DIRECTORATE OF ONLINE EDUCATION
(A unit of Manipal Academy of Higher Education, Manipal)

R PROGRAMMING AND TIDYVERSE CAPSTONE PROJECT

REPORT SUBMITTED TO

MANIPAL ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

in partial fulfilment for the award of the degree

MASTER OF BUSINESS ADMINISTRATION

(Specialization : Business Analytics)

Submitted by:

U Kishore

Reg. No: 22154040466

August 2024

1. Introduction to Coursera Capstone Project Completed

The Coursera Capstone Project centered on the analysis of COVID-19 data, leveraging R programming to explore trends, visualize data, and derive insights at the country level within the United States. The project was divided into multiple parts, each aimed at systematically understanding the spread of COVID-19 and its impact on various countries using publicly available datasets. This comprehensive analysis included data wrangling, statistical calculations, and the creation of visualizations, enabling the application of theoretical knowledge in a real-world scenario.

2. Skills/Techniques Learned in Coursera Capstone Project

Throughout the capstone project, several key skills and techniques were developed:

- **Data Wrangling and Cleaning**: Proficient use of R packages like tidyverse, lubridate, and zoo to clean, transform, and merge datasets.
- **Data Visualization**: Utilized ggplot2 and usmap for creating insightful visualizations, including maps and plots, to represent COVID-19 spread and impact across different regions.
- **Statistical Analysis**: Gained experience in calculating per capita metrics, handling time series data, and performing trend analysis.
- **Geospatial Analysis**: Used FIPS codes and other geospatial identifiers to analyze and visualize county-level data effectively.
- **R Markdown**: Enhanced skills in documenting the analysis process in an organized manner, ensuring reproducibility and clarity.

3. Key Takeaways from the Capstone Project

- **Real-World Application of R**: The project reinforced the practical application of R programming in solving real-world problems, particularly in public health.

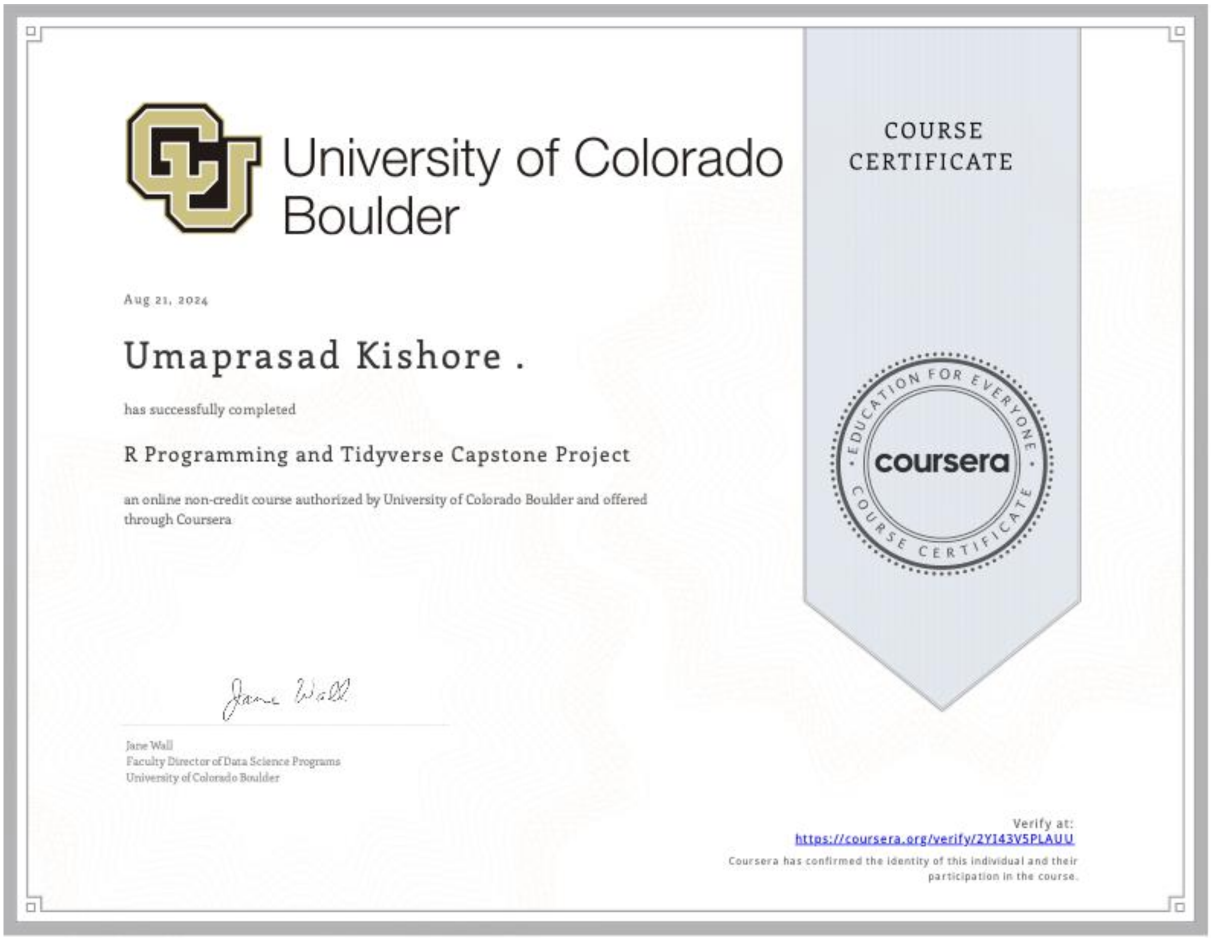
- **Importance of Data Cleaning**: Data cleaning and preprocessing were crucial in ensuring accurate analysis, highlighting the importance of meticulous data handling.
- **Impact of Visualization**: Visualizations played a vital role in conveying complex data trends in an understandable manner, emphasizing the power of visual communication in data analysis.
- **Handling Large Datasets**: The project provided experience in managing and analyzing large datasets, a common scenario in data science.

4. Brief Note on Real-Time Applications of Key Takeaways from This Project

The techniques and skills learned from this capstone project have direct applications in various real-time scenarios:

- **Public Health Monitoring**: The ability to analyze and visualize epidemiological data is crucial in monitoring and responding to public health crises.
- **Policy Decision Support**: The insights derived from data analysis can inform policymakers about trends, enabling more informed decision-making, especially in crisis management.
- **Data-Driven Journalism**: The skills can be applied in data journalism to provide the public with accurate and meaningful insights on critical issues like pandemics.
- **Corporate Analytics**: The data wrangling and visualization techniques are transferable to other domains, such as business intelligence and customer analytics, where understanding trends and patterns is essential.

5. Course Completion Certificate



University of Colorado Boulder

COURSE CERTIFICATE

Aug 21, 2024

Umaprasad Kishore .

has successfully completed

R Programming and Tidyverse Capstone Project

an online non-credit course authorized by University of Colorado Boulder and offered through Coursera

Jane Wall
Faculty Director of Data Science Programs
University of Colorado Boulder

EDUCATION FOR EVERYONE coursera COURSE CERTIFICATE

Verify at:
https://coursera.org/verify/2YI43V5PLAUU

Coursera has confirmed the identity of this individual and their participation in the course.

COVID 19 Analysis

2024

Required Packages

```r
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   3.5.1     ✔ tibble    3.2.1
## ✔ lubridate 1.9.3     ✔ tidyr     1.3.1
## ✔ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(lubridate)
library(usmap)
library(zoo)
```

```
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(readr)
```

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.

```r
# 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.
```

```r
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.
```

```r
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.
```

```r
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`.

```
# 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
```

```
# 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.

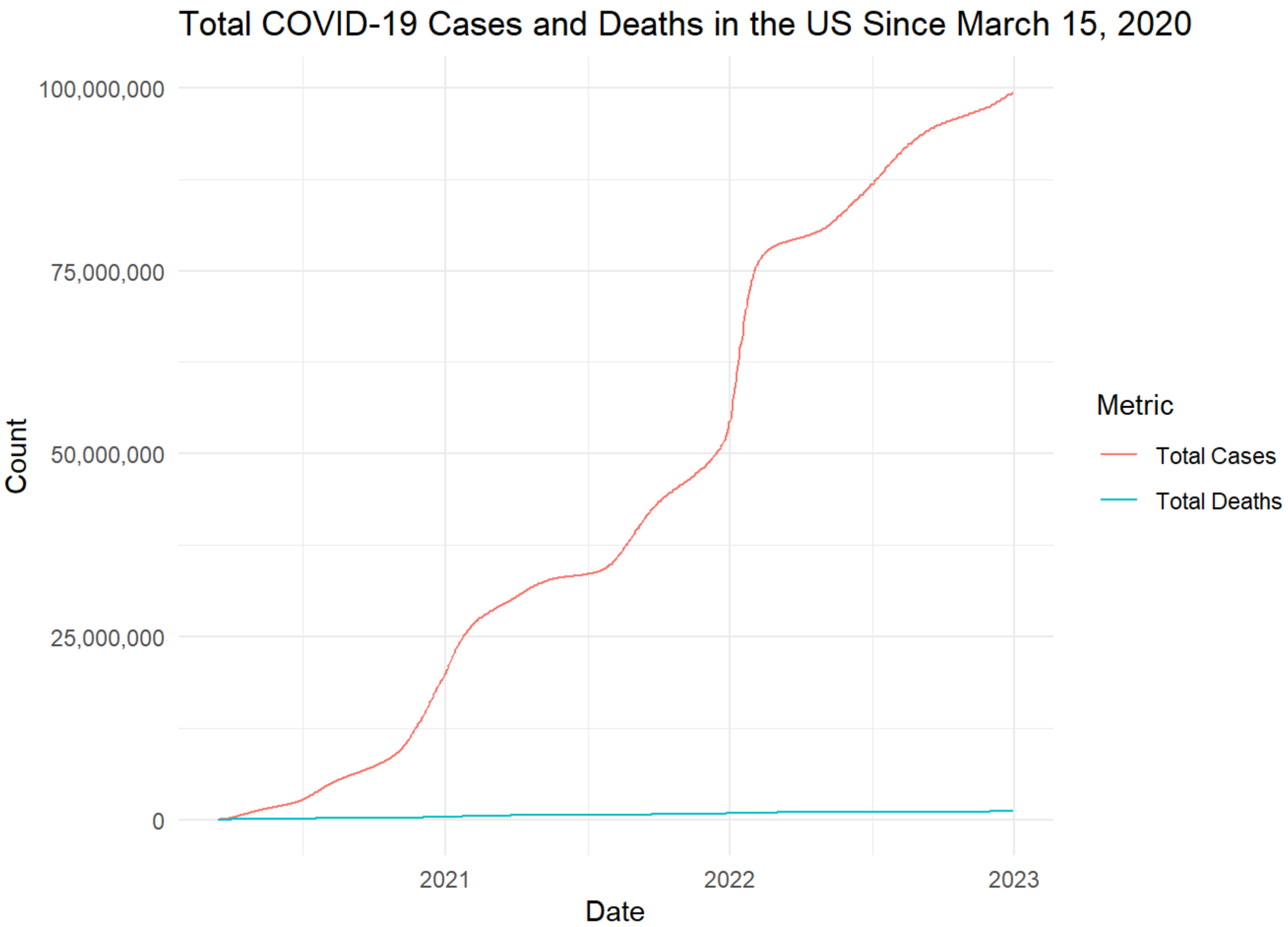
Calculate Total Cases and Deaths:

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.

```r
# 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 cases 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 cases per 100,000 people.

# Hint: To calculate per 100,000 people, first tidy the population estimates data and calculate 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) %>%
```

```r
  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.

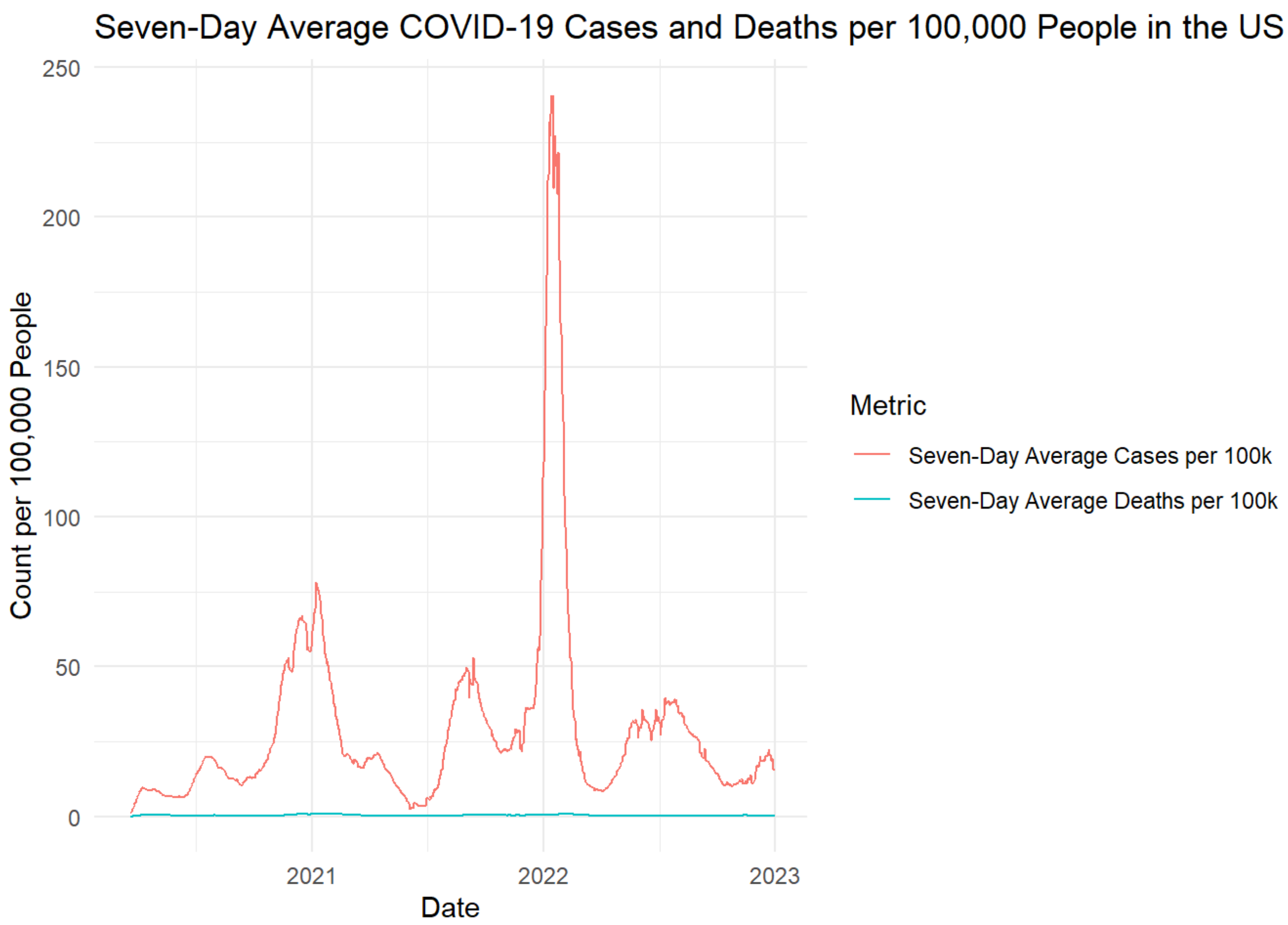
Results and Interpretation

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)
```



– 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.

