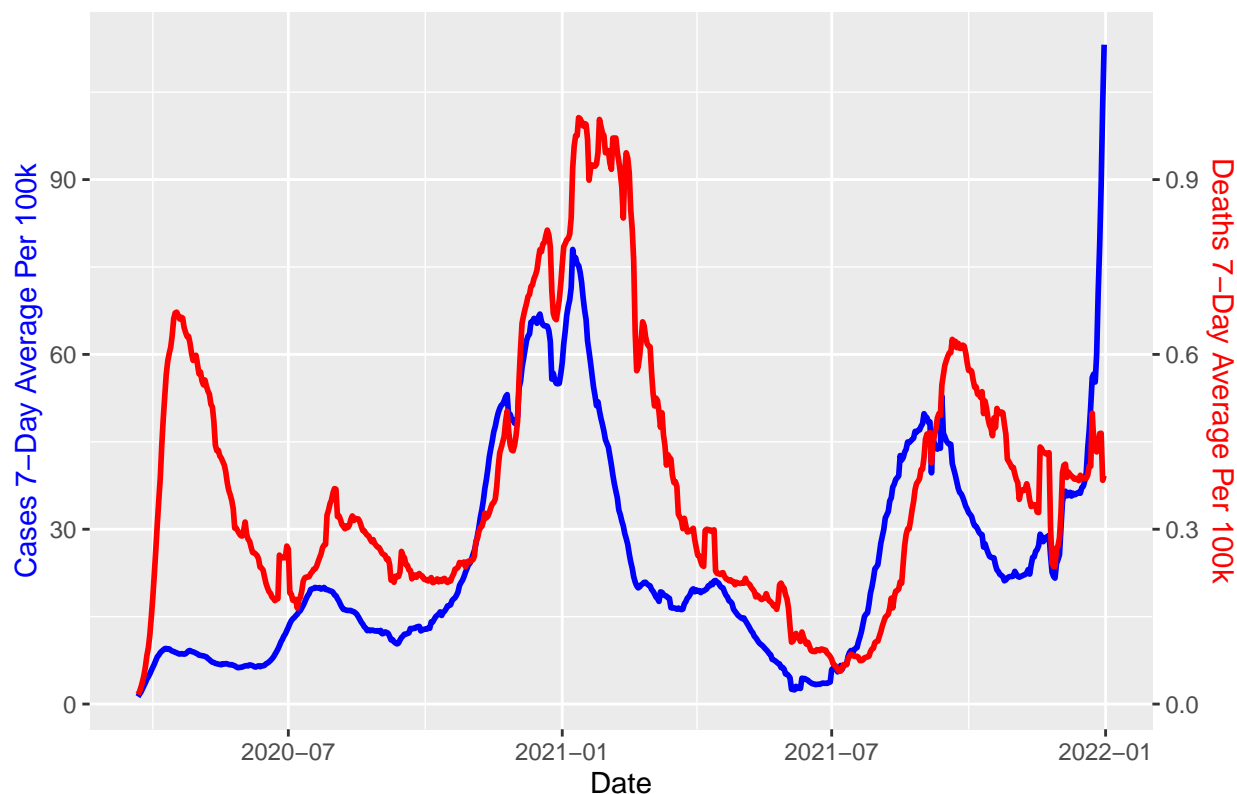
Question 5

Create a visualization to compare the seven-day average cases and deaths per 100,000 people.

```
# compare 7-day average cases and deaths per 100k people
daily_7day_avg_per_100k %>%
  filter(!is.na(delta_cases_7)) %>%
  ggplot(aes(x = date)) +
  geom_line(aes(y = delta_cases_7, color = "Cases"), linewidth = 1) +
  # scale for visibility
  geom_line(aes(y = delta_deaths_7 * 100, color = "Deaths"), linewidth = 1) +
  # set up y axis for 2 variables
  scale_y_continuous(name = "Cases 7-Day Average Per 100k",
    # add second axis for deaths
    sec.axis = sec_axis(~./100, name = "Deaths 7-Day Average Per 100k")) +
  # color the y axis titles to match the plotted lines
  theme(axis.title.y.left = element_text(color = "blue"),
    axis.title.y.right = element_text(color = "red")) +
  labs(title = "7-Day Average of Cases and Deaths per 100,000 People",
    x = "Date") +
  # remove legend name
```

```
theme(legend.position = "none") +
scale_color_manual(values = c("Cases" = "blue", "Deaths" = "red"))
```

7-Day Average of Cases and Deaths per 100,000 People



The plot really highlights the increased rates of cases and deaths around January 2021.

Part 2 - US State Comparison

While understanding the trends on a national level can be helpful in understanding how COVID-19 impacted the United States, it is important to remember that the virus arrived in the United States at different times. For the next part of your analysis, you will begin to look at COVID related deaths and cases at the state and county-levels.

Question 1

Your first task in Part 2 is to determine the top 10 states in terms of total deaths and cases between March 15, 2020, and December 31, 2021.

Once you have both lists, briefly describe your methodology and your results.

```
# top 10 states for total deaths and cases between 2020-03-15 and 2021-12-31
state_totals_end2021 <- us_counties %>%
  # just need to filter to 2021-12-31 as totals are already as of that date
  filter(date == "2021-12-31") %>%
  group_by(state, date) %>%
  summarise(total_deaths = sum(deaths),
            total_cases = sum(cases)) %>%
  # sort totals
  arrange(desc(total_cases),
           desc(total_deaths))
```

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

```
state_totals_end2021
```

```
## # A tibble: 55 x 4
## # Groups:   state [55]
##   state      date      total_deaths total_cases
##   <chr>    <date>      <dbl>      <dbl>
## 1 California 2021-12-31    76709    5515613
## 2 Texas      2021-12-31    76062    4574881
## 3 Florida    2021-12-31    62504    4166392
## 4 New York   2021-12-31    58993    3473970
## 5 Illinois   2021-12-31    31017    2154058
## 6 Pennsylvania 2021-12-31    36705    2036424
## 7 Ohio       2021-12-31    29447    2016095
## 8 Georgia    2021-12-31    30283    1798497
## 9 Michigan   2021-12-31    28984    1706355
## 10 North Carolina 2021-12-31    19436    1685504
## # i 45 more rows
```

```
# Your transformed data should look similar to the following tibble:
```

```
#
# A tibble: 51 x 4
#   state      date      total_deaths total_cases
#   <chr>    <date>      <dbl>      <dbl>
# 1 California 2021-12-31    76709    5515613
# 2 Texas      2021-12-31    76062    4574881
# 3 Florida    2021-12-31    62504    4166392
# 4 New York   2021-12-31    58993    3473970
# 5 Illinois   2021-12-31    31017    2154058
# 6 Pennsylvania 2021-12-31    36705    2036424
# 7 Ohio       2021-12-31    29447    2016095
# 8 Georgia    2021-12-31    30283    1798497
# 9 Michigan   2021-12-31    28984    1706355
# 10 North Carolina 2021-12-31    19436    1685504
# ... with 41 more rows
```

The three most populated states (California, Texas, and Florida) also had the most deaths and cases among the states.

