Peer-graded Assignment: Reproducible Report on COVID-19 Data

Mel Delgado

2024-03-01

Steps in the Data Science Process

The goal of this project is to analyze data about the effects of COVID-19 on the world and the United States. The sections below outline the approach and methodology used by data scientists to answer questions revealed within each iteration as discussed in this course. The steps are import, tidy, and iterate through transforming, visualizing, modeling, and finally communicating results.

What follows is an application of this process as we reveals insights into the data about the effects of COVID-19 on the population.

Step 1 - Importing Data

This is the first step in the data science process where we obtain data in a reproducible way. Doing so means we avoid absolute paths for accessing data if the data exists or is copied to a local disk and use relative paths instead. For this project, we retrieve the data from GitHub via a base URL and construct a fully qualified path for the the different data sets by appending the file name to the base URL as seen below using `str_c` (string concatenate) .

Then, we create four different variables named `global_cases`, `global_deaths`, `US_cases`, and `US_deaths` corresponding to the fully qualified path of the four files found in the variable named `urls`.

```
# Create and store the base URL
url_in <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covi
# Add a list of file names
file_names <- c("time_series_covid19_confirmed_global.csv", "time_series_covid19_deaths_global.csv", "ti
# Call string concatenate to add the base URL and file name to a variable named urls
urls <- str_c(url_in, file_names)

# Store data for each file in the corresponding variable
global_cases <- read_csv(urls[1])
global_deaths <- read_csv(urls[2])
US_cases <- read_csv(urls[3])
US_deaths <- read_csv(urls[4])
```

Step 2 - Tidying and Transforming the Data

Tidying Data

In this next step of the data science process, we want to see what the raw data looks like and clean it up to make it more usable and easier to work with. This could mean only pulling in what is necessary for our analysis or transforming the data in a way that makes it more useful.

For example, at first glance, the data contained in `global_cases` include the columns named `Lat` and `Long` with the latitude and longitude information respectively. We won't need these columns for our analysis. Another observation is the case numbers are stored in columns by date. We can transform the data so the column data is changed to data contained in rows. We want to make all columns be rows of data with the exception of `Province/State`, `Country/Region`, `Lat`, and `Long`. Similar steps are taken to wrangle the data stored in `global_deaths`.

```
# Pivot all columns to rows except `Province/State`, `Country/Region`, `Lat`, and `Long`, exclude `Lat`
global_cases <- global_cases %>%
  pivot_longer(cols = -c('Province/State',
                         'Country/Region', Lat, Long),
               names_to = "date",
               values_to = "cases") %>%
  select(-c(Lat, Long))

# Pivot all columns to rows except `Province/State`, `Country/Region`, `Lat`, and `Long`, exclude `Lat`
global_deaths <- global_deaths %>%
  pivot_longer(cols = -c('Province/State',
                         'Country/Region',
                         Lat, Long),
               names_to = "date",
               values_to = "deaths") %>%
  select(-c(Lat, Long))
```

Transforming Data

We would like to transform the data by joining the cases with the deaths followed by