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.

```
# Determine the top 10 states in terms of total deaths and cases between March 15, 2020, a
nd December 31, 2021. To do this, transform your combined COVID-19 data to summarize total
deaths and cases by state up to December 31, 2021.

# Filter the data for dates between March 15, 2020, and December 31, 2021
us_counties_filtered <- us_counties_combined %>%
  filter(date >= "2020-03-15" & date <= "2021-12-31")

# Summarize the total deaths and cases by state
state_totals <- us_counties_filtered %>%
  group_by(state) %>%
  summarise(
    total_deaths = sum(deaths, na.rm = TRUE),
    total_cases = sum(cases, na.rm = TRUE)
  ) %>%
  arrange(desc(total_deaths), desc(total_cases))

# Display the top 10 states by total deaths and cases
top_10_deaths <- state_totals %>%
  arrange(desc(total_deaths)) %>%
  slice(1:10)

top_10_cases <- state_totals %>%
  arrange(desc(total_cases)) %>%
  slice(1:10)

# Output the results
print(top_10_deaths)
```

```
## # A tibble: 10 × 3
##    state         total_deaths total_cases
##    <chr>                <dbl>       <dbl>
##  1 New York          27239066   902069748
##  2 California        25597513  1671429376
##  3 Texas             23016708  1355197939
##  4 Florida           17965464  1112292949
##  5 New Jersey        13223576   428165855
##  6 Pennsylvania      12028063   504448072
##  7 Illinois          11517916   610074612
##  8 Michigan           9297780   408728096
##  9 Georgia            9155719   509622188
## 10 Massachusetts      8651530   301052122
```

```
print(top_10_cases)
```

```
## # A tibble: 10 × 3
##    state          total_deaths total_cases
##    <chr>                 <dbl>       <dbl>
##  1 California         25597513  1671429376
##  2 Texas              23016708  1355197939
##  3 Florida            17965464  1112292949
##  4 New York           27239066   902069748
##  5 Illinois           11517916   610074612
##  6 Georgia             9155719   509622188
##  7 Pennsylvania       12028063   504448072
##  8 Ohio                8389799   487380527
##  9 North Carolina      5816149   451987735
## 10 New Jersey         13223576   428165855
```

```
# 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
```

– Communicate your methodology, results, and interpretation here –

Data Preparation, Summarization and Sorting and Filtering

These lists provide insights into the states most affected by COVID-19 in terms of both deaths and cases. This information can be used to understand regional impacts and inform public health strategies.

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?

```r
# Determine the top 10 states in terms of deaths and cases per 100,000 people between Marc
h 15, 2020, and December 31, 2021. You should first tidy and transform the population esti
mates to include population totals by state. Use your relational data verbs (e.g. full_joi
n()) to join the population estimates with the cases and death statistics using the state
name as a key. Then, use case_when() and grepl() to add a population column to your table
that only includes the estimated population for the associated year. Finally, mutate your
table to calculate deaths and cases per 100,000 people and summarize by state.

# 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 between March 15, 2020, and December 31, 2021
us_counties_filtered <- us_counties_combined %>%
  filter(date >= "2020-03-15" & date <= "2021-12-31")

# Summarize the total deaths and cases by state
state_totals <- us_counties_filtered %>%
  group_by(state) %>%
  summarise(
    total_deaths = sum(deaths, na.rm = TRUE),
    total_cases = sum(cases, na.rm = TRUE)
  )

# Summarize population by state
state_population <- us_population_estimates %>%
  group_by(STNAME) %>%
  summarise(total_population = sum(Estimate, na.rm = TRUE))

# Join the state_totals with state_population
state_totals <- state_totals %>%
  left_join(state_population, by = c("state" = "STNAME"))

# Calculate deaths and cases per 100,000 people
state_totals <- state_totals %>%
  mutate(
    deaths_per_100k = (total_deaths / total_population) * 100000,
    cases_per_100k = (total_cases / total_population) * 100000
  )

# Determine the top 10 states by deaths per 100,000 people
top_10_deaths_per_100k <- state_totals %>%
  arrange(desc(deaths_per_100k)) %>%
  slice(1:10)

# Determine the top 10 states by cases per 100,000 people
top_10_cases_per_100k <- state_totals %>%
  arrange(desc(cases_per_100k)) %>%
  slice(1:10)

# Output the results
```

```
print(top_10_deaths_per_100k)
```

```
## # A tibble: 10 × 6
##    state          total_deaths total_cases total_population deaths_per_100k
##    <chr>                 <dbl>       <dbl>            <dbl>           <dbl>
##  1 New Jersey        13223576   428165855         18546873          71298.
##  2 New York          27239066   902069748         39990846          68113.
##  3 Massachusetts      8651530   301052122         14006943          61766.
##  4 Mississippi        3476862   157394304          5906835          58862.
##  5 Louisiana          5401191   240915268          9275250          58232.
##  6 Connecticut        4096430   144118119          7205857          56849.
##  7 Rhode Island       1236144    63727011          2191839          56398.
##  8 Arizona            7639621   397628355         14454302          52854.
##  9 Alabama            4856906   260019795         10064680          48257.
## 10 South Dakota        830670    55113212          1782475          46602.
## # ℹ 1 more variable: cases_per_100k <dbl>
```

```
print(top_10_cases_per_100k)
```

```
## # A tibble: 10 × 6
##    state            total_deaths total_cases total_population deaths_per_100k
##    <chr>                   <dbl>       <dbl>            <dbl>           <dbl>
##  1 North Dakota           673677    50379884          1553910          43354.
##  2 South Dakota           830670    55113212          1782475          46602.
##  3 Rhode Island          1236144    63727011          2191839          56398.
##  4 Tennessee             5331701   392376492         13895337          38370.
##  5 Utah                  1022793   182647550          6619659          15451.
##  6 Arizona               7639621   397628355         14454302          52854.
##  7 Arkansas              2653528   163193858          6038123          43946.
##  8 Mississippi           3476862   157394304          5906835          58862.
##  9 Iowa                  2573396   169479989          6381748          40324.
## 10 South Carolina        4421226   271565792         10321434          42835.
## # ℹ 1 more variable: cases_per_100k <dbl>
```

```
# 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
```

– Communicate your methodology, results, and interpretation here –

Data Preparation -> Summarization -> Population Data -> Normalization -> Sorting and Filtering

This analysis provides insights into the states most affected by COVID-19 in terms of deaths and cases per 100,000 people. Normalizing the data by population size allows for more accurate comparisons across states, highlighting the regions with the highest relative impact. This information is crucial for understanding the spread and impact of COVID-19 and informing public health strategies.

Question 3

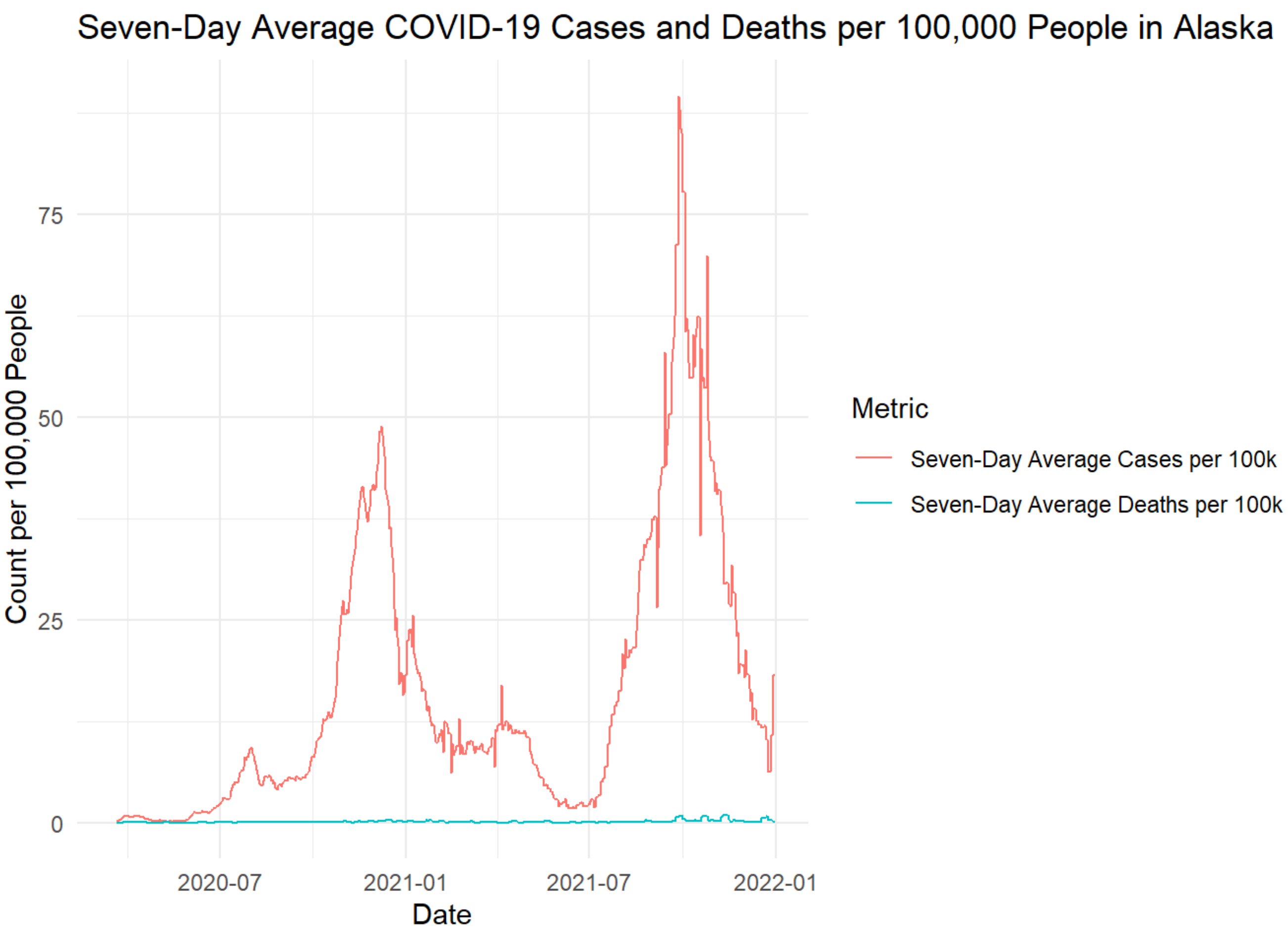
Now, select a state and calculate the seven-day averages for new cases and deaths per 100,000 people. Once you have calculated the averages, create a visualization using ggplot2 to represent the data.

```
# Select a state and then filter by state and date range your data from Question 1. Calcul
ate the seven-day average following the same procedure as Part 1.

# 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 between March 15, 2020, and December 31, 2021
us_counties_filtered <- us_counties_combined %>%
  filter(date >= "2020-03-15" & date <= "2021-12-31")

# Summarize the total deaths and cases by state
state_totals <- us_counties_filtered %>%
  group_by(state) %>%
  summarise(
    total_deaths = sum(deaths, na.rm = TRUE),
    total_cases = sum(cases, na.rm = TRUE)
  )

# Summarize population by state
state_population <- us_population_estimates %>%
  group_by(STNAME) %>%
  summarise(total_population = sum(Estimate, na.rm = TRUE))

# Join the state_totals with state_population
state_totals <- state_totals %>%
  left_join(state_population, by = c("state" = "STNAME"))

# Select Alaska
alaska_data <- us_counties_filtered %>%
  filter(state == "Alaska") %>%
  group_by(date) %>%
  summarise(
    total_deaths = sum(deaths, na.rm = TRUE),
    total_cases = sum(cases, na.rm = TRUE)
  )

# Get population for Alaska
alaska_population <- state_population %>%
  filter(STNAME == "Alaska") %>%
  pull(total_population)

# Calculate new cases and deaths each day and their 7-day averages
alaska_data <- alaska_data %>%
  mutate(
    new_deaths = total_deaths - lag(total_deaths, default = 0),
    new_cases = total_cases - lag(total_cases, default = 0),
    deaths_per_100k = (new_deaths / alaska_population) * 100000,
    cases_per_100k = (new_cases / alaska_population) * 100000,
    deaths_7_day = rollmean(deaths_per_100k, 7, fill = NA, align = "right"),
    cases_7_day = rollmean(cases_per_100k, 7, fill = NA, align = "right")
```

```
  )

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

```
## # A tibble: 657 × 9
##    date       total_deaths total_cases new_deaths new_cases deaths_per_100k
##    <date>            <dbl>       <dbl>      <dbl>     <dbl>           <dbl>
##  1 2020-03-15            0           1          0         1               0
##  2 2020-03-16            0           3          0         2               0
##  3 2020-03-17            0           6          0         3               0
##  4 2020-03-18            0           9          0         3               0
##  5 2020-03-19            0          12          0         3               0
##  6 2020-03-20            0          14          0         2               0
##  7 2020-03-21            0          21          0         7               0
##  8 2020-03-22            0          22          0         1               0
##  9 2020-03-23            0          36          0        14               0
## 10 2020-03-24            0          42          0         6               0
## # ℹ 647 more rows
## # ℹ 3 more variables: cases_per_100k <dbl>, deaths_7_day <dbl>,
## #   cases_7_day <dbl>
```

```
# Create the visualization
ggplot(alaska_data, aes(x = date)) +
  geom_line(aes(y = cases_7_day, color = "Seven-Day Average Cases per 100k")) +
  geom_line(aes(y = deaths_7_day, color = "Seven-Day Average Deaths per 100k")) +
  labs(
    title = "Seven-Day Average COVID-19 Cases and Deaths per 100,000 People in Alaska",
    x = "Date",
    y = "Count per 100,000 People",
    color = "Metric"
  ) +
  theme_minimal() +
  scale_y_continuous(labels = scales::comma)
```


Seven-Day Average COVID-19 Cases and Deaths per 100,000 People in Alaska
Count per 100,000 People
75
50
25
0
2020-07
2021-01
2021-07
2022-01
Date
Metric
Seven-Day Average Cases per 100k
Seven-Day Average Deaths per 100k

```
# 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_10
0k deaths_7_day  cases_7_day
#     <chr>    <date>            <dbl>    <dbl>      <dbl>         <dbl>          <dbl>
<dbl>         <dbl>
#  1 Colorado 2020-03-15            2      136      5784308        0.0346          2.35
NA            NA
#  2 Colorado 2020-03-16            2      161      5784308        0.0346          2.78
NA            NA
#  3 Colorado 2020-03-17            3      183      5784308        0.0519          3.16
NA            NA
#  4 Colorado 2020-03-18            3      216      5784308        0.0519          3.73
NA            NA
#  5 Colorado 2020-03-19            5      278      5784308        0.0864          4.81
NA            NA
#  6 Colorado 2020-03-20            5      364      5784308        0.0864          6.29
NA            NA
#  7 Colorado 2020-03-21            6      475      5784308        0.104           8.21
NA            NA
#  8 Colorado 2020-03-22            7      591      5784308        0.121           10.2
0.0123         1.12
#  9 Colorado 2020-03-23           10      721      5784308        0.173           12.5
0.0198         1.38
# 10 Colorado 2020-03-24           11      912      5784308        0.190           15.8
0.0198         1.80
# … with 646 more rows
```

– Communicate your methodology, results, and interpretation here –

Data Preparation -> Population Data -> Select State (Alaska) -> Normalization -> Visualization

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

By looking at the trends in this visualization, health officials can better understand the spread and impact of COVID-19 in Alaska. 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.

Question 4

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

```r
# Using the same state as Question 2, filter your state and date range from the combined d
ata set from Part 1 and summarize cases and deaths. Produce two lists arranged by deaths a
nd cases. When transforming the data, be sure to include the "fips" column as you will nee
d this to complete Question 5.