Question 2

Determine the top 10 states in terms of deaths per 100,000 people and cases per 100,000 people between March 15, 2020, and December 31, 2021.

Once you have both lists, briefly describe your methodology and your results. Do you expect the lists to be different than the one produced in Question 1? Which method, total or per 100,000 people, is a better method for reporting the statistics?

```
# create population totals for the states
state_population_totals <- us_population_estimates %>%
  group_by(STNAME, Year) %>%
  summarize(total_population = sum(Estimate))
```

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



```

# join to state totals
state_totals_end2021 %>%
  # add year to join on
  mutate(year = year(date)) %>%
  left_join(
    # filter to just 2021
    filter(state_population_totals, Year == 2021),
    by = c("state" = "STNAME")) %>%
  # make _per_100k columns
  mutate(deaths_per_100k = (total_deaths / total_population) * 100000,
         cases_per_100k = (total_cases / total_population) * 100000) %>%
  # select the needed columns
  select(state, date, deaths_per_100k, cases_per_100k) %>%
  # sort totals per 100k
  arrange(desc(cases_per_100k), desc(deaths_per_100k))

```

```

## # A tibble: 55 x 4
## # Groups:   state [55]
##   state      date      deaths_per_100k cases_per_100k
##   <chr>    <date>          <dbl>         <dbl>
## 1 North Dakota 2021-12-31      265.         22482.
## 2 Alaska      2021-12-31      130.         21310.
## 3 Rhode Island 2021-12-31      280.         21093.
## 4 South Dakota 2021-12-31      278.         20014.
## 5 Wyoming     2021-12-31      264.         19979.
## 6 Tennessee   2021-12-31      296.         19783.
## 7 Kentucky    2021-12-31      269.         19173.
## 8 Florida     2021-12-31      287.         19128.
## 9 Utah        2021-12-31      113.         19088.
## 10 Wisconsin  2021-12-31      190.         19008.
## # i 45 more rows

```

```

# Your transformed data should look similar to the following tibble:
#
# A tibble: 51 x 4
#   state      date      deaths_per_100k cases_per_100k
#   <chr>    <date>          <dbl>         <dbl>
# 1 North Dakota 2021-12-31      265.         22482.
# 2 Alaska      2021-12-31      130.         21310.
# 3 Rhode Island 2021-12-31      280.         21093.
# 4 South Dakota 2021-12-31      278.         20014.
# 5 Wyoming     2021-12-31      264.         19979.
# 6 Tennessee   2021-12-31      296.         19783.
# 7 Kentucky    2021-12-31      269.         19173.
# 8 Florida     2021-12-31      287.         19128.
# 9 Utah        2021-12-31      113.         19088.
# 10 Wisconsin  2021-12-31      190.         19008.
# ... with 41 more rows

```

North Dakota, Alaska, Rhode Island, South Dakota, and Wyoming are some of the least populated states in the United states and that could be why the cases per 100K people are high.

Question 3

Now, select a state and calculate the seven-day averages for new cases and deaths per 100,000 people. Once you have calculated the averages, create a visualization using ggplot2 to represent the data.

```

# Colorado case and death totals from 2020-03-15 to 2021-12-31 per 100k people
colorado_7day_avg <- us_counties %>%
  # filter dates
  filter(between(date, as.Date("2020-03-15"), as.Date("2021-12-31")),
         state == "Colorado") %>%
  group_by(state, date) %>%
  # sum up deaths and cases
  summarise(total_deaths = sum(deaths),
            total_cases = sum(cases)) %>%
  # add year to join on
  mutate(year = year(date)) %>%
  left_join(
    # filter to just Colorado and join on year
    filter(state_population_totals, STNAME == "Colorado"),
    by = c("year" = "Year")) %>%
  mutate(
    # create 1-day deltas to create 7-day averages
    delta_deaths_1 = total_deaths - lag(total_deaths),
    delta_cases_1 = total_cases - lag(total_cases),
    # create the per 100k metrics
    deaths_per_100k = (total_deaths / total_population) * 100000,
    cases_per_100k = (total_cases / total_population) * 100000,
    # re-use function for 7-day rolling average
    deaths_7_day = calculate_rolling_average((delta_deaths_1 / total_population) * 100000, 7),
    cases_7_day = calculate_rolling_average((delta_cases_1 / total_population) * 100000, 7),
    # shorten name of column
    population = total_population) %>%
  # grab only the needed columns
  select(state,
         date,
         total_deaths,
         total_cases,
         population,
         deaths_per_100k,
         cases_per_100k,
         deaths_7_day,
         cases_7_day)

