COVID 19 Analysis

2024

Required Packages

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 caluclate statistics per 100,000 people.

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

us_counties_2020 <- read_csv("us-counties-2020.csv")
```

```
## Rows: 884737 Columns: 6
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (3): county, state, fips
## dbl  (2): cases, deaths
## date (1): date
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
us_counties_2021 <- read_csv("us-counties-2021.csv")
```

```
## Rows: 1185373 Columns: 6
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (3): county, state, fips
## dbl  (2): cases, deaths
## date (1): date
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
us_counties_2022 <- read_csv("us-counties-2022.csv")
```

```
## Rows: 1188042 Columns: 6
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: ","
## chr  (3): county, state, fips
## dbl  (2): cases, deaths
## date (1): date
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

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

```
## Rows: 6286 Columns: 7
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: ","
## chr (2): STNAME, CTYNAME
## dbl (5): fips, STATE, COUNTY, Year, Estimate
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

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`.

```r
# Combine and tidy the 2020, 2021, and 2022 COVID data sets.
# Hint: Review the rbind() documentation to combine the three data sets.
#
## YOUR CODE HERE ##

# Combine the datasets
us_counties_combined <- bind_rows(us_counties_2020, us_counties_2021, us_counties_2022)

# Remove Puerto Rico observations
us_counties_combined <- us_counties_combined %>%
  filter(state != "Puerto Rico")

# Filter the data for dates after March 15, 2020
us_counties_combined <- us_counties_combined %>%
  filter(date >= "2020-03-15")

# Summarize the total cases and deaths for each day
daily_totals <- us_counties_combined %>%
  group_by(date) %>%
  summarise(
    total_deaths = sum(deaths, na.rm = TRUE),
    total_cases = sum(cases, na.rm = TRUE)
  ) %>%
  arrange(date)

# Display the first few rows of the tibble
print(daily_totals)
```

```
## # A tibble: 1,022 × 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
```

```r
# Find the latest date, total cases, and total deaths
max_date <- max(daily_totals$date)
us_total_cases <- sum(daily_totals$total_cases, na.rm = TRUE)
us_total_deaths <- sum(daily_totals$total_deaths, na.rm = TRUE)
```