# Filter the data for Alaska and dates between March 15, 2020, and December 31, 2021
alaska_data <- us_counties_combined %>%
  filter(state == "Alaska" & date >= "2020-03-15" & date <= "2021-12-31")

# Summarize the total deaths and cases by county
county_totals <- alaska_data %>%
  group_by(county, fips) %>%
  summarise(
    total_deaths = sum(deaths, na.rm = TRUE),
    total_cases = sum(cases, na.rm = TRUE)
  )
```

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

```r
# Convert fips to character in both datasets
county_totals <- county_totals %>%
  mutate(fips = as.character(fips))

us_population_estimates <- us_population_estimates %>%
  mutate(fips = as.character(fips))

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

# Summarize population by county (using fips)
county_population <- us_population_estimates %>%
  filter(STNAME == "Alaska") %>%
  group_by(fips) %>%
  summarise(total_population = sum(Estimate, na.rm = TRUE))

# Join the county_totals with county_population
county_totals <- county_totals %>%
  left_join(county_population, by = "fips")

# Calculate deaths and cases per 100,000 people
county_totals <- county_totals %>%
  mutate(
    deaths_per_100k = (total_deaths / total_population) * 100000,
    cases_per_100k = (total_cases / total_population) * 100000
  )

# Determine the top 5 counties by deaths per 100,000 people
top_5_deaths_per_100k <- county_totals %>%
  arrange(desc(deaths_per_100k)) %>%
  slice(1:5)

# Determine the top 5 counties by cases per 100,000 people
top_5_cases_per_100k <- county_totals %>%
  arrange(desc(cases_per_100k)) %>%
  slice(1:5)

# Output the results
print(top_5_deaths_per_100k)
```

```
## # A tibble: 28 × 7
## # Groups:   county [28]
##    county        fips  total_deaths total_cases total_population deaths_per_100k
##    <chr>         <chr>        <dbl>       <dbl>            <dbl>           <dbl>
##  1 Aleutians Ea… 02013          870      134048               NA              NA
##  2 Aleutians We… 02016           54      279979               NA              NA
##  3 Anchorage     02020        88208    15181403               NA              NA
##  4 Bethel Censu… 02050         8342     1731124               NA              NA
##  5 Bristol Bay … 02997           38      119966               NA              NA
##  6 Denali Borou… 02068           99       58464               NA              NA
##  7 Dillingham C… 02070         1208      158283               NA              NA
##  8 Fairbanks No… 02090        21404     3871383               NA              NA
##  9 Haines Borou… 02100           65       35867               NA              NA
## 10 Juneau City … 02110         2127      867358               NA              NA
## # ℹ 18 more rows
## # ℹ 1 more variable: cases_per_100k <dbl>
```

```
print(top_5_cases_per_100k)
```

```
## # A tibble: 28 × 7
## # Groups:   county [28]
##    county        fips  total_deaths total_cases total_population deaths_per_100k
##    <chr>         <chr>        <dbl>       <dbl>            <dbl>           <dbl>
##  1 Aleutians Ea… 02013          870      134048               NA              NA
##  2 Aleutians We… 02016           54      279979               NA              NA
##  3 Anchorage     02020        88208    15181403               NA              NA
##  4 Bethel Censu… 02050         8342     1731124               NA              NA
##  5 Bristol Bay … 02997           38      119966               NA              NA
##  6 Denali Borou… 02068           99       58464               NA              NA
##  7 Dillingham C… 02070         1208      158283               NA              NA
##  8 Fairbanks No… 02090        21404     3871383               NA              NA
##  9 Haines Borou… 02100           65       35867               NA              NA
## 10 Juneau City … 02110         2127      867358               NA              NA
## # ℹ 18 more rows
## # ℹ 1 more variable: cases_per_100k <dbl>
```

```
# 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
```

– Communicate your methodology, results, and interpretation here –

Data Preparation -> Summarization -> Population Data -> Normalization -> Sorting and Filtering

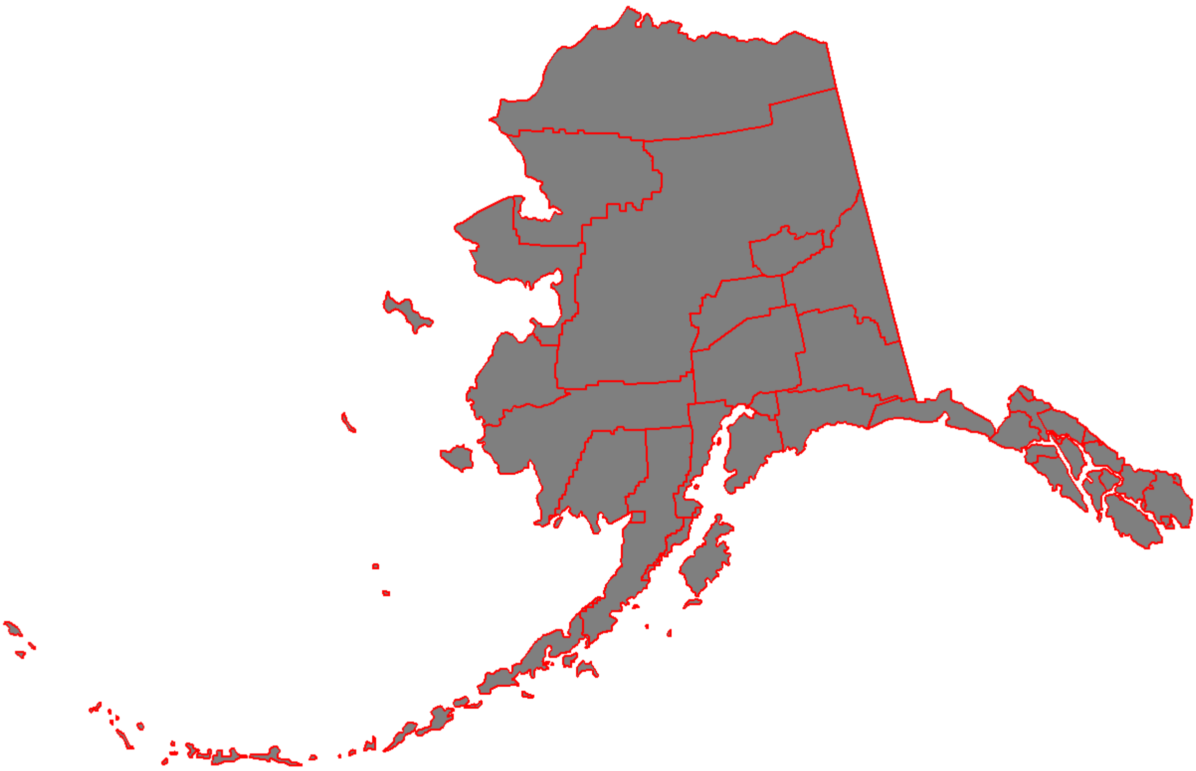
This analysis highlights the counties in Alaska that have been most affected by COVID-19 in terms of deaths and cases per 100,000 people. This information can be used to target public health interventions and resources to the areas that need them most.

Question 5

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

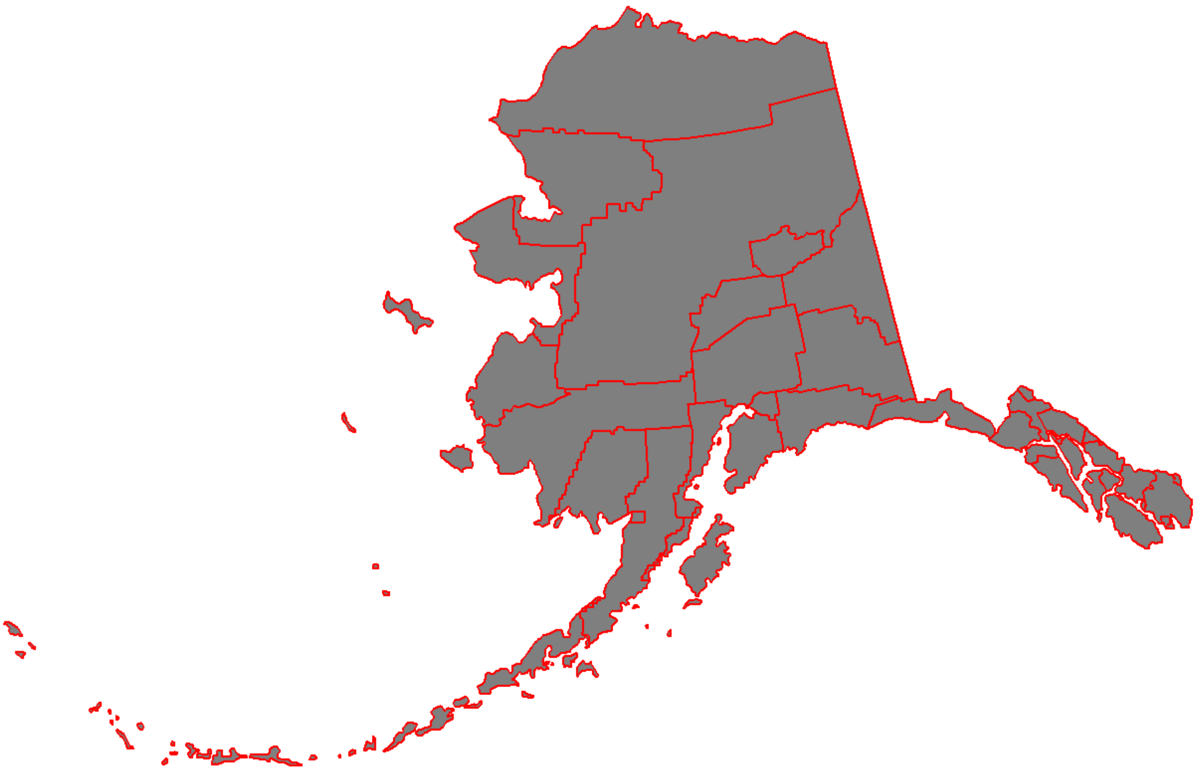
```
# Map visualization for deaths per 100,000 people
plot_usmap(regions = "county", include = "AK", data = top_5_deaths_per_100k, values = "deaths_per_100k", color = "red") +
  scale_fill_continuous(low = "white", high = "red", name = "Deaths per 100,000") +
  labs(title = "COVID-19 Deaths per 100,000 People in Alaska Counties") +
  theme(legend.position = "right")
```

COVID-19 Deaths per 100,000 People in Alaska Counties



```r
# Map visualization for deaths per 100,000 people
plot_usmap(regions = "county", include = "AK", data = top_5_cases_per_100k, values = "case
s_per_100k", color = "red") +
  scale_fill_continuous(low = "white", high = "red", name = "Cases per 100,000") +
  labs(title = "COVID-19 Cases per 100,000 People in Alaska Counties") +
  theme(legend.position = "right")
```

COVID-19 Cases per 100,000 People in Alaska Counties



```
# Copy and modify the code below for your state.
#
# plot_usmap arguments:
#   regions: can be one of ("states", "state", "counties", "county"). The default is "stat
es"
#   include: The regions to include in the resulting map. If regions is "states"/"state",
the value can be either a state name, abbreviation or FIPS code. For counties, the FIPS mu
st be provided as there can be multiple counties with the same name.
#   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_deat
hs", color = "blue") +  scale_fill_continuous(low = "white", high = "blue", name = "Deaths
per 100,000")
```

– Communicate your methodology, results, and interpretation here –

Same as before expect added visualization for clarity.

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.

```
# 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 between March 15, 2020, and December 31, 2021
us_counties_filtered <- us_counties_combined %>%
  filter(date >= "2020-03-15" & date <= "2021-12-31")

# List of states to analyze
states <- c("Alabama", "Arizona", "Arkansas")

# Summarize the total deaths and cases by state
state_totals <- us_counties_filtered %>%
  filter(state %in% states) %>%
  group_by(state, date) %>%
  summarise(
    total_deaths = sum(deaths, na.rm = TRUE),
    total_cases = sum(cases, na.rm = TRUE)
  ) %>%
  ungroup()
```

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

```r
# Summarize population by state
state_population <- us_population_estimates %>%
  filter(STNAME %in% states) %>%
  group_by(STNAME) %>%
  summarise(total_population = sum(Estimate, na.rm = TRUE)) %>%
  rename(state = STNAME)

# Join the state_totals with state_population
state_totals <- state_totals %>%
  left_join(state_population, by = "state")

# Calculate new cases and deaths each day and their 7-day averages per 100,000 people
state_totals <- state_totals %>%
  group_by(state) %>%
  mutate(
    new_deaths = total_deaths - lag(total_deaths, default = 0),
    new_cases = total_cases - lag(total_cases, default = 0),
    deaths_per_100k = (new_deaths / total_population) * 100000,
    cases_per_100k = (new_cases / total_population) * 100000,
    deaths_7_day = rollmean(deaths_per_100k, 7, fill = NA, align = "right"),
    cases_7_day = rollmean(cases_per_100k, 7, fill = NA, align = "right")
  ) %>%
  ungroup()

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

```
## # A tibble: 1,971 × 11
##    state   date       total_deaths total_cases total_population new_deaths
##    <chr>   <date>            <dbl>       <dbl>            <dbl>      <dbl>
##  1 Alabama 2020-03-15            0          23         10064680          0
##  2 Alabama 2020-03-16            0          29         10064680          0
##  3 Alabama 2020-03-17            0          39         10064680          0
##  4 Alabama 2020-03-18            0          51         10064680          0
##  5 Alabama 2020-03-19            0          78         10064680          0
##  6 Alabama 2020-03-20            0         106         10064680          0
##  7 Alabama 2020-03-21            0         131         10064680          0
##  8 Alabama 2020-03-22            0         157         10064680          0
##  9 Alabama 2020-03-23            0         196         10064680          0
## 10 Alabama 2020-03-24            0         242         10064680          0
## # ℹ 1,961 more rows
## # ℹ 5 more variables: new_cases <dbl>, deaths_per_100k <dbl>,
## #   cases_per_100k <dbl>, deaths_7_day <dbl>, cases_7_day <dbl>
```

– Communicate your methodology, results, and interpretation here –

Data Preparation -> Population Data -> Normalization

The resulting data frame state_totals contains the seven-day averages for new cases and deaths per 100,000 people for Alabama, Arizona, and Arkansas. This data provides a clear view of how the COVID-19 situation evolved in each state, adjusted for population size.