```

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

```
colorado_7day_avg
```

```

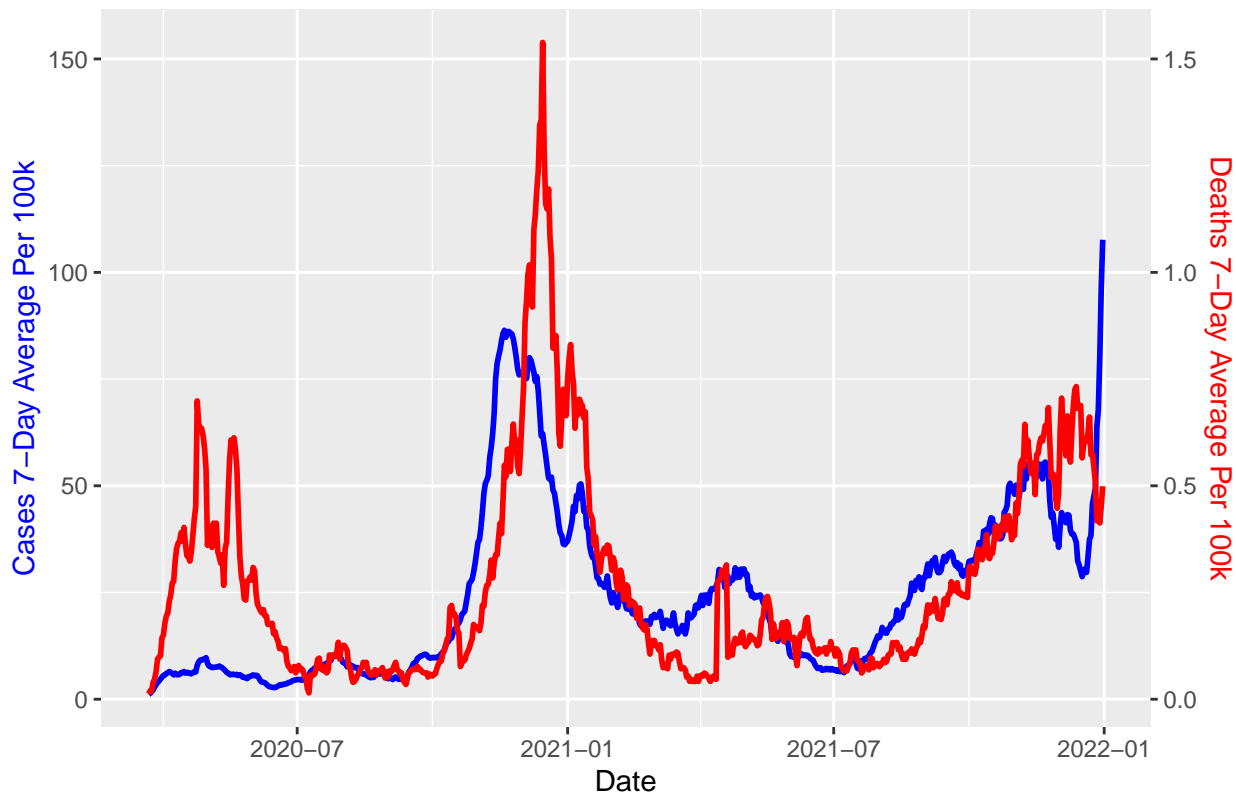
## # A tibble: 657 x 9
## # Groups:   state [1]
##   state   date      total_deaths total_cases population deaths_per_100k
##   <chr>   <date>         <dbl>         <dbl>         <dbl>         <dbl>
## 1 Colorado 2020-03-15           2          136     5784308         0.0346
## 2 Colorado 2020-03-16           2          161     5784308         0.0346
## 3 Colorado 2020-03-17           3          183     5784308         0.0519
## 4 Colorado 2020-03-18           3          216     5784308         0.0519
## 5 Colorado 2020-03-19           5          278     5784308         0.0864
## 6 Colorado 2020-03-20           5          364     5784308         0.0864
## 7 Colorado 2020-03-21           6          475     5784308         0.104
## 8 Colorado 2020-03-22           7          591     5784308         0.121
## 9 Colorado 2020-03-23          10          721     5784308         0.173
## 10 Colorado 2020-03-24          11          912     5784308         0.190
## # i 647 more rows

```

```
## # i 3 more variables: cases_per_100k <dbl>, deaths_7_day <dbl>,  
## #   cases_7_day <dbl>
```

```
# plot the 7-day average  
colorado_7day_avg %>%  
  filter(!is.na(cases_7_day)) %>%  
  ggplot(aes(x = date)) +  
  geom_line(aes(y = cases_7_day, color = "Cases"), linewidth = 1) +  
  # scale for visibility  
  geom_line(aes(y = deaths_7_day * 100, color = "Deaths"), linewidth = 1) +  
  # set up y axis for 2 variables  
  scale_y_continuous(name = "Cases 7-Day Average Per 100k",  
    # add second axis for deaths  
    sec.axis = sec_axis(~./100, name = "Deaths 7-Day Average Per 100k")) +  
  # color the y axis titles to match the plotted lines  
  theme(axis.title.y.left = element_text(color = "blue"),  
    axis.title.y.right = element_text(color = "red")) +  
  labs(title = "Colorado 7-Day Average of Cases and Deaths per 100,000 People",  
    x = "Date") +  
  # remove legend name  
  theme(legend.position = "none") +  
  scale_color_manual(values = c("Cases" = "blue", "Deaths" = "red"))
```

Colorado 7-Day Average of Cases and Deaths per 100,000 People



```
# Your transformed data should look similar to the following tibble:
```

```
#  
# A tibble: 656 × 9  
#   state   date      total_deaths total_cases population deaths_per_100k cases_per_100k deaths_7_day ca  
#   <chr>   <date>         <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>  
# 1 Colorado 2020-03-15         2        136    5784308     0.0346      2.35      NA  
# 2 Colorado 2020-03-16         2        161    5784308     0.0346      2.78      NA
```

```
# 3 Colorado 2020-03-17      3      183      5784308      0.0519      3.16      NA
# 4 Colorado 2020-03-18      3      216      5784308      0.0519      3.73      NA
# 5 Colorado 2020-03-19      5      278      5784308      0.0864      4.81      NA
# 6 Colorado 2020-03-20      5      364      5784308      0.0864      6.29      NA
# 7 Colorado 2020-03-21      6      475      5784308      0.104      8.21      NA
# 8 Colorado 2020-03-22      7      591      5784308      0.121      10.2     0.0123
# 9 Colorado 2020-03-23     10      721      5784308      0.173      12.5     0.0198
# 10 Colorado 2020-03-24     11      912      5784308      0.190      15.8     0.0198
# ... with 646 more rows
```

Looks like a lot of cases and deaths around the winter holidays at the end of 2021 and 2022 in Colorado.