We accomplish all of these objectives with the following code for the global cases:

```
# Transform the data by joining cases with the deaths, renaming `Country/Region` to `Country_Region`, r
global <- global_cases %>%
  full_join(global_deaths) %>%
  rename(Country_Region = 'Country/Region',
         Province_State = 'Province/State') %>%
  mutate(date = mdy(date))

# The data contain rows with negative cases so filter for only cases that are positive
global <- global %>% filter(cases > 0)

# Take a look at the data to see if what summary describes looks reasonable
summary(global)
```

```
##  Province_State     Country_Region          date                cases
##  Length:306827      Length:306827      Min.   :2020-01-22   Min.   :        1
##  Class :character   Class :character   1st Qu.:2020-12-12   1st Qu.:     1316
##  Mode  :character   Mode  :character   Median :2021-09-16   Median :    20365
##                                        Mean   :2021-09-11   Mean   :  1032863
##                                        3rd Qu.:2022-06-15   3rd Qu.:   271281
##                                        Max.   :2023-03-09   Max.   :103802702
##      deaths
##  Min.   :      0
##  1st Qu.:      7
##  Median :    214
##  Mean   :  14405
##  3rd Qu.:   3665
##  Max.   :1123836
```

```r
# Test the data by looking for cases greater than 100,000,000
global %>% filter(cases > 100000000)
```

```
## # A tibble: 80 x 5
##    Province_State Country_Region date           cases  deaths
##    <chr>          <chr>          <date>         <dbl>   <dbl>
##  1 <NA>           US             2022-12-20 100050937 1088341
##  2 <NA>           US             2022-12-21 100233060 1089383
##  3 <NA>           US             2022-12-22 100329204 1089979
##  4 <NA>           US             2022-12-23 100368433 1090186
##  5 <NA>           US             2022-12-24 100374955 1090208
##  6 <NA>           US             2022-12-25 100378169 1090223
##  7 <NA>           US             2022-12-26 100390601 1090252
##  8 <NA>           US             2022-12-27 100501536 1090608
##  9 <NA>           US             2022-12-28 100614880 1091598
## 10 <NA>           US             2022-12-29 100718983 1092522
## # i 70 more rows
```

A similar approach is taken for the US data whereby we examine the data to understand what we are working with.

```r
# Start building the data by trying out pivoting the UID through Combined_Key to rows to see what it pr
US_cases %>%
  pivot_longer(cols = -(UID:Combined_Key),
               names_to = "date",
               values_to = "cases")
```

```
## # A tibble: 3,819,906 x 13
##         UID iso2  iso3  code3  FIPS Admin2  Province_State Country_Region   Lat
##       <dbl> <chr> <chr> <dbl> <dbl> <chr>   <chr>          <chr>          <dbl>
##  1 84001001 US    USA     840  1001 Autauga Alabama        US              32.5
##  2 84001001 US    USA     840  1001 Autauga Alabama        US              32.5
##  3 84001001 US    USA     840  1001 Autauga Alabama        US              32.5
##  4 84001001 US    USA     840  1001 Autauga Alabama        US              32.5
##  5 84001001 US    USA     840  1001 Autauga Alabama        US              32.5
##  6 84001001 US    USA     840  1001 Autauga Alabama        US              32.5
##  7 84001001 US    USA     840  1001 Autauga Alabama        US              32.5
##  8 84001001 US    USA     840  1001 Autauga Alabama        US              32.5
##  9 84001001 US    USA     840  1001 Autauga Alabama        US              32.5
## 10 84001001 US    USA     840  1001 Autauga Alabama        US              32.5
## # i 3,819,896 more rows
## # i 4 more variables: Long_ <dbl>, Combined_Key <chr>, date <chr>, cases <dbl>
```

Now that we have a better understanding of the data and how we want to transform it, we apply the same logic, select Admin2 through cases, mutate the data from a `<str>` to a `date` object, and remove the columns named `Lat` and `Long_`. Similarly, do the same to the data stored in `US_deaths`.

We combined `US_cases` and `US_deaths` with a `full_join` and store the result in a variable named `US`.

```r
# Building on the last step as an experiment, pivot the columns UID through Combined_Key to rows with c
US_cases <- US_cases %>%
  pivot_longer(cols = -(UID:Combined_Key),
               names_to = "date",
               values_to = "cases") %>%
  select(Admin2:cases) %>%
  mutate(date = mdy(date)) %>%
  select(-c(Lat, Long_))
```

```r
# Similarly, repeat the steps above for `US_deaths`
US_deaths <- US_deaths %>%
  pivot_longer(cols = -(UID:Population),
               names_to = "date",
               values_to = "deaths") %>%
  select(Admin2:deaths) %>%
  mutate(date = mdy(date)) %>%
  select(-c(Lat, Long_))

# Join `US_cases` and `US_deaths` data to pull in the population that is absent in `US_cases`
US <- US_cases %>%
  full_join(US_deaths)
```

We can take a similar approach for the `global` data set by combining `Province_State` and `Country_Region` into `Combined_key` along with a comma and space as a separator. The result is the `global` data set has very similar data to `US` with the exception of the population data.