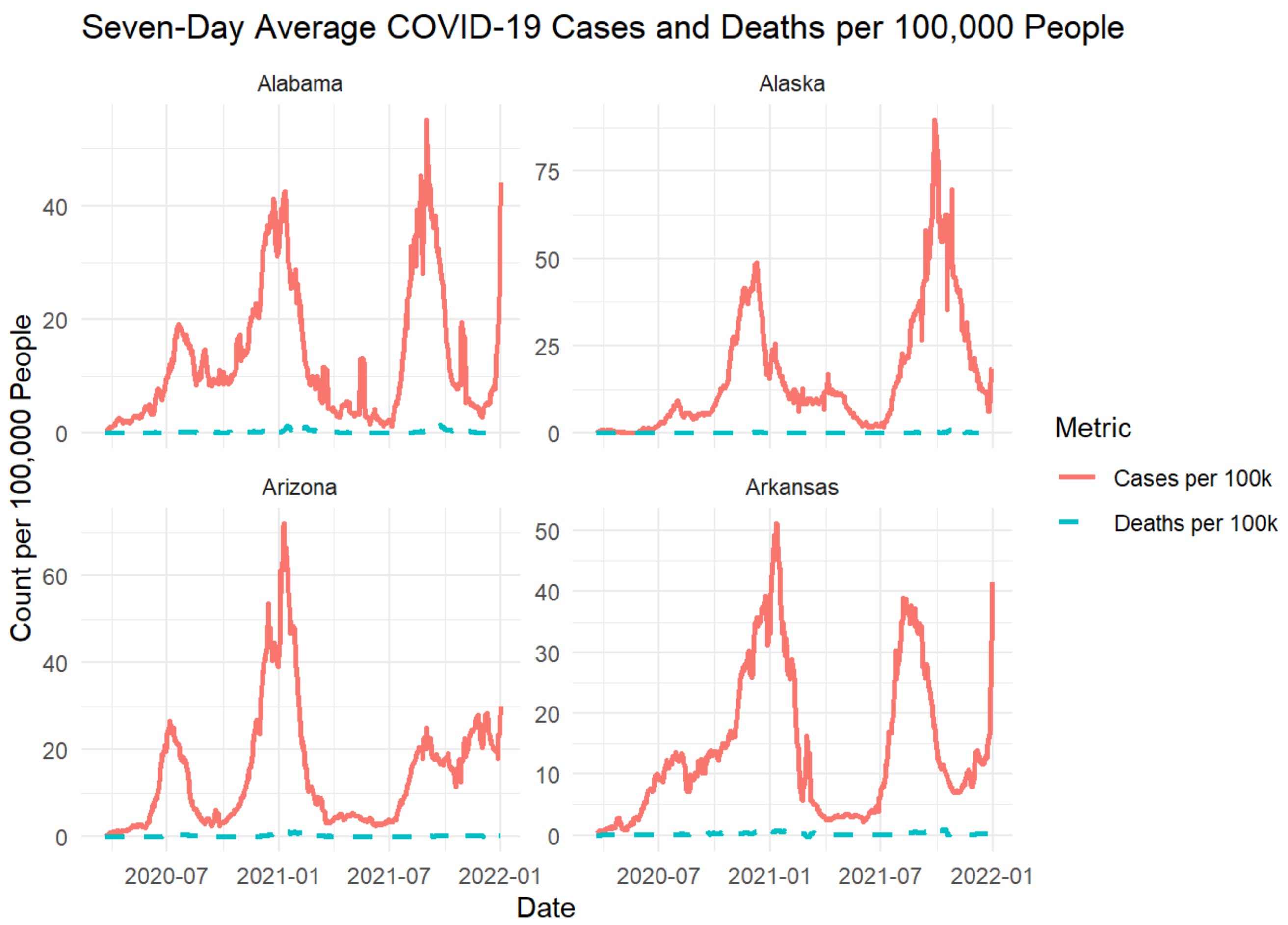
Question 7

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

```
# 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 between March 15, 2020, and December 31, 2021
us_counties_filtered <- us_counties_combined %>%
  filter(date >= "2020-03-15" & date <= "2021-12-31")

# List of states to analyze
states <- c("Alabama", "Arizona", "Arkansas", "Alaska")

# Summarize the total deaths and cases by state
state_totals <- us_counties_filtered %>%
  filter(state %in% states) %>%
  group_by(state, date) %>%
  summarise(
    total_deaths = sum(deaths, na.rm = TRUE),
    total_cases = sum(cases, na.rm = TRUE)
  ) %>%
  ungroup()
```

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

```
# Summarize population by state
state_population <- us_population_estimates %>%
  filter(STNAME %in% states) %>%
  group_by(STNAME) %>%
  summarise(total_population = sum(Estimate, na.rm = TRUE)) %>%
  rename(state = STNAME)

# Join the state_totals with state_population
state_totals <- state_totals %>%
  left_join(state_population, by = "state")

# Calculate new cases and deaths each day and their 7-day averages per 100,000 people
state_totals <- state_totals %>%
  group_by(state) %>%
  mutate(
    new_deaths = total_deaths - lag(total_deaths, default = 0),
    new_cases = total_cases - lag(total_cases, default = 0),
    deaths_per_100k = (new_deaths / total_population) * 100000,
    cases_per_100k = (new_cases / total_population) * 100000,
    deaths_7_day = rollmean(deaths_per_100k, 7, fill = NA, align = "right"),
    cases_7_day = rollmean(cases_per_100k, 7, fill = NA, align = "right")
  ) %>%
  ungroup()

# Visualization
ggplot(state_totals, aes(x = date)) +
  geom_line(aes(y = cases_7_day, color = "Cases per 100k"), size = 1) +
  geom_line(aes(y = deaths_7_day, color = "Deaths per 100k"), size = 1, linetype = "dashe
d") +
  facet_wrap(~ state, scales = "free_y") +
  labs(
    title = "Seven-Day Average COVID-19 Cases and Deaths per 100,000 People",
    x = "Date",
    y = "Count per 100,000 People",
    color = "Metric"
  ) +
  theme_minimal() +
  scale_y_continuous(labels = scales::comma)
```



– Communicate your methodology, results, and interpretation here –

Data Preparation -> Visualization

The visualization shows the seven-day average of new COVID-19 cases and deaths per 100,000 people for Alabama, Alaska, Arizona, and Arkansas. The solid lines represent the cases per 100,000 people, and the dashed lines represent the deaths per 100,000 people.

By comparing these trends, we can see how the pandemic affected each state over time. This information is crucial for understanding regional differences in the spread and impact of COVID-19 and can inform public health strategies and resource allocation.

Part 3 - Global Comparison

```
# Import global COVID-19 statistics aggregated by the Center for Systems Science and Engin
eering (CSSE) at Johns Hopkins University.
# Import global population estimates from the World Bank.

csse_global_deaths <- read_csv("time_series_covid19_deaths_global.csv")
```

```
## Rows: 289 Columns: 1147
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr    (2): Province/State, Country/Region
## dbl (1145): Lat, Long, 1/22/20, 1/23/20, 1/24/20, 1/25/20, 1/26/20, 1/27/20,...
##
## 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.
```

```
csse_global_cases <- read_csv("time_series_covid19_confirmed_global.csv")
```

```
## Rows: 289 Columns: 1147
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr    (2): Province/State, Country/Region
## dbl (1145): Lat, Long, 1/22/20, 1/23/20, 1/24/20, 1/25/20, 1/26/20, 1/27/20,...
##
## 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.
```

```
csse_us_deaths <- read_csv("time_series_covid19_deaths_US.csv")
```

```
## Rows: 3342 Columns: 1155
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr    (6): iso2, iso3, Admin2, Province_State, Country_Region, Combined_Key
## dbl (1149): UID, code3, FIPS, Lat, Long_, Population, 1/22/20, 1/23/20, 1/24...
##
## 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.
```

```
csse_us_cases <- read_csv("time_series_covid19_confirmed_US.csv")
```

```
## Rows: 3342 Columns: 1154
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr    (6): iso2, iso3, Admin2, Province_State, Country_Region, Combined_Key
## dbl (1148): UID, code3, FIPS, Lat, Long_, 1/22/20, 1/23/20, 1/24/20, 1/25/20...
##
## 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.
```

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

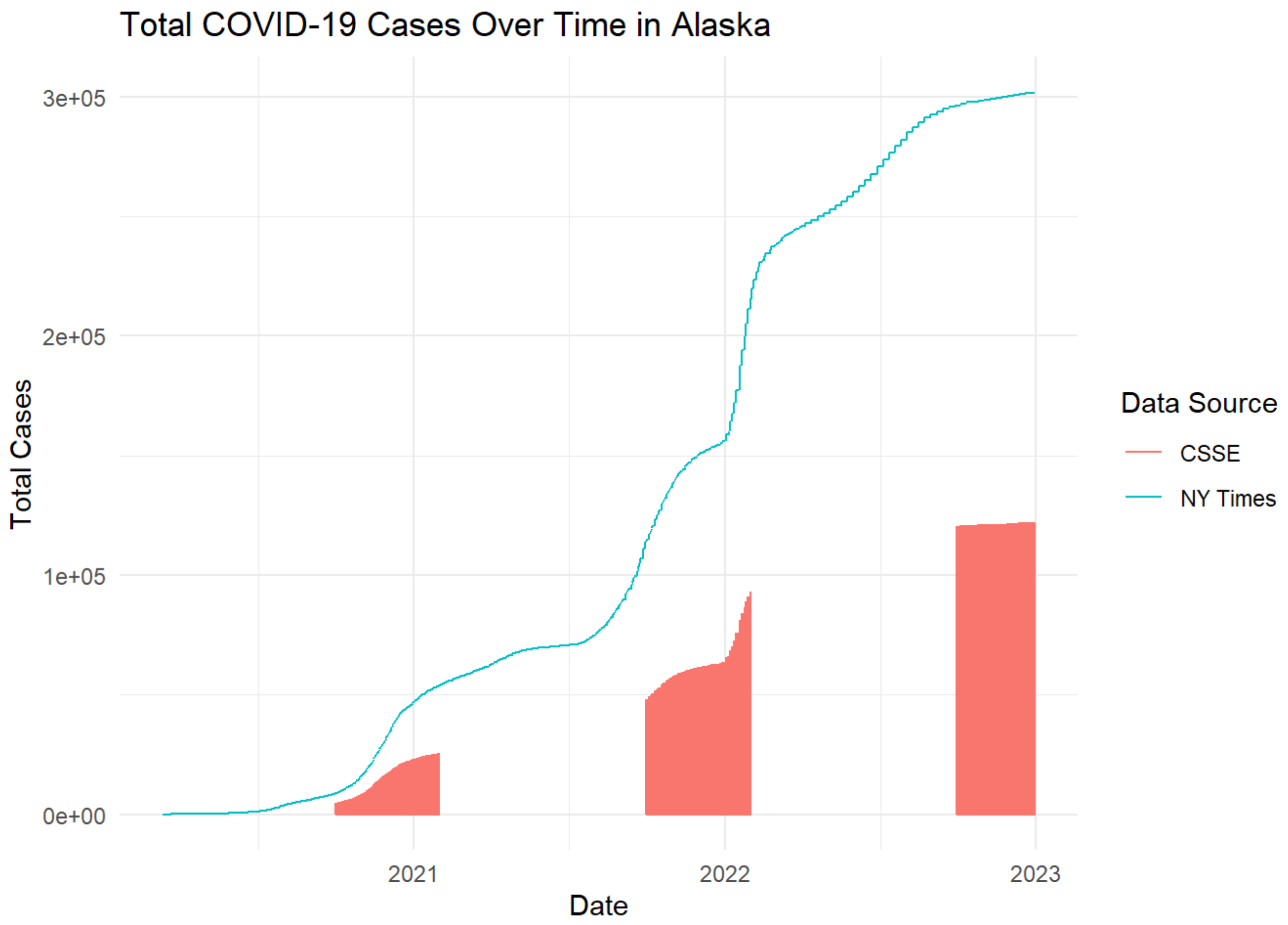
```
## Rows: 267 Columns: 6
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (6): Country Name, Country Code, Series Name, Series Code, 2020 [YR2020]...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Question 1

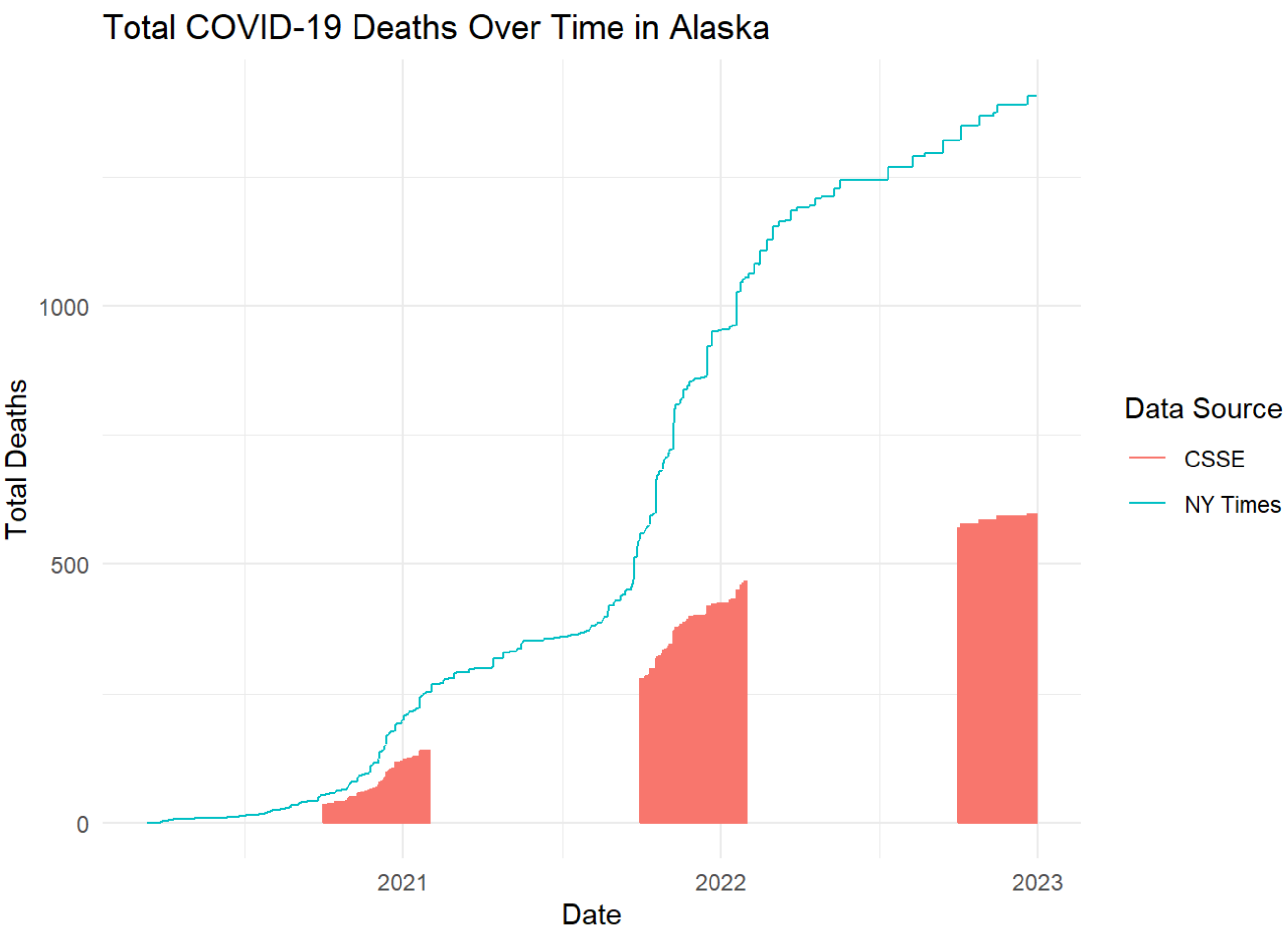
Using the state you selected in Part 2 Question 2 compare the daily number of cases and deaths reported from the CSSE and NY Times.

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

# Filter CSSE data for Alaska and pivot longer to tidy format
csse_cases_alaska <- csse_us_cases %>%
  filter(Province_State == "Alaska") %>%
  select(-c(UID, iso2, iso3, code3, FIPS, Admin2, Country_Region, Lat, Long_, Combined_Ke
y)) %>%
  pivot_longer(cols = starts_with("1"), names_to = "date", values_to = "total_cases") %>%
  mutate(date = mdy(date))

csse_deaths_alaska <- csse_us_deaths %>%
  filter(Province_State == "Alaska") %>%
  select(-c(UID, iso2, iso3, code3, FIPS, Admin2, Country_Region, Lat, Long_, Combined_Ke
y)) %>%
  pivot_longer(cols = starts_with("1"), names_to = "date", values_to = "total_deaths") %>%
  mutate(date = mdy(date))

# Join the CSSE cases and deaths data
csse_alaska <- csse_cases_alaska %>%
  left_join(csse_deaths_alaska, by = c("Province_State", "date"))



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

# Read in the NY Times data
#us_counties_2020 <- read_csv("us-counties-2020.csv")
#us_counties_2021 <- read_csv("us-counties-2021.csv")
#us_counties_2022 <- read_csv("us-counties-2022.csv")

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

# Filter the NY Times data for Alaska
nytimes_alaska <- us_counties_combined %>%
  filter(state == "Alaska") %>%
  group_by(date) %>%
  summarise(
    total_cases = sum(cases, na.rm = TRUE),
    total_deaths = sum(deaths, na.rm = TRUE)
  )

# Join NY Times and CSSE data
comparison_data <- nytimes_alaska %>%
  rename(nytimes_total_cases = total_cases, nytimes_total_deaths = total_deaths) %>%
  left_join(csse_alaska, by = "date") %>%
  rename(csse_total_cases = total_cases, csse_total_deaths = total_deaths)


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