Question 4

Using the same state, identify the top 5 counties in terms of deaths and cases per 100,000 people.

```
# get totals for each Colorado county
colorado_county_totals <- us_counties %>%
  # filter dates (sample below has 2021-12-20 so I'm filtering to that)
  filter(date == as.Date("2021-12-20"),
         state == "Colorado") %>%
  # need fips in case of duplicate county names
  group_by(county, date, fips) %>%
  select(county, date, fips, deaths, cases)
# most deaths
colorado_county_totals %>% arrange(desc(deaths))
```

```
## # A tibble: 64 x 5
## # Groups:   county, date, fips [64]
##   county    date      fips deaths cases
##   <chr>    <date>    <chr> <dbl> <dbl>
## 1 El Paso  2021-12-20 08041  1355 119772
## 2 Denver   2021-12-20 08031  1065 106747
## 3 Jefferson 2021-12-20 08059  1061  76732
## 4 Adams    2021-12-20 08001  1057  90476
## 5 Arapahoe 2021-12-20 08005  1046  95769
## 6 Pueblo   2021-12-20 08101   643  30739
## 7 Weld     2021-12-20 08123   569  55599
## 8 Mesa     2021-12-20 08077   445  29542
## 9 Larimer  2021-12-20 08069   393  47444
## 10 Douglas 2021-12-20 08035   361  48740
## # i 54 more rows
```

```
# most cases
colorado_county_totals %>% arrange(desc(cases))
```

```
## # A tibble: 64 x 5
## # Groups:   county, date, fips [64]
##   county    date      fips deaths cases
##   <chr>    <date>    <chr> <dbl> <dbl>
## 1 El Paso  2021-12-20 08041  1355 119772
## 2 Denver   2021-12-20 08031  1065 106747
## 3 Arapahoe 2021-12-20 08005  1046  95769
## 4 Adams    2021-12-20 08001  1057  90476
## 5 Jefferson 2021-12-20 08059  1061  76732
## 6 Weld     2021-12-20 08123   569  55599
```

```
## 7 Douglas      2021-12-20 08035      361 48740
## 8 Larimer      2021-12-20 08069      393 47444
## 9 Boulder      2021-12-20 08013      323 36754
## 10 Pueblo      2021-12-20 08101      643 30739
## # i 54 more rows
```

```
# samples below don't account for per 100k people
# so I don't know why the question asks for it
# but here is how the data would look like per 100k people

# create Colorado counties population tibble
co_counties_population2021 <- us_population_estimates %>%
  filter(Year == 2021, STNAME == "Colorado") %>%
  mutate(CTYNAME = gsub(" County", "", CTYNAME))
# per 100k Colorado county totals
colorado_county_totals_per_100k <- colorado_county_totals %>%
  # join to population tibble on county names
  left_join(co_counties_population2021, by = c("county" = "CTYNAME")) %>%
  # get per 100k columns
  mutate(deaths_per_100k = (deaths / Estimate) * 100000,
         cases_per_100k = (cases / Estimate) * 100000,
         fips = fips.x) %>%
  # select columns we want
  select(county, date, fips, deaths_per_100k, cases_per_100k)
# most deaths per 100k
colorado_county_totals_per_100k %>% arrange(desc(deaths_per_100k))
```

```
## # A tibble: 64 x 5
## # Groups:   county, date [64]
##   county      date      fips deaths_per_100k cases_per_100k
##   <chr>      <date>    <chr>      <dbl>      <dbl>
## 1 Bent       2021-12-20 08011          660.      33721.
## 2 Otero      2021-12-20 08089          613.      17801.
## 3 Conejos    2021-12-20 08021          578.      15449.
## 4 Washington 2021-12-20 08121          494.      15676.
## 5 Cheyenne   2021-12-20 08017          469.      14294.
## 6 Crowley    2021-12-20 08025          449.      41434.
## 7 Logan      2021-12-20 08075          442.      24354.
## 8 Morgan     2021-12-20 08087          421.      14365.
## 9 Pueblo     2021-12-20 08101          379.      18122.
## 10 Moffat    2021-12-20 08081          364.      17004.
## # i 54 more rows
```

```
# most cases per 100k
colorado_county_totals_per_100k %>% arrange(desc(cases_per_100k))
```

```
## # A tibble: 64 x 5
## # Groups:   county, date [64]
##   county      date      fips deaths_per_100k cases_per_100k
##   <chr>      <date>    <chr>      <dbl>      <dbl>
## 1 Crowley    2021-12-20 08025          449.      41434.
## 2 Bent       2021-12-20 08011          660.      33721.
## 3 Lincoln    2021-12-20 08073          141.      24947.
## 4 Logan      2021-12-20 08075          442.      24354.
## 5 Fremont    2021-12-20 08043          306.      21065.
## 6 Pitkin     2021-12-20 08097           34.6      19593.
## 7 Rio Blanco 2021-12-20 08103          170.      19456.
## 8 Kiowa      2021-12-20 08061          275.      19146.
```

```
## 9 Mesa      2021-12-20 08077      283.      18776.
## 10 Alamosa  2021-12-20 08003      314.      18324.
## # i 54 more rows
```

```
# Your transformed data should be similar to the following tibbles:
```

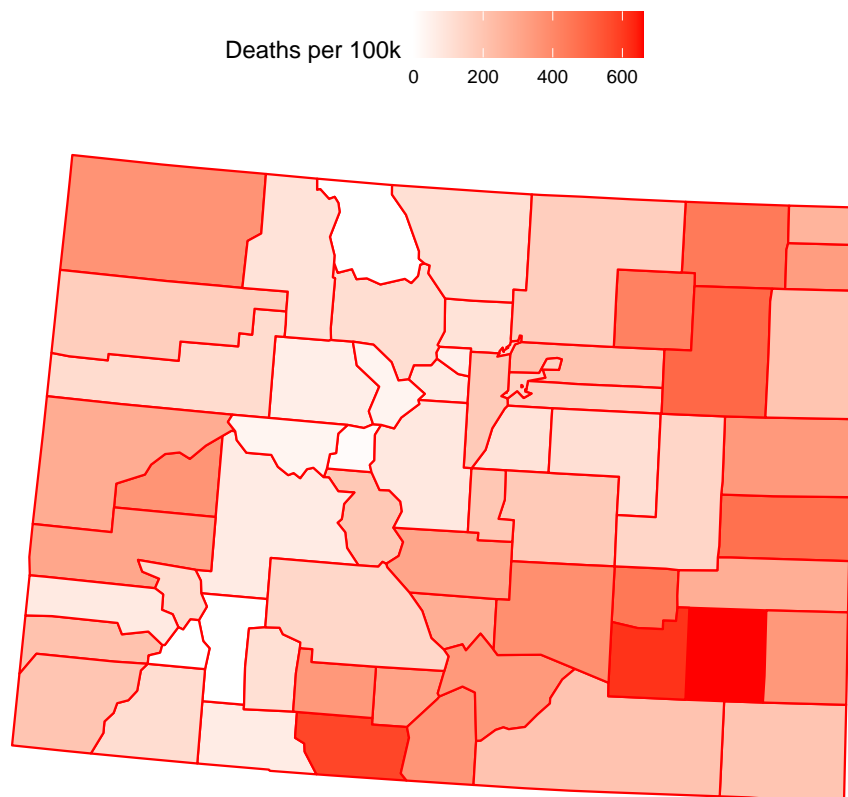
```
#
# Arranged by deaths:
# A tibble: 64 × 4
#   county      date      fips  total_deaths  total_cases
#   <chr>    <date>    <chr>      <dbl>      <dbl>
# 1 El Paso  2021-12-20  08041      1355      119772
# 2 Denver  2021-12-20  08031      1065      106747
# 3 Jefferson 2021-12-20  08059      1061      76732
# 4 Adams    2021-12-20  08001      1057      90476
# 5 Arapahoe 2021-12-20  08005      1046      95769
# 6 Pueblo   2021-12-20  08101       643      30739
# 7 Weld     2021-12-20  08123       569      55599
# 8 Mesa     2021-12-20  08077       445      29542
# 9 Larimer  2021-12-20  08069       393      47444
# 10 Douglas 2021-12-20  08035       361      48740
# ... with 54 more rows
#
#
# Arranged by cases:
# A tibble: 64 × 4
#   county      date      fips  total_deaths  total_cases
#   <chr>    <date>    <chr>      <dbl>      <dbl>
# 1 El Paso  2021-12-20  08041      1355      119772
# 2 Denver  2021-12-20  08031      1065      106747
# 3 Arapahoe 2021-12-20  08005      1046      95769
# 4 Adams    2021-12-20  08001      1057      90476
# 5 Jefferson 2021-12-20  08059      1061      76732
# 6 Weld     2021-12-20  08123       569      55599
# 7 Douglas  2021-12-20  08035       361      48740
# 8 Larimer  2021-12-20  08069       393      47444
# 9 Boulder  2021-12-20  08013       323      36754
# 10 Pueblo  2021-12-20  08101       643      30739
# ... with 54 more rows
```

Arapahoe county had more cases and less deaths than both Jefferson and Adams counties. However, when actually looking at the per 100k people data, we can see counties that are not in the sample lists.