To add the population data, we return to the Johns Hopkins website where there is a `*.csv` file containing population data we can add to the `global` data set. After downloading the data, remove the unneeded columns and join the resulting data with the `global` data set.

```r
# `unite()` combines `Province_State` and `Country_Region` with a comma and a space as a separator and 
global <- global %>%
  unite("Combined_Key",
        c(Province_State, Country_Region),
        sep = ", ",
        na.rm = TRUE,
        remove = FALSE)

# Get data from the Johns Hopkins website, remove the columns named `Lat`, `Long_`, `Combined_Key`, `co
uid_lookup_url <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/
uid <- read_csv(uid_lookup_url) %>%
  select(-c(Lat, Long_, Combined_Key, code3, iso2, iso3, Admin2))

global <- global %>%
  left_join(uid, by = c("Province_State", "Country_Region")) %>%
  select(-c(UID, FIPS)) %>%
  select(Province_State, Country_Region, date,
         cases, deaths, Population,
         Combined_Key)
```

Step 3 - Visualizing, Analyzing, and Modeling Data

Analyzing

Now that our data is tidy and transformed, it is ready for visualizing, analyzing and modeling. We start by analyzing data for the US as a whole and for a given state to see what we can we can glean from it.

The analysis below starts taking the US data set and grouping it by `Province_State`, `Country_Region`, and `date` and then call `summarize()` to produce a sum of the cases, deaths, and Population of the counties for each state. Then, `mutate()` creates a new column named `deaths_per_mill` containing a calculated value of the number of deaths per million (deaths * 1000000/ Population) followed by selecting the desired columns.

Similarly, the totals for the `US_by_state` data set is calculated as well.

```r
# Use the `US` data set to group by `Province_State`, `Country_Region`, and `date` and then call `summa
US_by_state <- US %>%
  group_by(Province_State, Country_Region, date) %>%
  summarize(cases = sum(cases), deaths = sum(deaths),
            Population = sum(Population)) %>%
  mutate(deaths_per_mill = deaths *1000000 / Population) %>%
  select(Province_State, Country_Region, date,
         cases, deaths, deaths_per_mill, Population) %>%
  ungroup()

# Similarly, Use the `US_by_state` data set to group by `Province_State`, `Country_Region`, and `date`
US_totals <- US_by_state %>%
  group_by(Country_Region, date) %>%
  summarize(cases = sum(cases), deaths = sum(deaths),
            Population = sum(Population)) %>%
  mutate(deaths_per_mill = deaths *1000000 / Population) %>%
  select(Country_Region, date,
         cases, deaths, deaths_per_mill, Population) %>%
  ungroup()

# Verify what was added to `US_totals` by inspecting the top end of the data set using head()
head(US_totals)
```

```
## # A tibble: 6 x 6
##   Country_Region date       cases deaths deaths_per_mill Population
##   <chr>          <date>     <dbl>  <dbl>           <dbl>      <dbl>
## 1 US             2020-01-22     1      1         0.00300  332875137
## 2 US             2020-01-23     1      1         0.00300  332875137
## 3 US             2020-01-24     2      1         0.00300  332875137
## 4 US             2020-01-25     2      1         0.00300  332875137
## 5 US             2020-01-26     5      1         0.00300  332875137
## 6 US             2020-01-27     5      1         0.00300  332875137
```

```r
# Verify the tail end of the data looks reasonable by calling `head()`
tail(US_totals)
```