```
# Visualization for cases
ggplot(comparison_data, aes(x = date)) +
  geom_line(aes(y = nytimes_total_cases, color = "NY Times")) +
  geom_line(aes(y = csse_total_cases, color = "CSSE")) +
  labs(
    title = "Total COVID-19 Cases Over Time in Alaska",
    x = "Date",
    y = "Total Cases",
    color = "Data Source"
  ) +
  theme_minimal()
```




```
# Visualization for deaths
ggplot(comparison_data, aes(x = date)) +
  geom_line(aes(y = nytimes_total_deaths, color = "NY Times")) +
  geom_line(aes(y = csse_total_deaths, color = "CSSE")) +
  labs(
    title = "Total COVID-19 Deaths Over Time in Alaska",
    x = "Date",
    y = "Total Deaths",
    color = "Data Source"
  ) +
  theme_minimal()
```




```
# Your tidied CSSE data for your selected state should look similar to the following tibbl
e:
#
# 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 –

Import and Tidy CSSE Data -> Join CSSE Cases and Deaths -> Tidy NY Times Data -> Join NY Times and CSSE Data -> Visualization

The visualizations show the total COVID-19 cases and deaths over time in Alaska, comparing data reported by NY Times and CSSE. By plotting the data from both sources, we can verify their consistency and investigate any discrepancies.

This analysis ensures the reliability of the data sources and helps to understand the impact of COVID-19 in Alaska using multiple data repositories.

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.

# Transform global cases data
global_cases <- csse_global_cases %>%
  pivot_longer(cols = starts_with("1"), names_to = "date", values_to = "total_cases") %>%
  mutate(date = mdy(date)) %>%
  select(`Country/Region`, date, total_cases)

# Transform global deaths data
global_deaths <- csse_global_deaths %>%
  pivot_longer(cols = starts_with("1"), names_to = "date", values_to = "total_deaths") %>%
  mutate(date = mdy(date)) %>%
  select(`Country/Region`, date, total_deaths)

# Transform US cases data
us_cases <- csse_us_cases %>%
  pivot_longer(cols = starts_with("1"), names_to = "date", values_to = "total_cases") %>%
  mutate(date = mdy(date)) %>%
  select(Province_State, `Country_Region` = `Country_Region`, date, total_cases)

# Transform US deaths data
us_deaths <- csse_us_deaths %>%
  pivot_longer(cols = starts_with("1"), names_to = "date", values_to = "total_deaths") %>%
  mutate(date = mdy(date)) %>%
  select(Province_State, `Country_Region` = `Country_Region`, date, total_deaths)

# Combine global and US cases
all_cases <- bind_rows(global_cases, us_cases)

# Combine global and US deaths
all_deaths <- bind_rows(global_deaths, us_deaths)

# Then, tidy the global population estimates. While tidying your data, remember to include columns that you will be able to use when joining the COVID-19 data.

# Tidy the population data
tidy_population <- global_population_estimates %>%
  rename(`Country/Region` = `Country Name`, population = `2021 [YR2021]`) %>%
  select(`Country/Region`, population) %>%
  mutate(population = as.numeric(population))




# You will notice that the population estimates data does not include every country reported in the CSSE data. When calculating statistics per 100,000 people, you will need to filter the CSSE data to only include countries that you have population estimates for.

# Summarize cases and deaths by country
summary_cases <- all_cases %>%
  filter(date >= "2020-03-15" & date <= "2021-12-31") %>%
  group_by(`Country/Region`) %>%
  summarise(total_cases = sum(total_cases, na.rm = TRUE))
```

```
summary_deaths <- all_deaths %>%
  filter(date >= "2020-03-15" & date <= "2021-12-31") %>%
  group_by(`Country/Region`) %>%
  summarise(total_deaths = sum(total_deaths, na.rm = TRUE))

# Join with population estimates
cases_with_population <- summary_cases %>%
  inner_join(tidy_population, by = "Country/Region") %>%
  mutate(cases_per_100k = (total_cases / population) * 100000)

deaths_with_population <- summary_deaths %>%
  inner_join(tidy_population, by = "Country/Region") %>%
  mutate(deaths_per_100k = (total_deaths / population) * 100000)

# Filter to include only countries with population estimates
valid_cases <- cases_with_population %>%
  filter(!is.na(population))

valid_deaths <- deaths_with_population %>%
  filter(!is.na(population))

# Top 10 countries by cases per 100,000 people
top_10_cases_per_100k <- valid_cases %>%
  arrange(desc(cases_per_100k)) %>%
  slice(1:10)

# Top 10 countries by deaths per 100,000 people
top_10_deaths_per_100k <- valid_deaths %>%
  arrange(desc(deaths_per_100k)) %>%
  slice(1:10)

# Output the results
print(top_10_cases_per_100k)
```

```
## # A tibble: 10 × 4
##    `Country/Region` total_cases population cases_per_100k
##    <chr>                  <dbl>      <dbl>          <dbl>
##  1 Andorra              2374138      77000       3083296.
##  2 Montenegro          18129027     621000       2919328.
##  3 Georgia             88580814    3712000       2386337.
##  4 San Marino            764709      34000       2249144.
##  5 Seychelles           2142979      99000       2164625.
##  6 Slovenia            44604578    2101000       2123017.
##  7 Bahrain             36247727    1748000       2073669.
##  8 Serbia             132407349    6863000       1929293.
##  9 Luxembourg          11951812     638000       1873325.
## 10 Israel             169986043    9357000       1816672.
```

```
print(top_10_deaths_per_100k)
```

```
## # A tibble: 10 × 4
##    `Country/Region`       total_deaths population deaths_per_100k
##    <chr>                         <dbl>      <dbl>           <dbl>
##  1 Peru                       29585844   33359000          88689.
##  2 Bosnia and Herzegovina      1444655    3263000          44274.
##  3 San Marino                    14747      34000          43374.
##  4 North Macedonia              893778    2072000          43136.
##  5 Bulgaria                    2965646    6882000          43093.
##  6 Montenegro                   262789     621000          42317.
##  7 Moldova                     1065117    2614000          40747.
##  8 Hungary                     3772729    9721000          38810.
##  9 United Kingdom             25960104   67503000          38458.
## 10 Belgium                     4406995   11579000          38060.
```

– Communicate your methodology, results, and interpretation here –

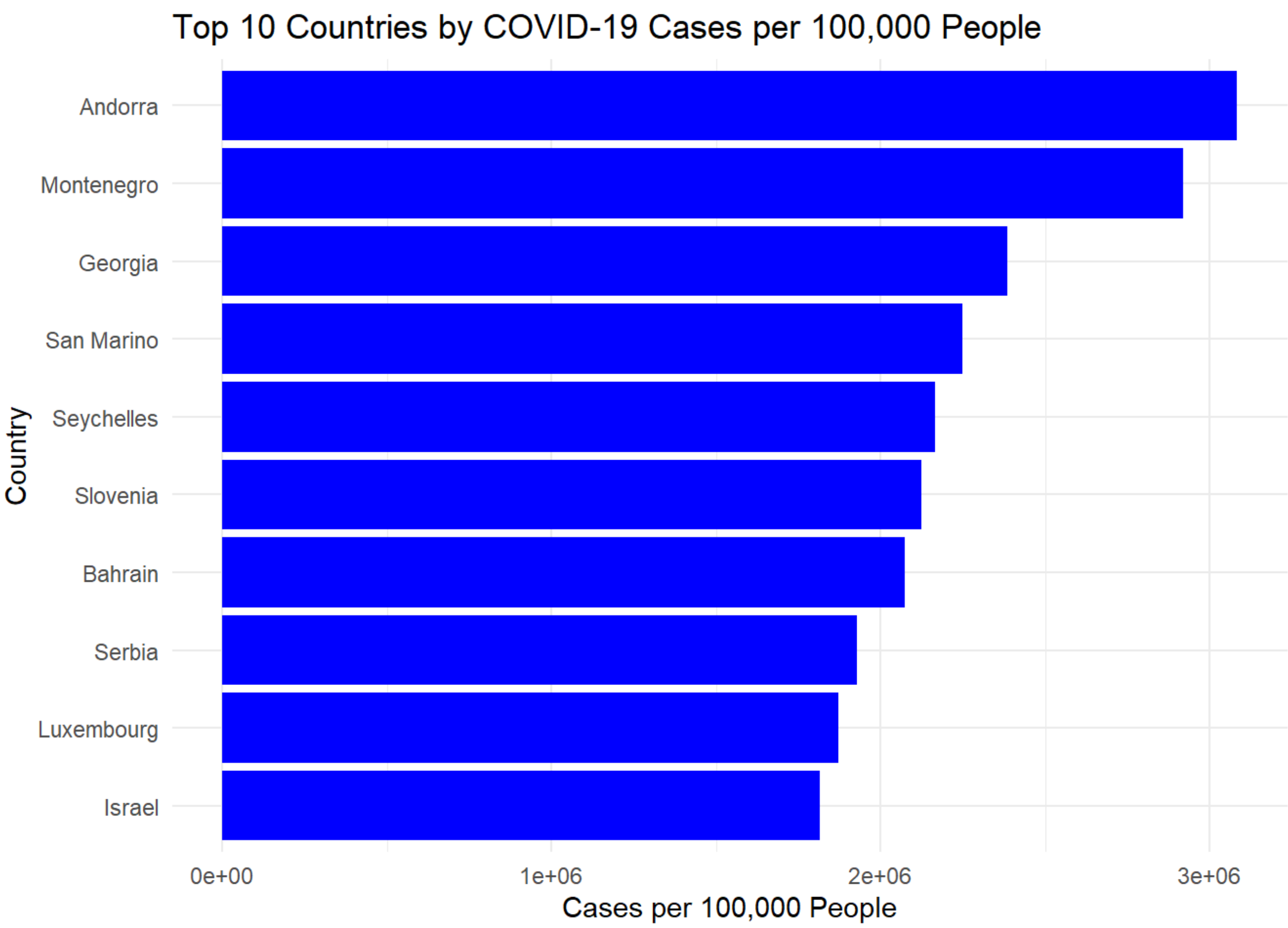
Combine and Tidy Data -> Tidy Population Estimates -> Calculate Statistics -> Filter Data -> Identify Top 10 Countries

The results will show the top 10 countries in terms of COVID-19 cases and deaths per 100,000 people between March 15, 2020, and December 31, 2021. This analysis helps to understand which countries were most affected by the pandemic relative to their population size. It provides insights into the global impact of COVID-19 and helps identify regions that may require more attention and resources.

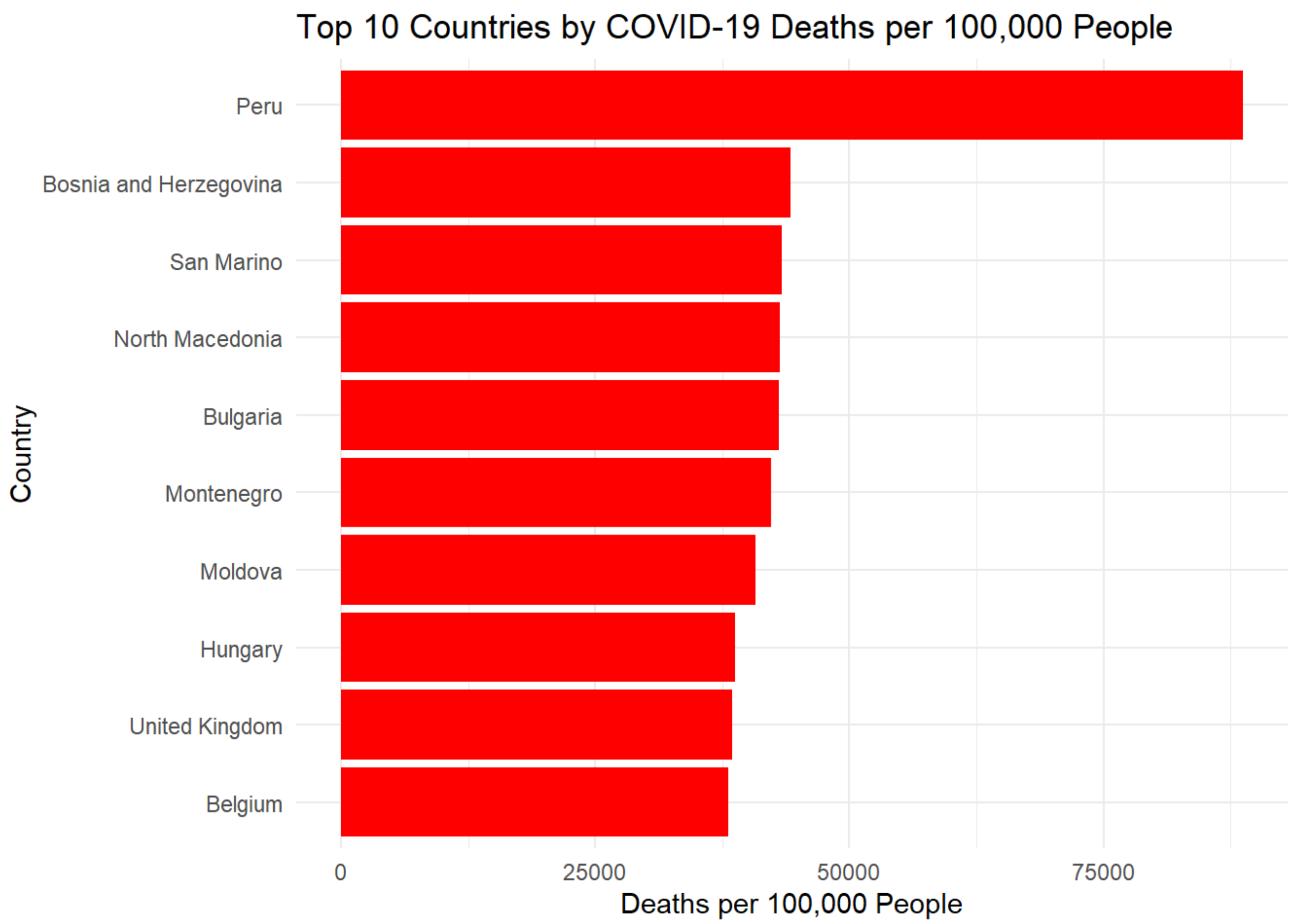
Question 3

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.