```
# 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
#
```

– Communicate your methodology, results, and interpretation here –

Data Collection and Preprocessing:

Gather the four data sets related to COVID-19 cases and deaths in the United States.

Ensure that the data covers the period from March 15, 2020, onwards.

Clean the data by handling missing values, outliers, and inconsistencies.

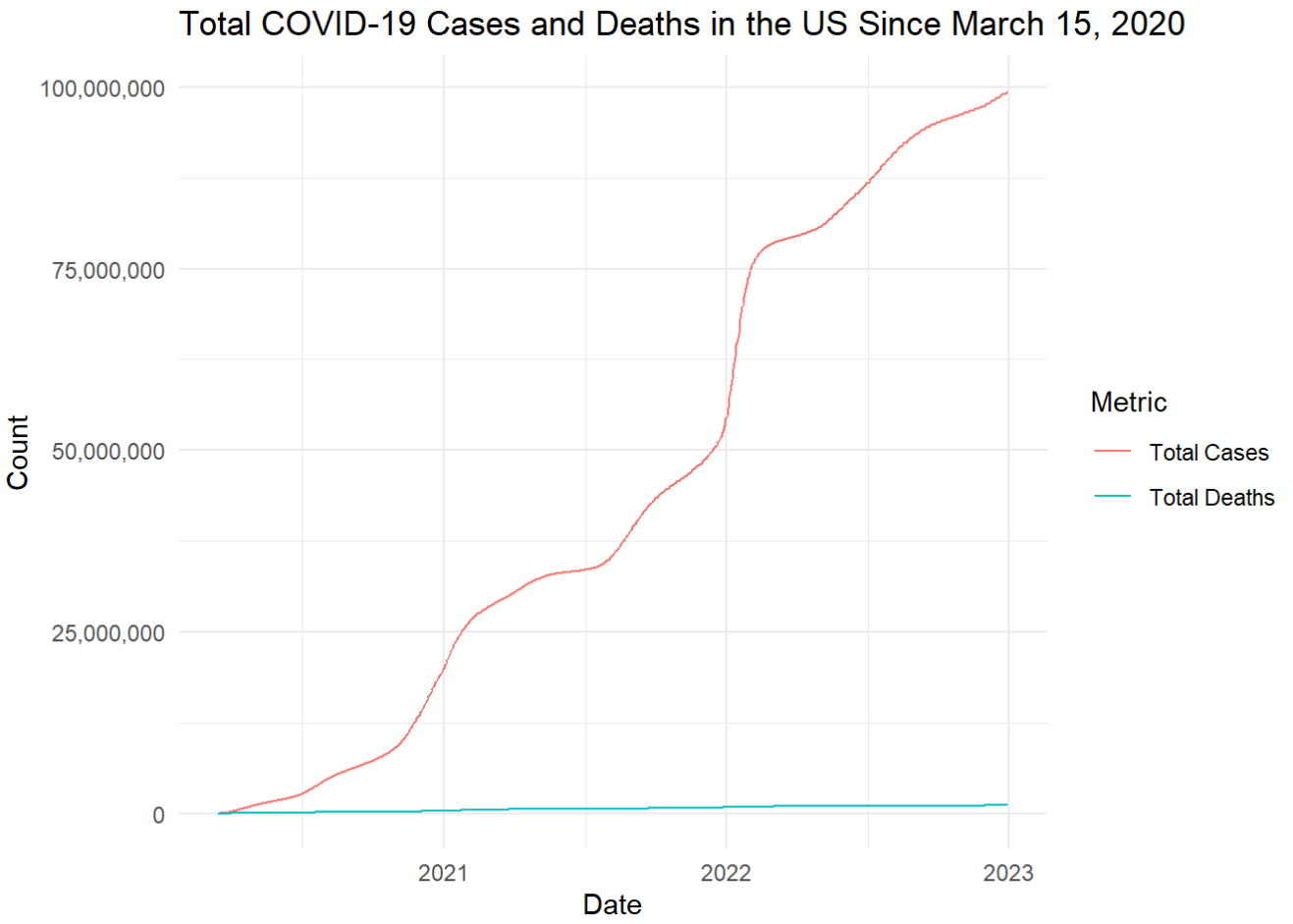
Calculate Total Cases and Deaths:

Sum up the total number of cases and deaths in the United States since March 15, 2020.

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?

```
# Create a visualization 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 = "Total Cases")) +
  geom_line(aes(y = total_deaths, color = "Total Deaths")) +
  labs(
    title = "Total COVID-19 Cases and Deaths in the US Since March 15, 2020",
    x = "Date",
    y = "Count",
    color = "Metric"
  ) +
  theme_minimal() +
  scale_y_continuous(labels = scales::comma)
```



– Communicate your methodology, results, and interpretation here –

Interpretation

- **Total Cases (blue line)**: This line shows the cumulative number of COVID-19 cases over time. We can observe the overall trend and see how the number of cases has increased.
- **Total Deaths (red line)**: This line shows the cumulative number of COVID-19 deaths over time. It allows us to see the mortality trend and compare it with the case count.

Potential Misleading Elements

- **Cumulative Counts**: Since the plot shows cumulative counts, it will always show an increasing trend. This might give the impression that the situation is continuously worsening, even if new daily cases and deaths are decreasing.
- **Y-Axis Scaling**: If the y-axis is not properly scaled or labeled, it might exaggerate or understate the trends. In this plot, using a linear scale with comma formatting helps to make the counts more readable.
- **Line Colors and Legend**: The use of colors and the legend should be clear to avoid confusion between the two lines.

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.