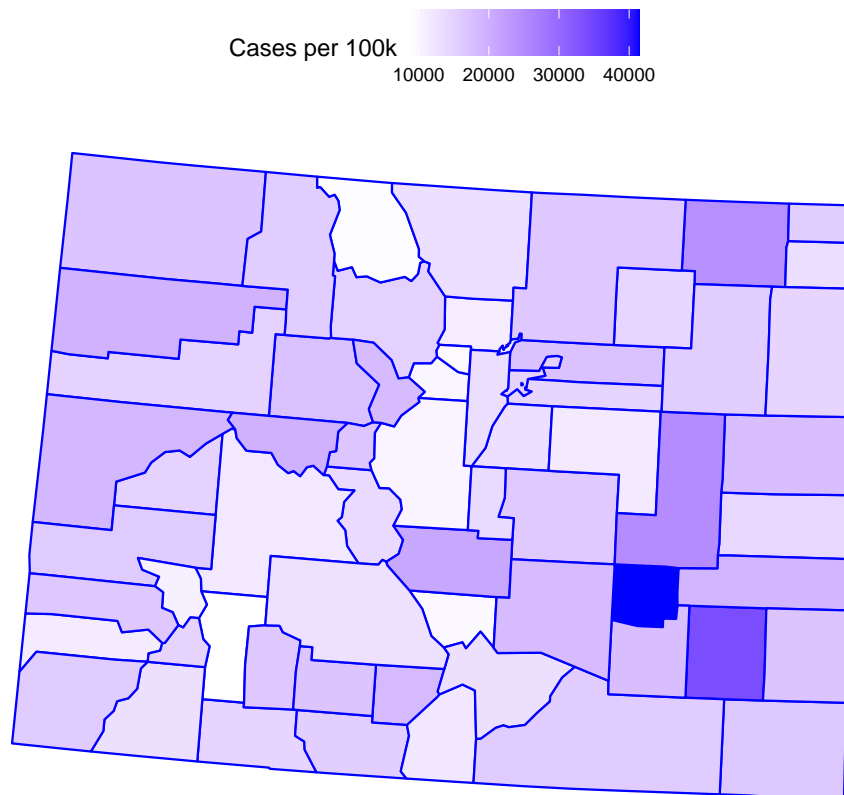
Question 5

Modify the code below for the map projection to plot county-level deaths and cases per 100,000 people for your state.

```
# plot deaths per 100k per county in Colorado
plot_usmap(regions = "county",
            include="CO",
            data = colorado_county_totals_per_100k,
            values = "deaths_per_100k",
            color = "red") +
  scale_fill_continuous(low = "white",
                        high = "red",
                        name = "Deaths per 100k") +
  theme(legend.position = "top")
```



```
# plot cases per 100k per county in Colorado
plot_usmap(regions = "county",
            include="CO",
            data = colorado_county_totals_per_100k,
            values = "cases_per_100k",
            color = "blue") +
scale_fill_continuous(low = "white",
                      high = "blue",
                      name = "Cases per 100k") +
theme(legend.position = "top")
```



```
# Copy and modify the code below for your state.
#
# plot_usmap arguments:
#   regions: can be one of ("states", "state", "counties", "county"). The default is "states"
#   include: The regions to include in the resulting map. If regions is "states"/"state", the value can be ei
#   data: values to plot on the map
#   values: the name of the column that contains the values to be associated with a given region.
#   color: the map outline color.
#
# Reference the plot_usmap documentation for further information using ?plot_usmap

# plot_usmap(regions = "county", include="CO", data = colorado_county, values = "total_deaths", color = "blue
#   scale_fill_continuous(low = "white", high = "blue", name = "Deaths per 100,000")
```

It can be seen that the deaths and cases per 100k people is high for counties that are not as populous as major metros like El Paso county.

Question 6

Finally, select three other states and calculate the seven-day averages for new deaths and cases per 100,000 people for between March 15, 2020, and December 31, 2021.

```
# create a function so that I'm not copy/pasting too much
state_7day_avg <- function(us_state) {
  us_counties %>%
    # filter dates
    filter(between(date, as.Date("2020-03-15"), as.Date("2021-12-31")),
           state == us_state) %>%
    group_by(state, date) %>%
```



```

# sum up deaths and cases
summarise(total_deaths = sum(deaths),
          total_cases = sum(cases)) %>%
# add tear to join on
mutate(year = year(date)) %>%
left_join(
  # filter to just Colorado and join on year
  filter(state_population_totals, STNAME == us_state),
  by = c("year" = "Year")) %>%
mutate(
  # create 1-day deltas to create 7-day averages
  delta_deaths_1 = total_deaths - lag(total_deaths),
  delta_cases_1 = total_cases - lag(total_cases),
  # create the per 100k metrics
  deaths_per_100k = (total_deaths / total_population) * 100000,
  cases_per_100k = (total_cases / total_population) * 100000,
  # re-use function for 7-day rolling average
  deaths_7_day = calculate_rolling_average((delta_deaths_1 / total_population) * 100000, 7),
  cases_7_day = calculate_rolling_average((delta_cases_1 / total_population) * 100000, 7),
  # shorten name of column
  population = total_population) %>%
# grab only the needed columns
select(state,
       date,
       total_deaths,
       total_cases,
       population,
       deaths_per_100k,
       cases_per_100k,
       deaths_7_day,
       cases_7_day)
}
# California 7-day average
state_7day_avg("California")

```

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

```

## # A tibble: 657 x 9
## # Groups:   state [1]
##   state      date      total_deaths total_cases population deaths_per_100k
##   <chr>    <date>          <dbl>         <dbl>      <dbl>         <dbl>
## 1 California 2020-03-15           6           478    39499738         0.0152
## 2 California 2020-03-16          11           588    39499738         0.0278
## 3 California 2020-03-17          14           732    39499738         0.0354
## 4 California 2020-03-18          17           893    39499738         0.0430
## 5 California 2020-03-19          19          1067    39499738         0.0481
## 6 California 2020-03-20          24          1283    39499738         0.0608
## 7 California 2020-03-21          28          1544    39499738         0.0709
## 8 California 2020-03-22          35          1851    39499738         0.0886
## 9 California 2020-03-23          39          2240    39499738         0.0987
## 10 California 2020-03-24          52          2644    39499738         0.132
## # i 647 more rows
## # i 3 more variables: cases_per_100k <dbl>, deaths_7_day <dbl>,
## #   cases_7_day <dbl>