```
# Visualization for cases per 100,000 people
ggplot(top_10_cases_per_100k, aes(x = reorder(`Country/Region`, cases_per_100k), y = cases
_per_100k)) +
  geom_bar(stat = "identity", fill = "blue") +
  coord_flip() +
  labs(
    title = "Top 10 Countries by COVID-19 Cases per 100,000 People",
    x = "Country",
    y = "Cases per 100,000 People"
  ) +
  theme_minimal()
```




```r
# Visualization for deaths per 100,000 people
ggplot(top_10_deaths_per_100k, aes(x = reorder(`Country/Region`, deaths_per_100k), y = dea
ths_per_100k)) +
  geom_bar(stat = "identity", fill = "red") +
  coord_flip() +
  labs(
    title = "Top 10 Countries by COVID-19 Deaths per 100,000 People",
    x = "Country",
    y = "Deaths per 100,000 People"
  ) +
  theme_minimal()
```



– Communicate your methodology, results, and interpretation here –

Same as Q2, add in Visualize Data.

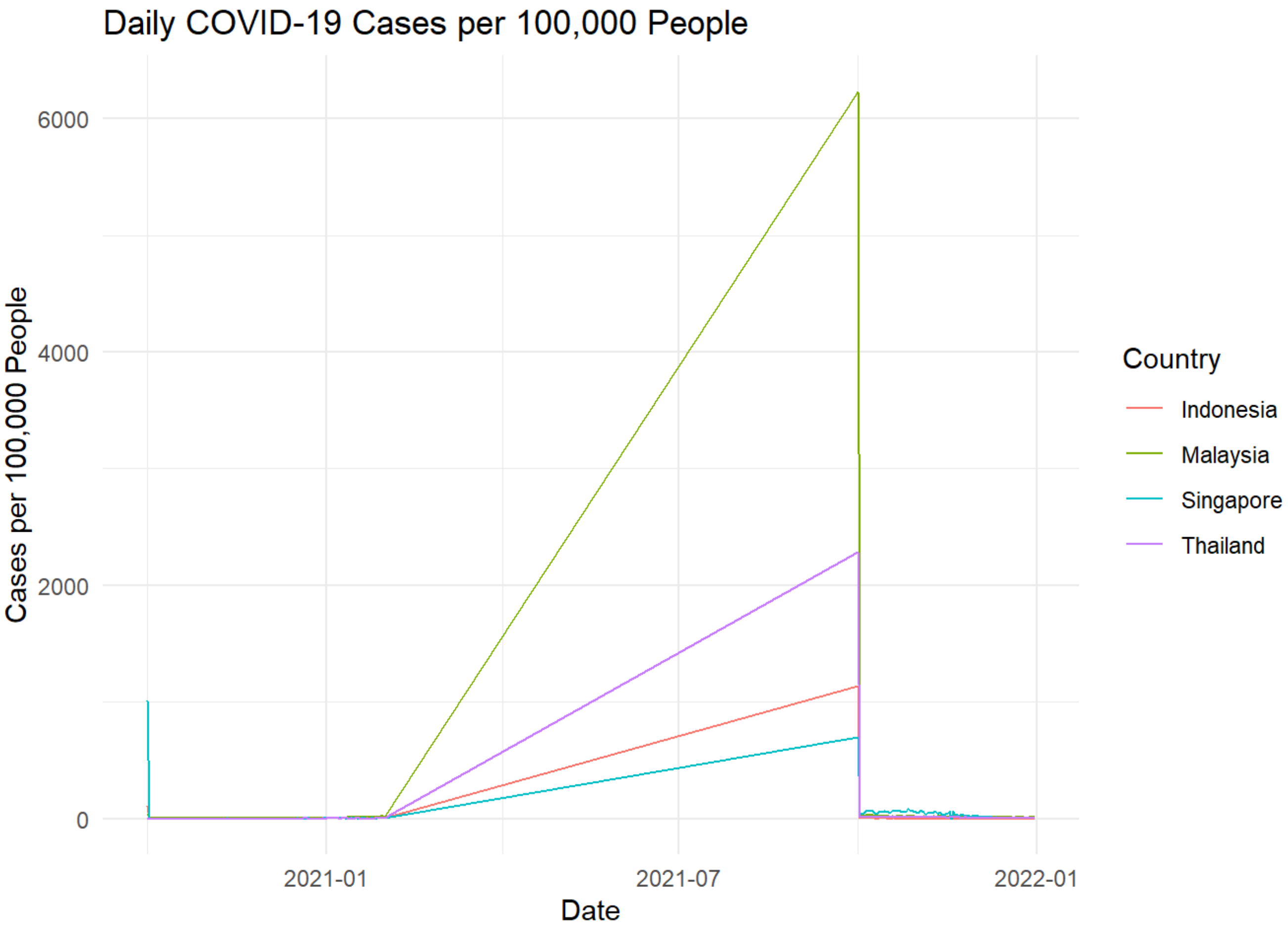
The visualizations show the top 10 countries in terms of COVID-19 cases and deaths per 100,000 people between March 15, 2020, and December 31, 2021. These visualizations provide insights into which countries were most affected by the pandemic relative to their population size. This information is useful for understanding the global impact of COVID-19 and identifying regions that may require more attention and resources.

Question 4

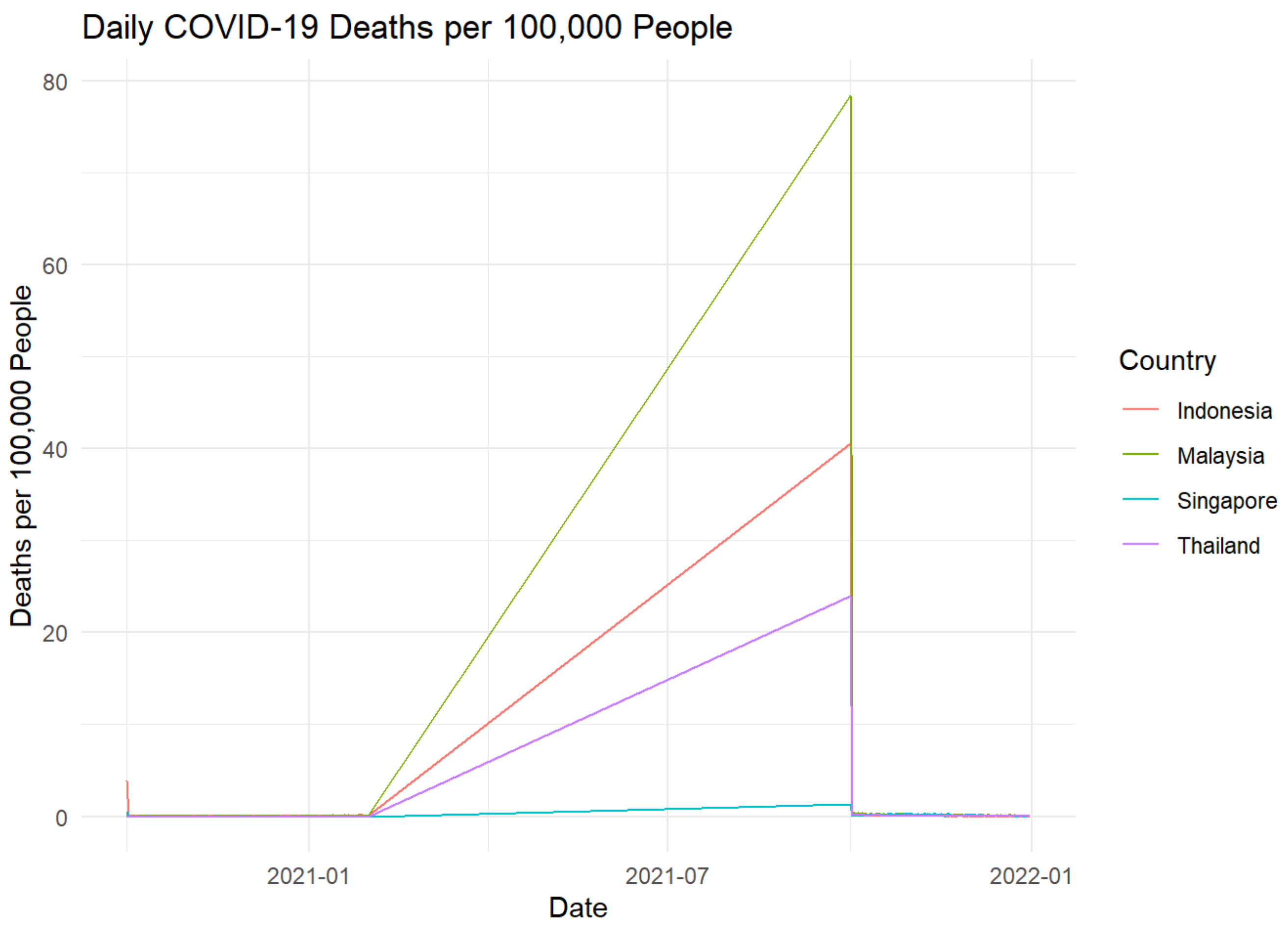
Finally, select four countries from one continent and create visualizations for the daily number of confirmed cases per 100,000 and the daily number of deaths per 100,000 people between March 15, 2020, and December 31, 2021.

```r
# Filter data for the selected countries
selected_countries <- c("Indonesia", "Malaysia", "Singapore", "Thailand")

# Transform global cases data
global_cases <- csse_global_cases %>%
  filter(`Country/Region` %in% selected_countries) %>%
  pivot_longer(cols = starts_with("1"), names_to = "date", values_to = "total_cases") %>%
  mutate(date = mdy(date)) %>%
  select(`Country/Region`, date, total_cases)

# Transform global deaths data
global_deaths <- csse_global_deaths %>%
  filter(`Country/Region` %in% selected_countries) %>%
  pivot_longer(cols = starts_with("1"), names_to = "date", values_to = "total_deaths") %>%
  mutate(date = mdy(date)) %>%
  select(`Country/Region`, date, total_deaths)

# Tidy the population data
tidy_population <- global_population_estimates %>%
  rename(`Country/Region` = `Country Name`, population = `2021 [YR2021]`) %>%
  select(`Country/Region`, population) %>%
  mutate(population = as.numeric(population))

# Join cases and deaths data with population estimates
cases_with_population <- global_cases %>%
  inner_join(tidy_population, by = "Country/Region")

deaths_with_population <- global_deaths %>%
  inner_join(tidy_population, by = "Country/Region")

# Calculate daily new cases and deaths
cases_with_population <- cases_with_population %>%
  group_by(`Country/Region`) %>%
  arrange(date) %>%
  mutate(new_cases = total_cases - lag(total_cases, default = 0)) %>%
  mutate(cases_per_100k = (new_cases / population) * 100000)

deaths_with_population <- deaths_with_population %>%
  group_by(`Country/Region`) %>%
  arrange(date) %>%
  mutate(new_deaths = total_deaths - lag(total_deaths, default = 0)) %>%
  mutate(deaths_per_100k = (new_deaths / population) * 100000)

# Filter for the date range between March 15, 2020, and December 31, 2021
cases_with_population <- cases_with_population %>%
  filter(date >= "2020-03-15" & date <= "2021-12-31")

deaths_with_population <- deaths_with_population %>%
  filter(date >= "2020-03-15" & date <= "2021-12-31")

# Visualizations
# Cases per 100,000 people
ggplot(cases_with_population, aes(x = date, y = cases_per_100k, color = `Country/Region`))
+
```

```
  geom_line() +
  labs(
    title = "Daily COVID-19 Cases per 100,000 People",
    x = "Date",
    y = "Cases per 100,000 People",
    color = "Country"
  ) +
  theme_minimal()
```




```
# Deaths per 100,000 people
ggplot(deaths_with_population, aes(x = date, y = deaths_per_100k, color = `Country/Region
`)) +
  geom_line() +
  labs(
    title = "Daily COVID-19 Deaths per 100,000 People",
    x = "Date",
    y = "Deaths per 100,000 People",
    color = "Country"
  ) +
  theme_minimal()
```



– Communicate your methodology, results, and interpretation here –

Data Preparation -> Population Data -> Calculate Daily Statistics -> Visualization

The visualizations show the daily number of confirmed COVID-19 cases and deaths per 100,000 people for Indonesia, Malaysia, Singapore, and Thailand between March 15, 2020, and December 31, 2021. These visualizations provide insights into the trends and severity of the pandemic in these countries over time. This information can help policymakers and health officials understand the spread and impact of COVID-19 and make informed decisions to mitigate its effects.

Part 1 - Basic Exploration of US Data

Project Description

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 Libraries

```
import numpy as np
from numpy import count_nonzero, median, mean
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import random

#Plotly
import plotly.express as px
import plotly.offline as py
import plotly.graph_objs as go

import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.formula.api import ols
import researchpy as rp

import datetime
from datetime import datetime, timedelta

# import eli5
# from IPython.display import display

#import os
#import zipfile
import scipy.stats
```

```
from collections import Counter

import sklearn
# from sklearn.preprocessing import StandardScaler, MinMaxScaler,
LabelEncoder, OneHotEncoder
# from sklearn.linear_model import LinearRegression,
LogisticRegression, ElasticNet, Lasso, Ridge
# from sklearn.model_selection import cross_val_score,
train_test_split
# from sklearn.metrics import accuracy_score, auc,
classification_report, confusion_matrix, f1_score
# from sklearn.metrics import plot_confusion_matrix, plot_roc_curve

# from sklearn.linear_model import ElasticNet, Lasso,
LinearRegression, LogisticRegression, Ridge
# from sklearn.tree import DecisionTreeClassifier,
DecisionTreeRegressor, ExtraTreeClassifier, ExtraTreeRegressor,
plot_tree
# from sklearn.svm import SVC, SVR, LinearSVC, LinearSVR
# from sklearn.naive_bayes import GaussianNB, MultinomialNB

%matplotlib inline
#sets the default autosave frequency in seconds
%autosave 60
sns.set_style('dark')
sns.set(font_scale=1.2)

plt.rc('axes', titlesize=9)
plt.rc('axes', labelsize=14)
plt.rc('xtick', labelsize=12)
plt.rc('ytick', labelsize=12)

import warnings
warnings.filterwarnings('ignore')

# Use Feature-Engine library
#import feature_engine
#from feature_engine import imputation as mdi
#from feature_engine.outlier_removers import Winsorizer
#from feature_engine import categorical_encoders as ce
#from feature_engine.discretisation import EqualWidthDiscretiser,
EqualFrequencyDiscretiser
#from feature_engine.discretisation import ArbitraryDiscretiser,
DecisionTreeDiscretiser
#from feature_engine.encoding import OrdinalEncoder

pd.set_option('display.max_columns',None)
#pd.set_option('display.max_rows',None)
pd.set_option('display.width', 1000)
pd.set_option('display.float_format','{:.2f}'.format)
```

```
random.seed(0)
np.random.seed(0)
np.set_printoptions(suppress=True)
```

```
Autosaving every 60 seconds
```

Exploratory Data Analysis

```
df1 = pd.read_csv("us-counties-2020.csv",parse_dates=['date'])
```

```
df1
```

```
             date     county       state     fips  cases  deaths
0      2020-01-21  Snohomish  Washington 53061.00      1    0.00
1      2020-01-22  Snohomish  Washington 53061.00      1    0.00
2      2020-01-23  Snohomish  Washington 53061.00      1    0.00
3      2020-01-24       Cook    Illinois 17031.00      1    0.00
4      2020-01-24  Snohomish  Washington 53061.00      1    0.00
...           ...        ...         ...      ...    ...     ...
884732 2020-12-31 Sweetwater     Wyoming 56037.00   2966   16.00
884733 2020-12-31      Teton     Wyoming 56039.00   2138    4.00
884734 2020-12-31      Uinta     Wyoming 56041.00   1558    7.00
884735 2020-12-31   Washakie     Wyoming 56043.00    780   19.00
884736 2020-12-31     Weston     Wyoming 56045.00    476    2.00