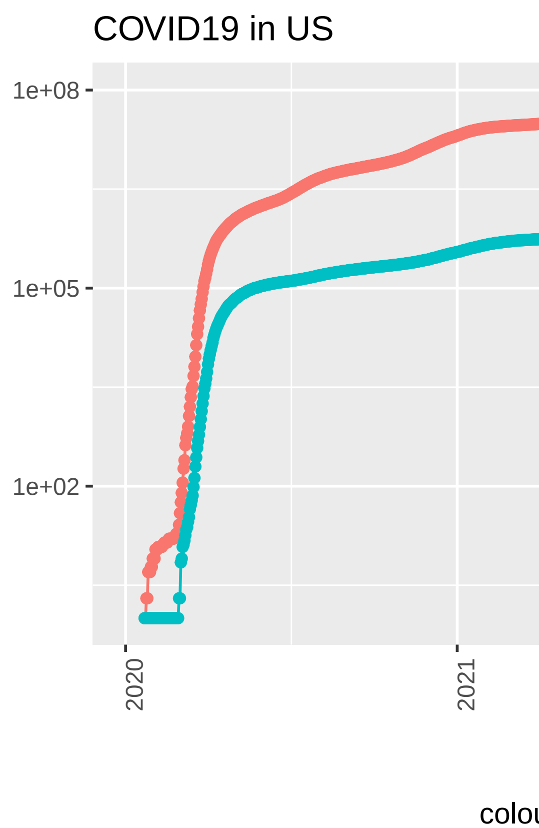
```
## # A tibble: 6 x 6
##   Country_Region date           cases  deaths deaths_per_mill Population
##   <chr>          <date>         <dbl>   <dbl>           <dbl>      <dbl>
## 1 US             2023-03-04 103650837 1122172           3371.  332875137
## 2 US             2023-03-05 103646975 1122134           3371.  332875137
## 3 US             2023-03-06 103655539 1122181           3371.  332875137
## 4 US             2023-03-07 103690910 1122516           3372.  332875137
## 5 US             2023-03-08 103755771 1123246           3374.  332875137
## 6 US             2023-03-09 103802702 1123836           3376.  332875137
```

Visualize

Let's visualize the US_total data set in the steps below.

First, apply a filter to visualize data for positive cases then set up the plot so the date is on the x-axis and the number of cases in on the y-axis. Then, plot a line and points for cases and another line for the number of deaths to the same graph and scale the y variable on a log scale.

```r
# Plot a line and points for cases and another line for the number of deaths to the same graph and scal
US_totals %>%
  filter(cases > 0) %>%
  ggplot(aes(x = date, y = cases)) +
  geom_line(aes(color = "cases")) +
  geom_point(aes(color = "cases")) +
  geom_line(aes(y = deaths, color = "deaths" )) +
  geom_point(aes(y = deaths, color = "deaths")) +
  scale_y_log10() +
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 90)) +
  labs(title = "COVID19 in US", y = NULL)
```