```

```
# Oregon 7-day average
state_7day_avg("Oregon")
```

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

```
## # A tibble: 657 x 9
## # Groups:   state [1]
##   state date      total_deaths total_cases population deaths_per_100k
##   <chr> <date>          <dbl>         <dbl>         <dbl>         <dbl>
## 1 Oregon 2020-03-15          1           39      4241544          0.0236
## 2 Oregon 2020-03-16          1           46      4241544          0.0236
## 3 Oregon 2020-03-17          2           66      4241544          0.0472
## 4 Oregon 2020-03-18          3           74      4241544          0.0707
## 5 Oregon 2020-03-19          3           87      4241544          0.0707
## 6 Oregon 2020-03-20          3          114      4241544          0.0707
## 7 Oregon 2020-03-21          4          137      4241544          0.0943
## 8 Oregon 2020-03-22          5          161      4241544          0.118
## 9 Oregon 2020-03-23          5          191      4241544          0.118
## 10 Oregon 2020-03-24          8          209      4241544          0.189
## # i 647 more rows
## # i 3 more variables: cases_per_100k <dbl>, deaths_7_day <dbl>,
## #   cases_7_day <dbl>
```

```
# Washington 7-day average
state_7day_avg("Washington")
```

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

```
## # A tibble: 657 x 9
## # Groups:   state [1]
##   state date      total_deaths total_cases population deaths_per_100k
##   <chr> <date>          <dbl>         <dbl>         <dbl>         <dbl>
## 1 Washington 2020-03-15          42           675      7718785          0.544
## 2 Washington 2020-03-16          48           794      7718785          0.622
## 3 Washington 2020-03-17          54           908      7718785          0.700
## 4 Washington 2020-03-18          68          1026      7718785          0.881
## 5 Washington 2020-03-19          75          1228      7718785          0.972
## 6 Washington 2020-03-20          83          1404      7718785          1.08
## 7 Washington 2020-03-21          95          1655      7718785          1.23
## 8 Washington 2020-03-22          97          1844      7718785          1.26
## 9 Washington 2020-03-23         110          2101      7718785          1.43
## 10 Washington 2020-03-24         123          2469      7718785          1.59
## # i 647 more rows
## # i 3 more variables: cases_per_100k <dbl>, deaths_7_day <dbl>,
## #   cases_7_day <dbl>
```

Creating a function from the code I used originally for Colorado (refer to Question 3) made things quick and easy.

Question 7

Create a visualization comparing the seven-day averages for new deaths and cases per 100,000 people for the four states you selected.

```

# functions for the win
plot_state_7day_avg <- function(us_state) {
  # reusing function from Question 6
  state_7day_avg(us_state) %>%
  filter(!is.na(cases_7_day)) %>%
  ggplot(aes(x = date)) +
  geom_line(aes(y = cases_7_day, color = "Cases"), linewidth = 1) +
  # scale for visibility
  geom_line(aes(y = deaths_7_day * 100, color = "Deaths"), linewidth = 1) +
  # set up y axis for 2 variables
  scale_y_continuous(name = "Cases 7-Day Average Per 100k",
    # add second axis for deaths
    sec.axis = sec_axis(~./100, name = "Deaths 7-Day Average Per 100k")) +
  # color the y axis titles to match the plotted lines
  theme(axis.title.y.left = element_text(color = "blue"),
    axis.title.y.right = element_text(color = "red")) +
  labs(title = paste(us_state, "7-Day Average of Cases and Deaths per 100,000 People"),
    x = "Date") +
  # remove legend name
  theme(legend.position = "none") +
  scale_color_manual(values = c("Cases" = "blue", "Deaths" = "red"))
}

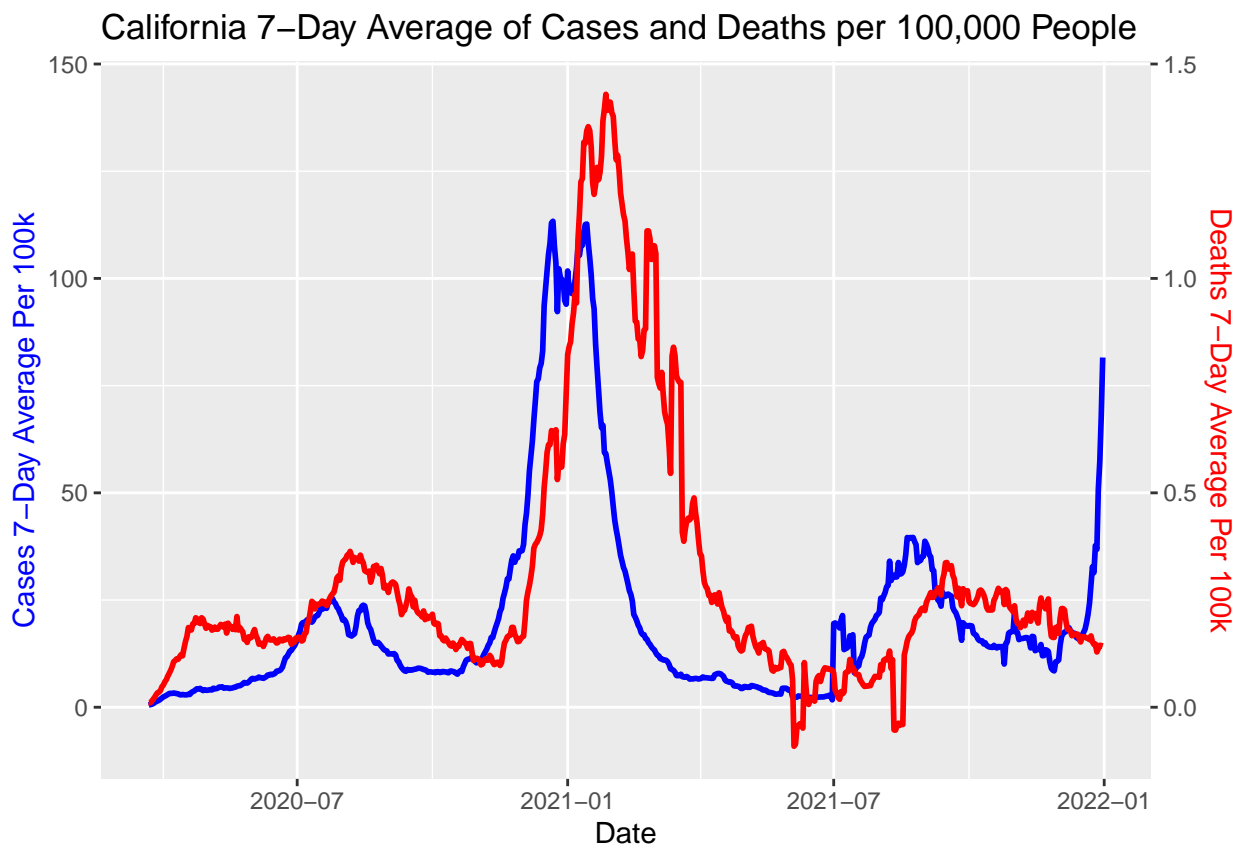
# it says four states even though Question 6 said 3,
# so I'll add another state besides Colorado as it was done in Question 3
# California
plot_state_7day_avg("California")