[884737 rows x 6 columns]
```

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 884737 entries, 0 to 884736
Data columns (total 6 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   date    884737 non-null  datetime64[ns]
 1   county  884737 non-null  object
 2   state   884737 non-null  object
 3   fips    876471 non-null  float64
 4   cases   884737 non-null  int64
 5   deaths  865976 non-null  float64
dtypes: datetime64[ns](1), float64(2), int64(1), object(2)
memory usage: 40.5+ MB
```

```
df1.describe()
```

```
           fips     cases    deaths
count 876471.00 884737.00 865976.00
mean   31262.22   1952.32     53.60
std    16295.23  10106.48    451.86
min     1001.00      0.00      0.00
```

```
25%    18183.00     36.00      0.00
50%    29215.00    228.00      4.00
75%    46099.00    993.00     21.00
max    78030.00 770915.00  25144.00
```

```
df1.columns
```

```
Index(['date', 'county', 'state', 'fips', 'cases', 'deaths'],
dtype='object')
```

```
df1.state.unique()
```

```
array(['Washington', 'Illinois', 'California', 'Arizona',
'Massachusetts',
       'Wisconsin', 'Texas', 'Nebraska', 'Utah', 'Oregon', 'Florida',
       'New York', 'Rhode Island', 'Georgia', 'New Hampshire',
       'North Carolina', 'New Jersey', 'Colorado', 'Maryland',
'Nevada',
       'Tennessee', 'Hawaii', 'Indiana', 'Kentucky', 'Minnesota',
       'Oklahoma', 'Pennsylvania', 'South Carolina',
       'District of Columbia', 'Kansas', 'Missouri', 'Vermont',
       'Virginia', 'Connecticut', 'Iowa', 'Louisiana', 'Ohio',
'Michigan',
       'South Dakota', 'Arkansas', 'Delaware', 'Mississippi',
       'New Mexico', 'North Dakota', 'Wyoming', 'Alaska', 'Maine',
       'Alabama', 'Idaho', 'Montana', 'Puerto Rico', 'Virgin Islands',
       'Guam', 'West Virginia', 'Northern Mariana Islands'],
dtype=object)
```

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.

```
df1["date"] == "2020-03-15"
```

```
0         False
1         False
2         False
3         False
4         False
          ...
884732    False
884733    False
884734    False
884735    False
```

```
884736    False
Name: date, Length: 884737, dtype: bool

df2020 = df1[df1["date"] >= "2020-03-15"]

df2020

              date      county    state      fips  cases  deaths
2309   2020-03-15     Baldwin  Alabama  1003.00      1    0.00
2310   2020-03-15      Elmore  Alabama  1051.00      1    0.00
2311   2020-03-15   Jefferson  Alabama  1073.00     13    0.00
2312   2020-03-15         Lee  Alabama  1081.00      1    0.00
2313   2020-03-15   Limestone  Alabama  1083.00      1    0.00
...           ...         ...      ...       ...    ...     ...
884732 2020-12-31  Sweetwater  Wyoming 56037.00   2966   16.00
884733 2020-12-31       Teton  Wyoming 56039.00   2138    4.00
884734 2020-12-31       Uinta  Wyoming 56041.00   1558    7.00
884735 2020-12-31    Washakie  Wyoming 56043.00    780   19.00
884736 2020-12-31      Weston  Wyoming 56045.00    476    2.00

[882428 rows x 6 columns]

df2020.reset_index(inplace=True, drop=True)

df2020

              date      county    state      fips  cases  deaths
0      2020-03-15     Baldwin  Alabama  1003.00      1    0.00
1      2020-03-15      Elmore  Alabama  1051.00      1    0.00
2      2020-03-15   Jefferson  Alabama  1073.00     13    0.00
3      2020-03-15         Lee  Alabama  1081.00      1    0.00
4      2020-03-15   Limestone  Alabama  1083.00      1    0.00
...           ...         ...      ...       ...    ...     ...
882423 2020-12-31  Sweetwater  Wyoming 56037.00   2966   16.00
882424 2020-12-31       Teton  Wyoming 56039.00   2138    4.00
882425 2020-12-31       Uinta  Wyoming 56041.00   1558    7.00
882426 2020-12-31    Washakie  Wyoming 56043.00    780   19.00
882427 2020-12-31      Weston  Wyoming 56045.00    476    2.00

[882428 rows x 6 columns]

max_date = df2020.date.max()
max_date

Timestamp('2020-12-31 00:00:00')

us_total_cases = df2020.groupby("date")["cases"].sum()
us_total_cases

date
2020-03-15        3600
2020-03-16        4507
```

```
2020-03-17       5906
2020-03-18       8350
2020-03-19      12393
                 ...
2020-12-27   19174788
2020-12-28   19363798
2020-12-29   19564828
2020-12-30   19793777
2020-12-31   20024801
Name: cases, Length: 292, dtype: int64
```

```python
us_total_deaths = df2020.groupby("date")["deaths"].sum()
us_total_deaths
```

```
date
2020-03-15       68.00
2020-03-16       91.00
2020-03-17      117.00
2020-03-18      162.00
2020-03-19      212.00
                 ...
2020-12-27   333253.00
2020-12-28   335152.00
2020-12-29   338780.00
2020-12-30   342588.00
2020-12-31   346050.00
Name: deaths, Length: 292, dtype: float64
```

Question 2

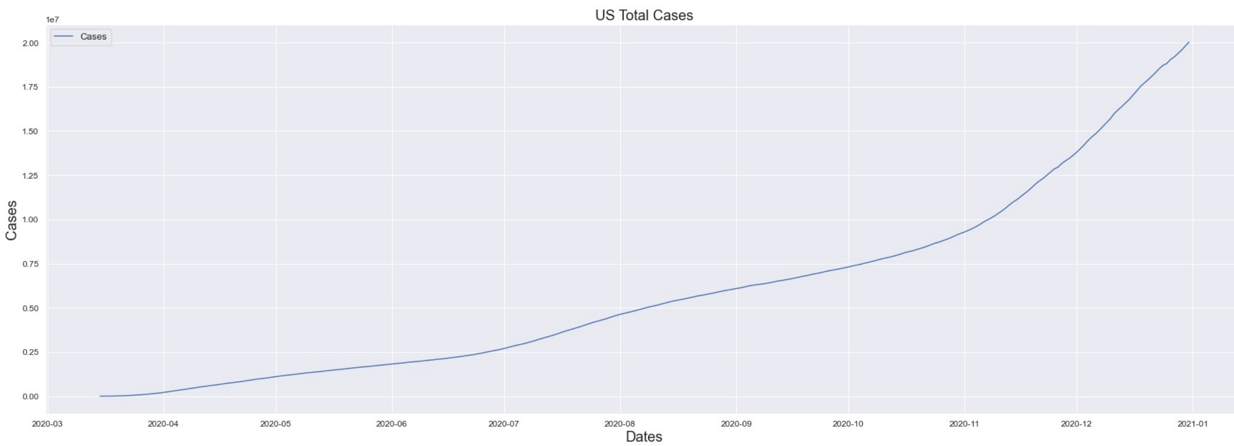
Create a visualization for the total number of deaths and cases in the US since March 15, 2020. Before you create your visualization, review the types of plots you can create using the ggplot2 library and think about which plots would be effective in communicating your results. After you have created your visualization, write a few sentences describing your visualization. How could the plot be interpreted? Could it be misleading?

```python
us_total_cases = pd.DataFrame(us_total_cases)

us_total_cases
```

```
               cases
date
2020-03-15      3600
2020-03-16      4507
2020-03-17      5906
2020-03-18      8350
2020-03-19     12393
...              ...
2020-12-27  19174788
2020-12-28  19363798
```

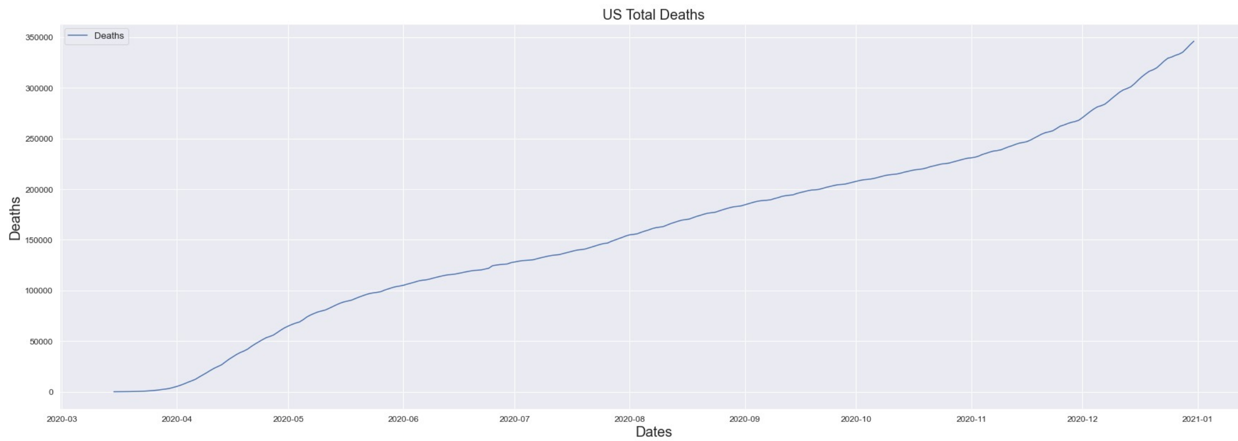
```
2020-12-29  19564828
2020-12-30  19793777
2020-12-31  20024801

[292 rows x 1 columns]
```

```
fig = plt.figure(figsize=(30,10))
sns.lineplot(x=us_total_cases.index,y=us_total_cases.cases,data=us_tot
al_cases, estimator=None)
plt.title("US Total Cases", fontsize=20)
plt.xlabel("Dates", fontsize=20)
plt.ylabel("Cases", fontsize=20)
plt.legend(['Cases'])
plt.show()
```




```
us_total_deaths = pd.DataFrame(us_total_deaths)

us_total_deaths

               deaths
date
2020-03-15      68.00
2020-03-16      91.00
2020-03-17     117.00
2020-03-18     162.00
2020-03-19     212.00
...               ...
2020-12-27 333253.00
2020-12-28 335152.00
2020-12-29 338780.00
2020-12-30 342588.00
2020-12-31 346050.00

[292 rows x 1 columns]
```

```
fig = plt.figure(figsize=(30,10))
sns.lineplot(x=us_total_deaths.index,y=us_total_deaths.deaths,data=us_total_deaths, estimator=None)
plt.title("US Total Deaths", fontsize=20)
plt.xlabel("Dates", fontsize=20)
plt.ylabel("Deaths", fontsize=20)
plt.legend(['Deaths'])
plt.show()
```



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.

```
us_total_cases
```

	cases
date	
2020-03-15	3600
2020-03-16	4507
2020-03-17	5906
2020-03-18	8350
2020-03-19	12393
...	...
2020-12-27	19174788
2020-12-28	19363798
2020-12-29	19564828
2020-12-30	19793777
2020-12-31	20024801

```
[292 rows x 1 columns]

us_total_cases["diff_cases"] = us_total_cases.diff()

us_total_cases

                 cases  diff_cases
date
2020-03-15        3600         NaN
2020-03-16        4507      907.00
2020-03-17        5906     1399.00
2020-03-18        8350     2444.00
2020-03-19       12393     4043.00
...                ...         ...
2020-12-27  19174788   152089.00
2020-12-28  19363798   189010.00
2020-12-29  19564828   201030.00
2020-12-30  19793777   228949.00
2020-12-31  20024801   231024.00

[292 rows x 2 columns]

us_total_cases.diff_cases.max()

280016.0

max_new_cases_date = us_total_cases[us_total_cases.diff_cases ==
280016.0]

max_new_cases_date

                 cases  diff_cases
date
2020-12-11  15977147   280016.00

us_total_deaths["diff_deaths"] = us_total_deaths.diff()

us_total_deaths

               deaths  diff_deaths
date
2020-03-15      68.00          NaN
2020-03-16      91.00        23.00
2020-03-17     117.00        26.00
2020-03-18     162.00        45.00
2020-03-19     212.00        50.00
...               ...          ...
2020-12-27 333253.00      1230.00
2020-12-28 335152.00      1899.00
2020-12-29 338780.00      3628.00
2020-12-30 342588.00      3808.00
```

```
2020-12-31 346050.00      3462.00

[292 rows x 2 columns]
```

```
us_total_deaths.diff_deaths.max()
```

```
3808.0
```

```
max_new_deaths_date = us_total_deaths[us_total_deaths.diff_deaths ==
3808.0]

max_new_deaths_date
```

```
              deaths  diff_deaths
date
2020-12-30 342588.00      3808.00
```

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.

Question 5.

Create a visualization for the seven-day averages for new cases and deaths per 100,000 people in the United States.

Python code done by Dennis Lam

Part 1 - Basic Exploration of US Data

Project Description

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.

Import Libraries

```
import numpy as np
from numpy import count_nonzero, median, mean
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import random

#Plotly
import plotly.express as px
import plotly.offline as py
import plotly.graph_objs as go

import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.formula.api import ols
import researchpy as rp

import datetime
from datetime import datetime, timedelta

# import eli5
# from IPython.display import display

#import os
#import zipfile
import scipy.stats
from collections import Counter


%matplotlib inline
#sets the default autosave frequency in seconds
%autosave 60
sns.set_style('dark')
sns.set(font_scale=1.2)
```

```
plt.rc('axes', titlesize=9)
plt.rc('axes', labelsize=14)
plt.rc('xtick', labelsize=12)
plt.rc('ytick', labelsize=12)

import warnings
warnings.filterwarnings('ignore')

pd.set_option('display.max_columns',None)
#pd.set_option('display.max_rows',None)
pd.set_option('display.width', 1000)
pd.set_option('display.float_format','{:.2f}'.format)

random.seed(0)
np.random.seed(0)
np.set_printoptions(suppress=True)
```

```
Autosaving every 60 seconds
```

Question 1.