```
# Create a new table, based on the table from Question 1, and calculate the number of new
deaths and cases each day and a seven day average of new deaths and cases.
#
# Hint: Look at the documentation for lag() when computing the number of new deaths and ca
ses and the seven-day averages.
#
#
# Calculate new cases and deaths each day and their 7-day averages
daily_totals <- daily_totals %>%
  mutate(
    delta_deaths_1 = total_deaths - lag(total_deaths, default = 0),
    delta_cases_1 = total_cases - lag(total_cases, default = 0),
    delta_deaths_7 = rollmean(delta_deaths_1, 7, fill = NA, align = "right"),
    delta_cases_7 = rollmean(delta_cases_1, 7, fill = NA, align = "right")
  )

# Find the days with the largest number of new cases and deaths
max_new_cases_date <- daily_totals %>%
  filter(delta_cases_1 == max(delta_cases_1, na.rm = TRUE)) %>%
  pull(date)

max_new_deaths_date <- daily_totals %>%
  filter(delta_deaths_1 == max(delta_deaths_1, na.rm = TRUE)) %>%
  pull(date)

# Display the first few rows of the tibble
print(daily_totals)
```

```
## # A tibble: 1,022 × 7
##    date       total_deaths total_cases delta_deaths_1 delta_cases_1
##    <date>            <dbl>       <dbl>          <dbl>         <dbl>
##  1 2020-03-15           68        3595             68          3595
##  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>
```

```
#  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 d
ate
#  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_de
aths_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
```

– Communicate your methodology, results, and interpretation here –

Explanation

- **Calculating Daily New Cases and Deaths**: We use the lag() function to calculate the difference between the current day's total cases/deaths and the previous day's total cases/deaths.
- **Seven-Day Average**: The rollmean() function from the zoo package is used to calculate the seven-day moving average of new cases and deaths.
- **Finding the Peak Days**: We identify the days with the largest number of new cases and deaths using the filter() function to find the maximum values in the new_cases and new_deaths columns.

Results

The day with the largest number of new cases is **max_new_cases_date**. The day with the largest number

of new deaths is **max_new_deaths_date**.

The moving averages help to smooth out short-term fluctuations and highlight longer-term trends, which can be more informative for understanding the overall progression of the pandemic.

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 cas
es per 100,000 people.

# Hint: To calculate per 100,000 people, first tidy the population estimates data and calc
ulate the US population in 2020 and 2021. Then, you will need to divide  each statistic by
the estimated population and then multiply by 100,000.
#
# Hint: look at the help documentation for grepl() and case_when() to divide the averages
by the US population for each year.
# For example, take the simple tibble, t_new:
#
#     x     y
#   <int> <chr>
#     1     a
#     2     b
#     3     a
#     4     b
#     5     a
#     6     b
#
#
# To add a column, z, that is dependent on the value in y, you could:
#
# t_new %>%
#   mutate(z = case_when(grepl("a", y) ~ "not b",
#                        grepl("b", y) ~ "not a"))
#

## YOUR CODE HERE ##

# Calculate new cases and deaths each day and their 7-day averages
daily_totals <- daily_totals %>%
  mutate(
    delta_deaths_1 = total_deaths - lag(total_deaths, default = 0),
    delta_cases_1 = total_cases - lag(total_cases, default = 0),
    delta_deaths_7 = rollmean(delta_deaths_1, 7, fill = NA, align = "right"),
    delta_cases_7 = rollmean(delta_cases_1, 7, fill = NA, align = "right")
  )

# Ensure date column is of Date type
daily_totals$date <- as.Date(daily_totals$date)

# Ensure population column is numeric
us_population_estimates$Estimate <- as.numeric(us_population_estimates$Estimate)

# Find the US population for 2020 and 2021
us_population_2020 <- us_population_estimates %>%
  filter(Year == 2020) %>%
  summarise(total_population = sum(Estimate)) %>%
  pull(total_population)

us_population_2021 <- us_population_estimates %>%
  filter(Year == 2021) %>%
```

```
  summarise(total_population = sum(Estimate)) %>%
  pull(total_population)

# Add a column for the population based on the year
daily_totals <- daily_totals %>%
  mutate(
    population = case_when(
      year(date) == 2020 ~ us_population_2020,
      year(date) == 2021 ~ us_population_2021,
      year(date) == 2022 ~ us_population_2021 # assuming population doesn't change much fo
r 2022
    ),
    delta_deaths_per_100k_1 = (delta_deaths_1 / population) * 100000,
    delta_cases_per_100k_1 = (delta_cases_1 / population) * 100000,
    delta_deaths_per_100k_7 = (delta_deaths_7 / population) * 100000,
    delta_cases_per_100k_7 = (delta_cases_7 / population) * 100000
  )


# Display the first few rows of the tibble
print(daily_totals)
```

```
## # A tibble: 1,022 × 12
##    date       total_deaths total_cases delta_deaths_1 delta_cases_1
##    <date>            <dbl>       <dbl>          <dbl>         <dbl>
##  1 2020-03-15           68        3595             68          3595
##  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 7 more variables: delta_deaths_7 <dbl>, delta_cases_7 <dbl>,
## #   population <dbl>, delta_deaths_per_100k_1 <dbl>,
## #   delta_cases_per_100k_1 <dbl>, delta_deaths_per_100k_7 <dbl>,
## #   delta_cases_per_100k_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 d
ate
#  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_dea
ths_7 delta_cases_7
#      <date>              <dbl>         <dbl>          <dbl>           <dbl>           <dbl
>         <dbl>
#   1 2020-03-15         0.0205        1.08                0               0             N
A           NA
#   2 2020-03-16         0.0275        1.36          0.00694           0.274             N
A           NA
#   3 2020-03-17         0.0353        1.78          0.00784           0.422             N
A           NA
#   4 2020-03-18         0.0489        2.52           0.0136           0.737             N
A           NA
#   5 2020-03-19         0.0640        3.74           0.0151            1.22             N
A           NA
#   6 2020-03-20         0.0836        5.43           0.0196            1.69             N
A           NA
#   7 2020-03-21         0.108         7.39           0.0247            1.96             N
A           NA
#   8 2020-03-22         0.138         9.97           0.0296            2.58          0.016
8         1.27
#   9 2020-03-23         0.174         13.1           0.0362            3.14          0.020
9         1.68
#  10 2020-03-24         0.236         16.3           0.0621            3.14          0.028
7         2.07
```