```

```

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

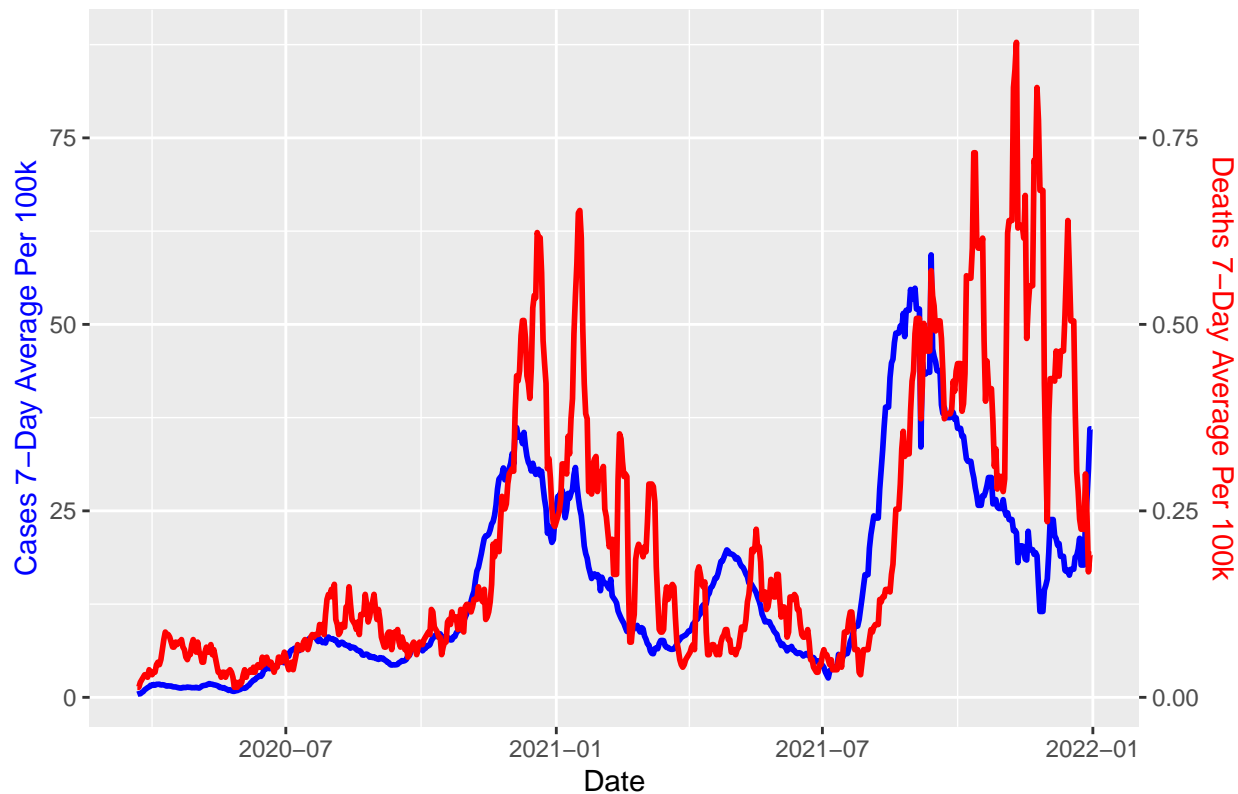
```



```
# Oregon
plot_state_7day_avg("Oregon")
```

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

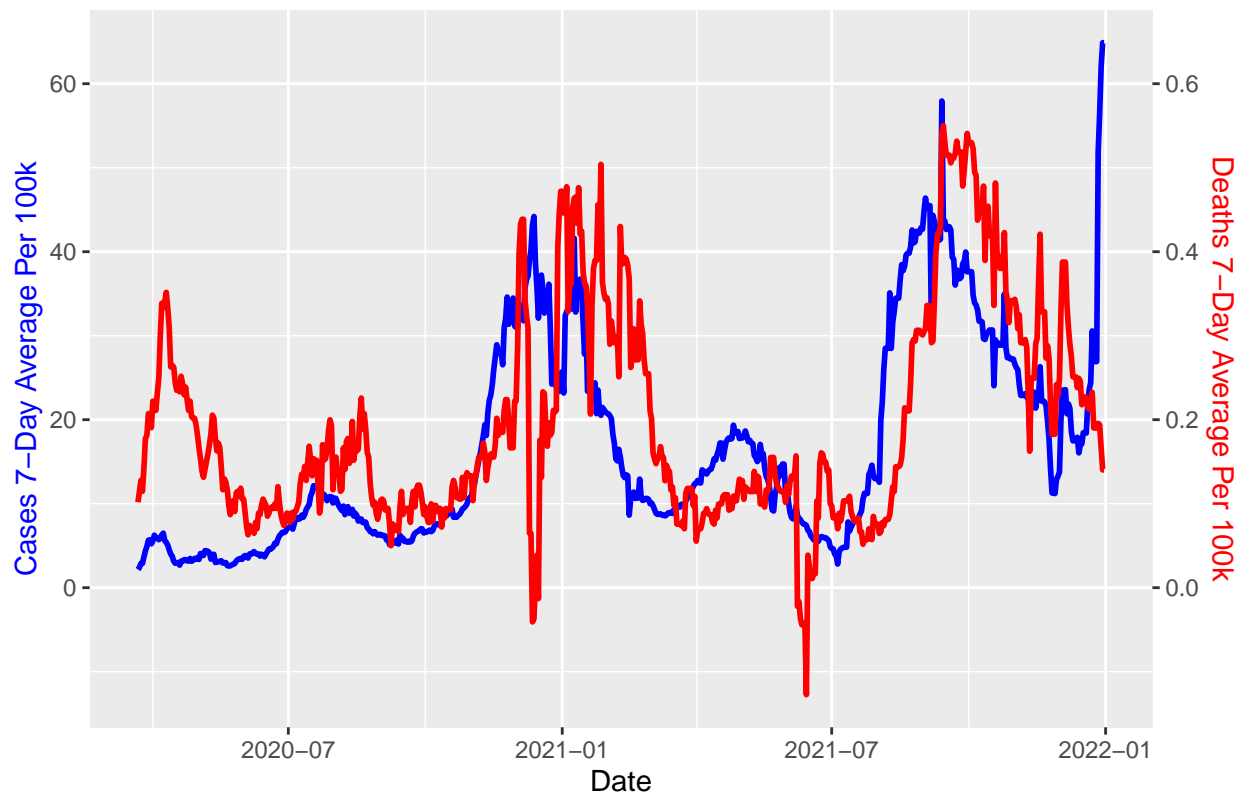
Oregon 7-Day Average of Cases and Deaths per 100,000 People



```
# Washington
plot_state_7day_avg("Washington")
```