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

```
df1 = pd.read_csv("us-counties-2020.csv",parse_dates=['date'])

df1
```

```
             date      county       state     fips  cases  deaths
0      2020-01-21   Snohomish  Washington 53061.00      1    0.00
1      2020-01-22   Snohomish  Washington 53061.00      1    0.00
2      2020-01-23   Snohomish  Washington 53061.00      1    0.00
3      2020-01-24        Cook    Illinois 17031.00      1    0.00
4      2020-01-24   Snohomish  Washington 53061.00      1    0.00
...           ...         ...         ...      ...    ...     ...
884732 2020-12-31  Sweetwater     Wyoming 56037.00   2966   16.00
884733 2020-12-31       Teton     Wyoming 56039.00   2138    4.00
884734 2020-12-31       Uinta     Wyoming 56041.00   1558    7.00
884735 2020-12-31    Washakie     Wyoming 56043.00    780   19.00
884736 2020-12-31      Weston     Wyoming 56045.00    476    2.00

[884737 rows x 6 columns]
```

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 884737 entries, 0 to 884736
Data columns (total 6 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
```

```
 0   date    884737 non-null  datetime64[ns]
 1   county  884737 non-null  object
 2   state   884737 non-null  object
 3   fips    876471 non-null  float64
 4   cases   884737 non-null  int64
 5   deaths  865976 non-null  float64
dtypes: datetime64[ns](1), float64(2), int64(1), object(2)
memory usage: 40.5+ MB
```

```
df1.describe()
```

```
           fips      cases    deaths
count 876471.00 884737.00 865976.00
mean   31262.22   1952.32     53.60
std    16295.23  10106.48    451.86
min     1001.00      0.00      0.00
25%    18183.00     36.00      0.00
50%    29215.00    228.00      4.00
75%    46099.00    993.00     21.00
max    78030.00 770915.00  25144.00
```

```
df1.columns
```

```
Index(['date', 'county', 'state', 'fips', 'cases', 'deaths'],
dtype='object')
```

```
df2020 = df1[df1["date"] >= "2020-03-15"]
```

```
df2020
```

```
             date     county    state     fips  cases  deaths
2309   2020-03-15    Baldwin  Alabama  1003.00      1    0.00
2310   2020-03-15     Elmore  Alabama  1051.00      1    0.00
2311   2020-03-15  Jefferson  Alabama  1073.00     13    0.00
2312   2020-03-15        Lee  Alabama  1081.00      1    0.00
2313   2020-03-15  Limestone  Alabama  1083.00      1    0.00
...           ...        ...      ...      ...    ...     ...
884732 2020-12-31 Sweetwater  Wyoming 56037.00   2966   16.00
884733 2020-12-31      Teton  Wyoming 56039.00   2138    4.00
884734 2020-12-31      Uinta  Wyoming 56041.00   1558    7.00
884735 2020-12-31   Washakie  Wyoming 56043.00    780   19.00
884736 2020-12-31     Weston  Wyoming 56045.00    476    2.00

[882428 rows x 6 columns]
```

```
df2020.reset_index(inplace=True, drop=True)
```

```
df2020
```

```
             date     county    state     fips  cases  deaths
0      2020-03-15    Baldwin  Alabama  1003.00      1    0.00
1      2020-03-15     Elmore  Alabama  1051.00      1    0.00
```

```
2      2020-03-15   Jefferson  Alabama  1073.00     13    0.00
3      2020-03-15         Lee  Alabama  1081.00      1    0.00
4      2020-03-15   Limestone  Alabama  1083.00      1    0.00
...           ...         ...      ...      ...    ...     ...
882423 2020-12-31  Sweetwater  Wyoming 56037.00   2966   16.00
882424 2020-12-31       Teton  Wyoming 56039.00   2138    4.00
882425 2020-12-31       Uinta  Wyoming 56041.00   1558    7.00
882426 2020-12-31    Washakie  Wyoming 56043.00    780   19.00
882427 2020-12-31      Weston  Wyoming 56045.00    476    2.00

[882428 rows x 6 columns]
```

```
state_total_cases = df2020.groupby("state")["cases"].sum()
state_total_cases
```

```
state
Alabama                      32235982
Alaska                        2854124
Arizona                      47146971
Arkansas                     18406345
California                  174966840
Colorado                     23441715
Connecticut                  17214186
Delaware                      5154364
District of Columbia          3644565
Florida                     138122838
Georgia                      62229960
Guam                           869347
Hawaii                        2129435
Idaho                        10608652
Illinois                     82118017
Indiana                      35947293
Iowa                         23267631
Kansas                       15948003
Kentucky                     18801501
Louisiana                    35116592
Maine                         1551404
Maryland                     29150884
Massachusetts                37418616
Michigan                     42112778
Minnesota                    29868329
Mississippi                  20500985
Missouri                     31469316
Montana                       5007372
Nebraska                     13158031
Nevada                       18286646
New Hampshire                 2724654
New Jersey                   57403086
New Mexico                    9586639
New York                    126305713
```

```
North Carolina               46306294
North Dakota                  6682194
Northern Mariana Islands        15491
Ohio                         44885170
Oklahoma                     20512523
Oregon                        8343426
Pennsylvania                 47304799
Puerto Rico                   9730408
Rhode Island                  7356944
South Carolina               29321120
South Dakota                  7219418
Tennessee                    43836115
Texas                       160158661
Utah                         20001534
Vermont                        561040
Virgin Islands                 216309
Virginia                     32850426
Washington                   21934366
West Virginia                 4681431
Wisconsin                    37998711
Wyoming                       2591441
Name: cases, dtype: int64
```

```
state_total_cases.sort_values().nlargest(10)
```

```
state
California        174966840
Texas             160158661
Florida           138122838
New York          126305713
Illinois           82118017
Georgia            62229960
New Jersey         57403086
Pennsylvania       47304799
Arizona            47146971
North Carolina     46306294
Name: cases, dtype: int64
```

```
state_total_deaths = df2020.groupby("state")["deaths"].sum()
state_total_deaths
```

```
state
Alabama                     526388.00
Alaska                       13147.00
Arizona                    1048821.00
Arkansas                    285868.00
California                 3065085.00
Colorado                    542865.00
Connecticut                1119863.00
Delaware                    145664.00
```

```
District of Columbia        146395.00
Florida                    2632219.00
Georgia                    1363974.00
Guam                         10424.00
Hawaii                       27227.00
Idaho                       109143.00
Illinois                   2211390.00
Indiana                     920773.00
Iowa                        335716.00
Kansas                      187626.00
Kentucky                    285739.00
Louisiana                  1184573.00
Maine                        35654.00
Maryland                    912049.00
Massachusetts              2222455.00
Michigan                   1837617.00
Minnesota                   531337.00
Mississippi                 572569.00
Missouri                    530055.00
Montana                      58866.00
Nebraska                    132992.00
Nevada                      319370.00
New Hampshire               102996.00
New Jersey                 3817144.00
New Mexico                  217023.00
New York                   8320596.00
North Carolina              734456.00
North Dakota                 87065.00
Northern Mariana Islands       544.00
Ohio                       1062373.00
Oklahoma                    233760.00
Oregon                      128395.00
Pennsylvania               2002629.00
Puerto Rico                 134516.00
Rhode Island                262024.00
South Carolina              618708.00
South Dakota                 86823.00
Tennessee                   540030.00
Texas                      2927740.00
Utah                        111490.00
Vermont                      16434.00
Virgin Islands                3503.00
Virginia                    668989.00
Washington                  504498.00
West Virginia                84127.00
Wisconsin                   410977.00
Wyoming                      20734.00
Name: deaths, dtype: float64
```

```
state_total_deaths.sort_values().nlargest(10)
```

```
state
New York         8320596.00
New Jersey       3817144.00
California       3065085.00
Texas            2927740.00
Florida          2632219.00
Massachusetts    2222455.00
Illinois         2211390.00
Pennsylvania     2002629.00
Michigan         1837617.00
Georgia          1363974.00
Name: deaths, dtype: float64
```

Question 2.

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

Question 3.

Calculate seven-day averages for new cases and deaths per 100,000 people in a state of your choice.

Question 4.

Identify the top 5 counties in terms of deaths and cases per 100,000 people in the state used in Question 2.

Question 5

Modify the exisiting code to produce a county-level map projection of deaths and cases per 100,000 people of the state used in Question 2.

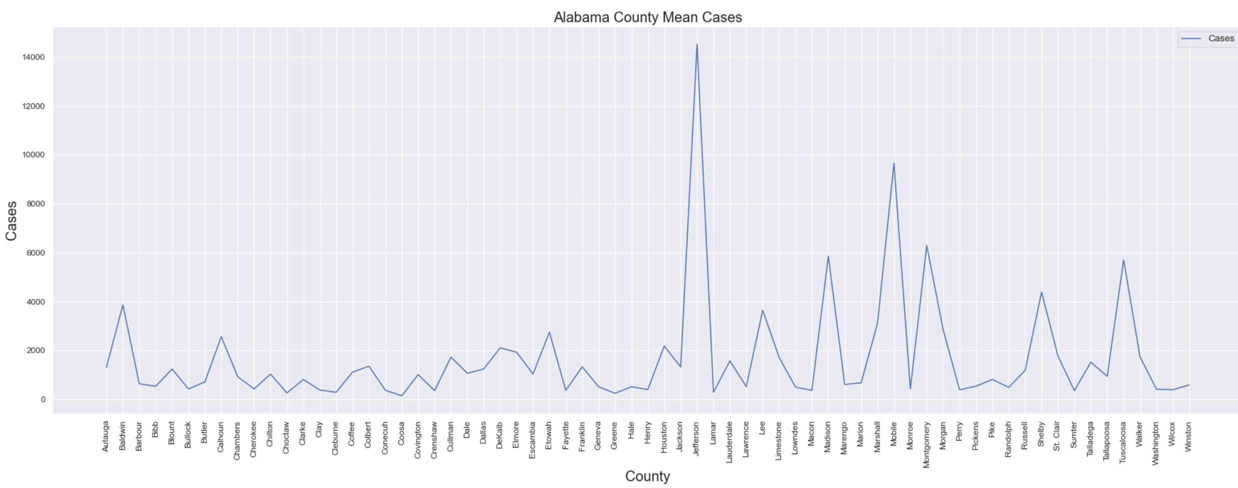
```
alabama = df2020[df2020["state"] == "Alabama"]

alabama
```

	date	county	state	fips	cases	deaths
0	2020-03-15	Baldwin	Alabama	1003.00	1	0.00
1	2020-03-15	Elmore	Alabama	1051.00	1	0.00
2	2020-03-15	Jefferson	Alabama	1073.00	13	0.00
3	2020-03-15	Lee	Alabama	1081.00	1	0.00
4	2020-03-15	Limestone	Alabama	1083.00	1	0.00
...	...	...	...	...	...	...
879245	2020-12-31	Tuscaloosa	Alabama	1125.00	18468	218.00
879246	2020-12-31	Walker	Alabama	1127.00	5259	138.00
879247	2020-12-31	Washington	Alabama	1129.00	1184	24.00
879248	2020-12-31	Wilcox	Alabama	1131.00	883	19.00
879249	2020-12-31	Winston	Alabama	1133.00	1968	30.00

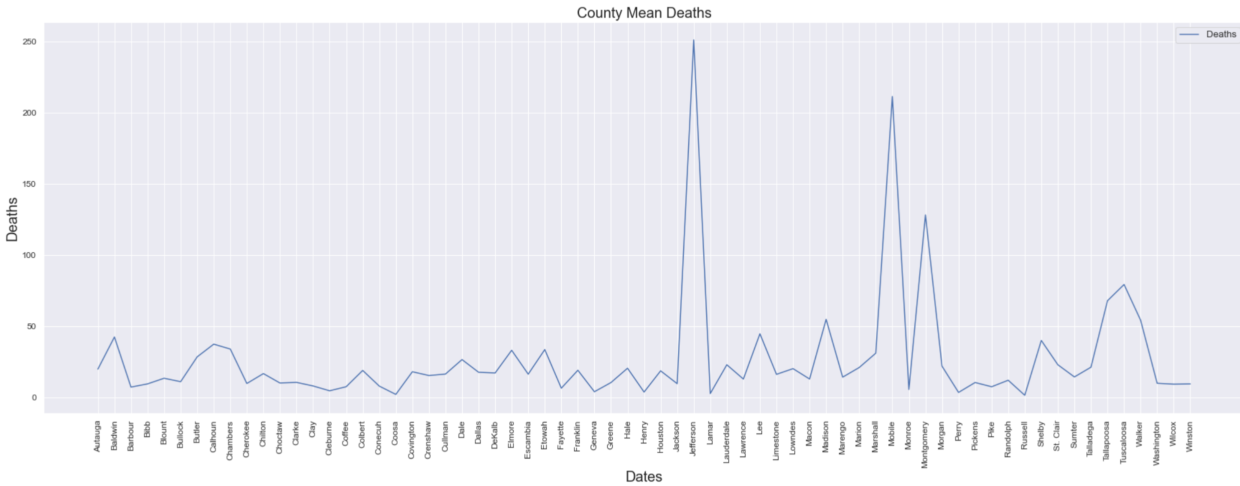
```
[18935 rows x 6 columns]

alabama.reset_index(inplace=True, drop=True)

alabama

              date      county    state    fips  cases  deaths
0      2020-03-15     Baldwin  Alabama 1003.00      1    0.00
1      2020-03-15      Elmore  Alabama 1051.00      1    0.00
2      2020-03-15   Jefferson  Alabama 1073.00     13    0.00
3      2020-03-15         Lee  Alabama 1081.00      1    0.00
4      2020-03-15   Limestone  Alabama 1083.00      1    0.00
...           ...         ...      ...     ...    ...     ...
18930 2020-12-31  Tuscaloosa  Alabama 1125.00  18468  218.00
18931 2020-12-31      Walker  Alabama 1127.00   5259  138.00
18932 2020-12-31  Washington  Alabama 1129.00   1184   24.00
18933 2020-12-31      Wilcox  Alabama 1131.00    883   19.00
18934 2020-12-31     Winston  Alabama 1133.00   1968   30.00

[18935 rows x 6 columns]

alabama.county.unique()

array(['Baldwin', 'Elmore', 'Jefferson', 'Lee', 'Limestone',
'Montgomery',
       'Shelby', 'Tuscaloosa', 'Madison', 'St. Clair', 'Calhoun',
       'Talladega', 'Chambers', 'Mobile', 'Walker', 'Cullman',
'Jackson',
       'Lamar', 'Lauderdale', 'Washington', 'Marion', 'Franklin',
       'Houston', 'Tallapoosa', 'Autauga', 'Morgan', 'Blount',
'Butler',
       'Cherokee', 'Chilton', 'Clay', 'Cleburne', 'Colbert', 'Dallas',
       'Etowah', 'Lawrence', 'Marshall', 'Pickens', 'Pike', 'Russell',
       'Wilcox', 'Bullock', 'Choctaw', 'Coosa', 'Crenshaw', 'DeKalb',
       'Lowndes', 'Marengo', 'Covington', 'Escambia', 'Greene',
       'Randolph', 'Winston', 'Monroe', 'Macon', 'Bibb', 'Fayette',
       'Hale', 'Sumter', 'Clarke', 'Conecuh', 'Dale', 'Coffee',
'Barbour',
       'Henry', 'Perry', 'Geneva'], dtype=object)

county = alabama.groupby("county").mean()
county

              fips   cases  deaths
county
Autauga    1001.00 1309.76   20.08
Baldwin    1003.00 3861.32   42.62
Barbour    1005.00  640.44    7.45
Bibb       1007.00  535.09    9.70
Blount     1009.00 1239.25   13.67
```

```
...              ...     ...     ...
Tuscaloosa 1125.00 5701.72   79.50
Walker     1127.00 1738.61   54.42
Washington 1129.00  415.54   10.14
Wilcox     1131.00  392.89    9.52
Winston    1133.00  589.89    9.70

[67 rows x 3 columns]
```

```
fig = plt.figure(figsize=(30,10))
sns.lineplot(x=county.index,y=county.cases,data=county,
estimator=None)
plt.title("Alabama County Mean Cases", fontsize=20)
plt.xlabel("County", fontsize=20)
plt.xticks(rotation = 90)
plt.ylabel("Cases", fontsize=20)
plt.legend(['Cases'])
plt.show()
```




```
fig = plt.figure(figsize=(30,10))
sns.lineplot(x=county.index,y=county.deaths ,data=county,
estimator=None)
plt.title("County Mean Deaths", fontsize=20)
plt.xlabel("Dates", fontsize=20)
plt.xticks(rotation = 90)
plt.ylabel("Deaths", fontsize=20)
plt.legend(['Deaths'])
plt.show()
```



Question 6.

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

Question 7.

Create a visualization to comparing the statistics from Question 6.

Python code done by Dennis Lam

```

```