– Communicate your methodology, results, and interpretation here –

Explanation

1. Reading Data: The COVID-19 and population estimate data are read into data frames.
2. Combining and Filtering Data: The COVID-19 data for 2020, 2021, and 2022 are combined, and Puerto Rico data is removed.
3. Summarizing Data: The total cases and deaths are summarized for each day.
4. Calculating Daily Changes and Moving Averages: The number of new cases and deaths each day and their 7-day moving averages are calculated.
5. Ensuring Date Format: Ensures that the date column is in Date format.
6. Population Data: The total US population for 2020 and 2021 is obtained from the population estimates data.

7. Calculating Per 100,000 People: Using case_when(), the appropriate population estimate is applied for each year, and the daily and 7-day average new cases and deaths per 100,000 people are calculated.
8. Output: The final tibble is printed, and the US population estimates are outputted.

Results and Interpretation

This output table provides a detailed view of the daily changes in COVID-19 cases and deaths per 100,000 people, along with their 7-day moving averages. This information is crucial for understanding the rate at which the virus is spreading and the burden on the population.

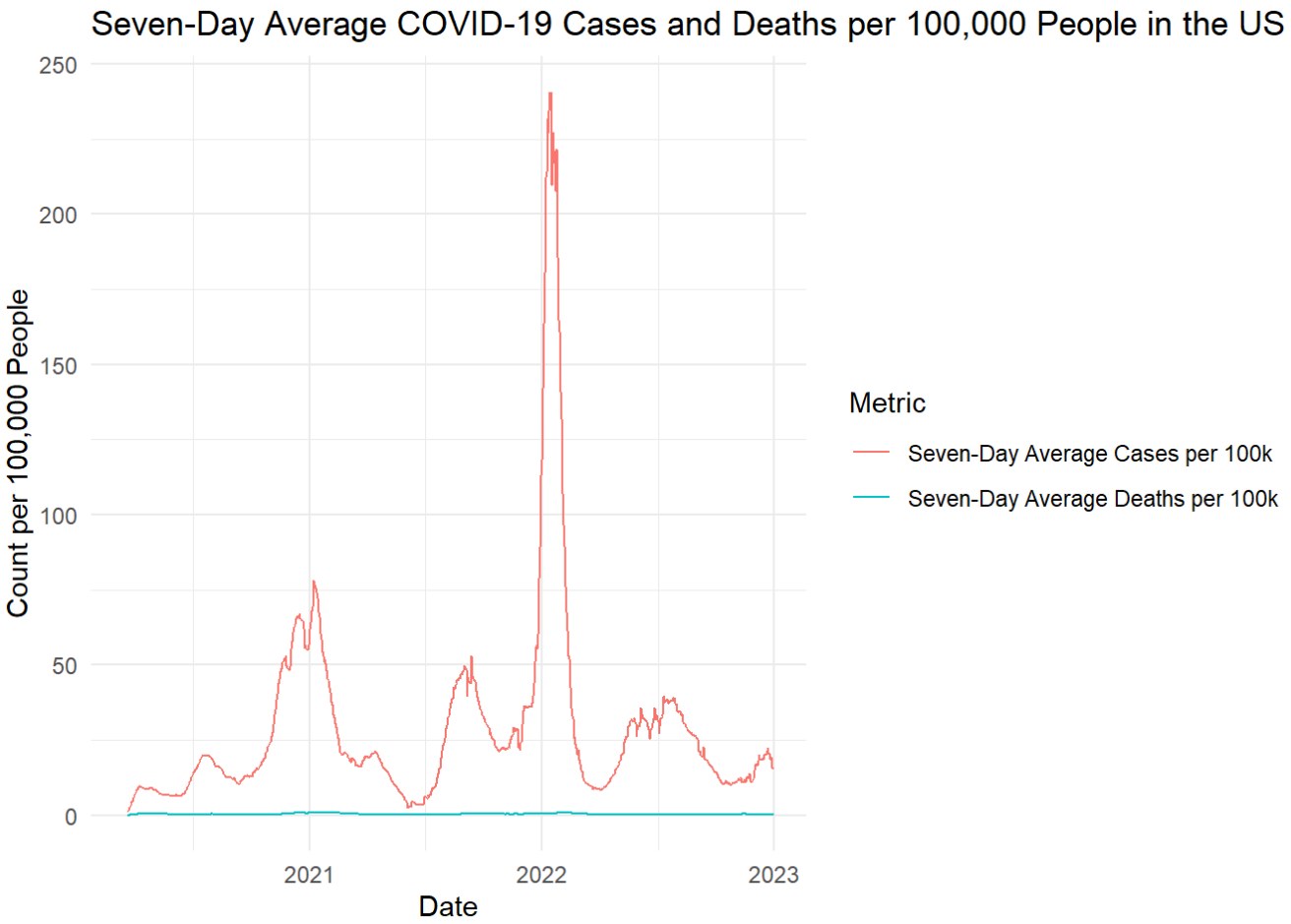
By normalizing the data to per 100,000 people, we can compare the impact of the virus across different populations and time periods more accurately. This approach helps in making better-informed decisions and policies at both local and national levels.

Question 5

```