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

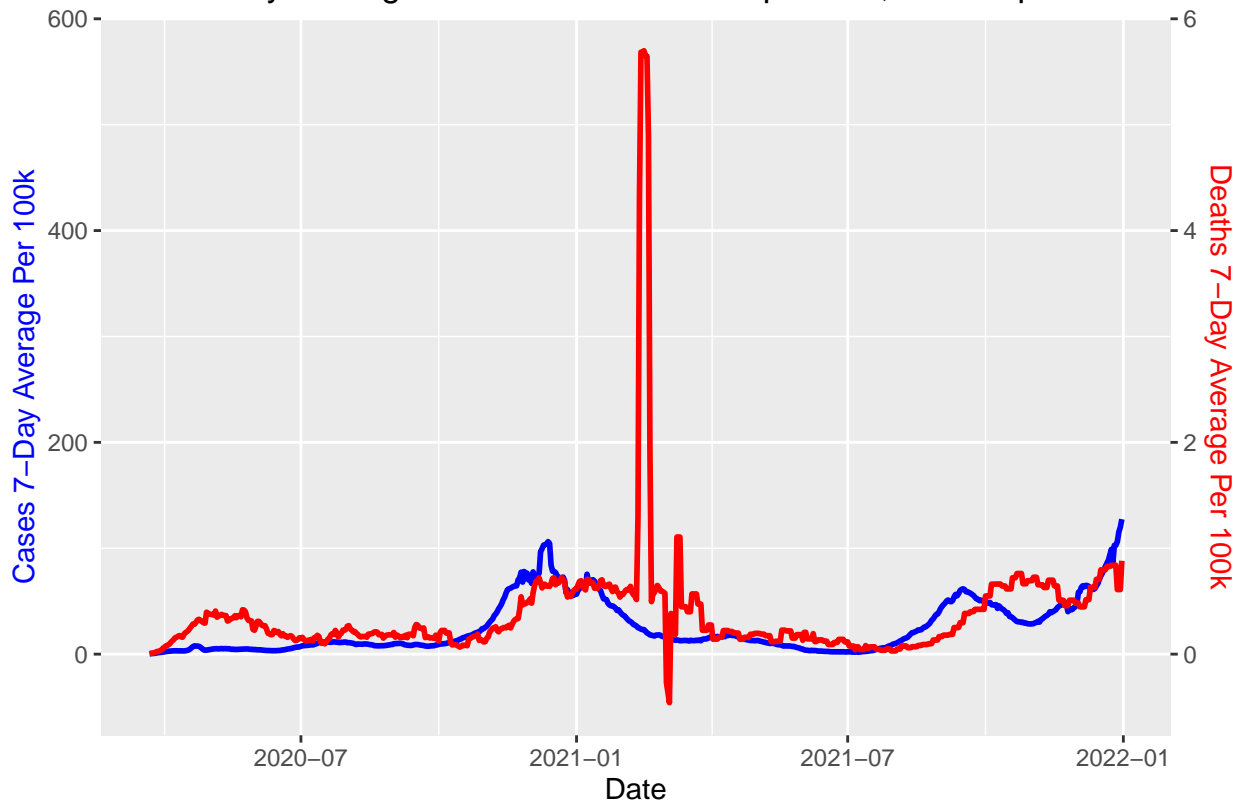
Washington 7-Day Average of Cases and Deaths per 100,000 People



```
# Ohio  
plot_state_7day_avg("Ohio")
```

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

Ohio 7-Day Average of Cases and Deaths per 100,000 People



Functions really help a lot and the plots give a good visual to see the differences in cases and deaths for each of the states.

Part 3 - Global Comparison

```
# Import global COVID-19 statistics aggregated by the Center for Systems Science and Engineering (CSSE) at Jo
# Import global population estimates from the World Bank.

#csse_global_deaths <- read_csv("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_1
#csse_global_cases <- read_csv("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_1
#csse_us_deaths <- read_csv("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_d
#csse_us_cases <- read_csv("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_da

#global_population_estimates <- read_csv("global_population_estimates.csv")
```

Question 1

Using the state you selected in Part 2 Question 2 compare the daily number of cases and deaths reported from the CSSE and NY Times.

```
# To compare your state data between the two data sets, you will first need to tidy the US CSSE death and cas
# Hint: Review the documentation for pivot_longer().

# Once you have tidied your data, join the two CSSE US data sets to include cases and deaths in one table.

# Finally, create two visualizations with one plotting the CSSE and NY Times cases and the other plotting the
```

```
# Your tidied CSSE data for your selected state should look similar to the following tibble:
#
# A tibble: 43,362 × 6
#   fips county state      date    cases  deaths
#   <dbl> <chr>   <chr>   <date>   <dbl>  <dbl>
# 1  8001 Adams Colorado 2020-03-15      6      0
# 2  8001 Adams Colorado 2020-03-16      8      0
# 3  8001 Adams Colorado 2020-03-17     10      0
# 4  8001 Adams Colorado 2020-03-18     10      0
# 5  8001 Adams Colorado 2020-03-19     10      0
# 6  8001 Adams Colorado 2020-03-20     12      0
# 7  8001 Adams Colorado 2020-03-21     14      0
# 8  8001 Adams Colorado 2020-03-22     18      0
# 9  8001 Adams Colorado 2020-03-23     25      0
# 10 8001 Adams Colorado 2020-03-24     27      0
# ... with 43,352 more rows
```

– Communicate your methodology, results, and interpretation here –

Question 2

Now that you have verified the data reported from the CSSE and NY Times are similar, combine the global and US CSSE data sets and identify the top 10 countries in terms of deaths and cases per 100,000 people between March 15, 2020, and December 31, 2021.

```
# First, combine and tidy the CSSE death and cases data sets. You may wish to keep the two sets separate.
# Then, tidy the global population estimates. While tidying your data, remember to include columns that you w
# You will notice that the population estimates data does not include every country reported in the CSSE data
```

– Communicate your methodology, results, and interpretation here –

Question 3

Construct a visualization plotting the 10 countries in terms of deaths and cases per 100,000 people between March 15, 2020, and December 31, 2021. In designing your visualization keep the number of data you will be plotting in mind. You may wish to create two separate visualizations, one for deaths and another for cases.

– Communicate your methodology, results, and interpretation here –

Question 4

Finally, select four countries from one continent and create visualizations for the daily number of confirmed cases per 100,000 and the daily number of deaths per 100,000 people between March 15, 2020, and December 31, 2021.

– Communicate your methodology, results, and interpretation here –