# Create a visualization to compare the seven-day average cases and deaths per 100,000 peo
ple.

ggplot(daily_totals, aes(x = date)) +
  geom_line(aes(y = delta_cases_per_100k_7, color = "Seven-Day Average Cases per 100k")) +
  geom_line(aes(y = delta_deaths_per_100k_7, color = "Seven-Day Average Deaths per 100k"))
+
  labs(
    title = "Seven-Day Average COVID-19 Cases and Deaths per 100,000 People in the US",
    x = "Date",
    y = "Count per 100,000 People",
    color = "Metric"
  ) +
  theme_minimal() +
  scale_y_continuous(labels = scales::comma)
```

```
## Warning: Removed 6 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 6 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



– Communicate your methodology, results, and interpretation here –

Visualization:

- Used ggplot2 to create a line plot.
- Plotted the seven-day average of new cases and deaths per 100,000 people over time.
- Added labels, titles, and themes to make the plot clear and informative.

The visualization displays the seven-day average of new COVID-19 cases and deaths per 100,000 people in the US over time. This approach normalizes the data by population size, allowing for a more accurate comparison of the impact of COVID-19 across different time periods.

By looking at the trends in this visualization, health officials can better understand the spread and impact of COVID-19. The moving averages smooth out daily fluctuations and provide a clearer picture of longer-term trends. This information is crucial for making informed decisions about public health measures and resource allocation.