Number of COVID Cases and Number of Deaths in the United States

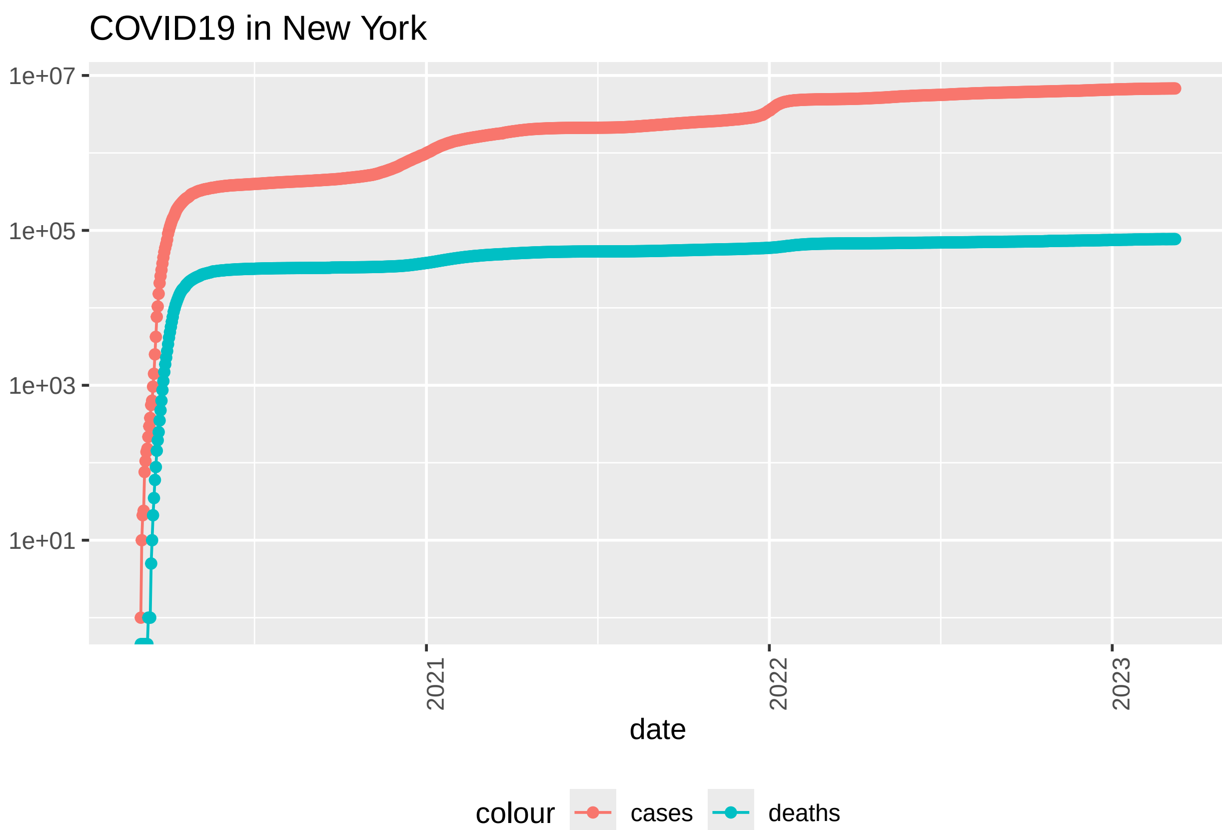
Number of COVID Cases in the State of New York

Similar to the visualization above of the US_totals, let's visualize the number of COVID cases and deaths in the state of New York using the `US_by_state` data set.

The visualizations lead to questions such as, what is the maximum date and are the maximum number of deaths recorded?

```r
# Set the string to `New York` for the `state` variable
state <- "New York"
# Plot a line and points for cases and another line for the number of deaths to the same graph and scal
US_by_state %>%
  filter(Province_State == state) %>%
  filter(cases > 0) %>%
```

```r
  ggplot(aes(x = date, y = cases)) +
  geom_line(aes(color = "cases")) +
  geom_point(aes(color = "cases")) +
  geom_line(aes(y = deaths, color = "deaths")) +
  geom_point(aes(y = deaths, color = "deaths")) +
  scale_y_log10() +
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 90)) +
  labs(title = str_c("COVID19 in ", state), y = NULL)
```




```r
# Have a look at the maximum date for the data set
max(US_totals$date)
```

```
## [1] "2023-03-09"
```

```r
# Have a look at the maximum number of deaths
max(US_totals$deaths)
```

```
## [1] 1123836
```

Further Analysis and Modeling

In the visualization, notice the number of cases seems to level off which raises questions. Does that mean there are no or very few new cases?

The approach is to transform our data again by adding two new columns named `new_cases` and `new_deaths` to the data set.

```r
# Add new column named `new_cases` and `new_deaths` to the data set
US_by_state <- US_by_state %>%
```

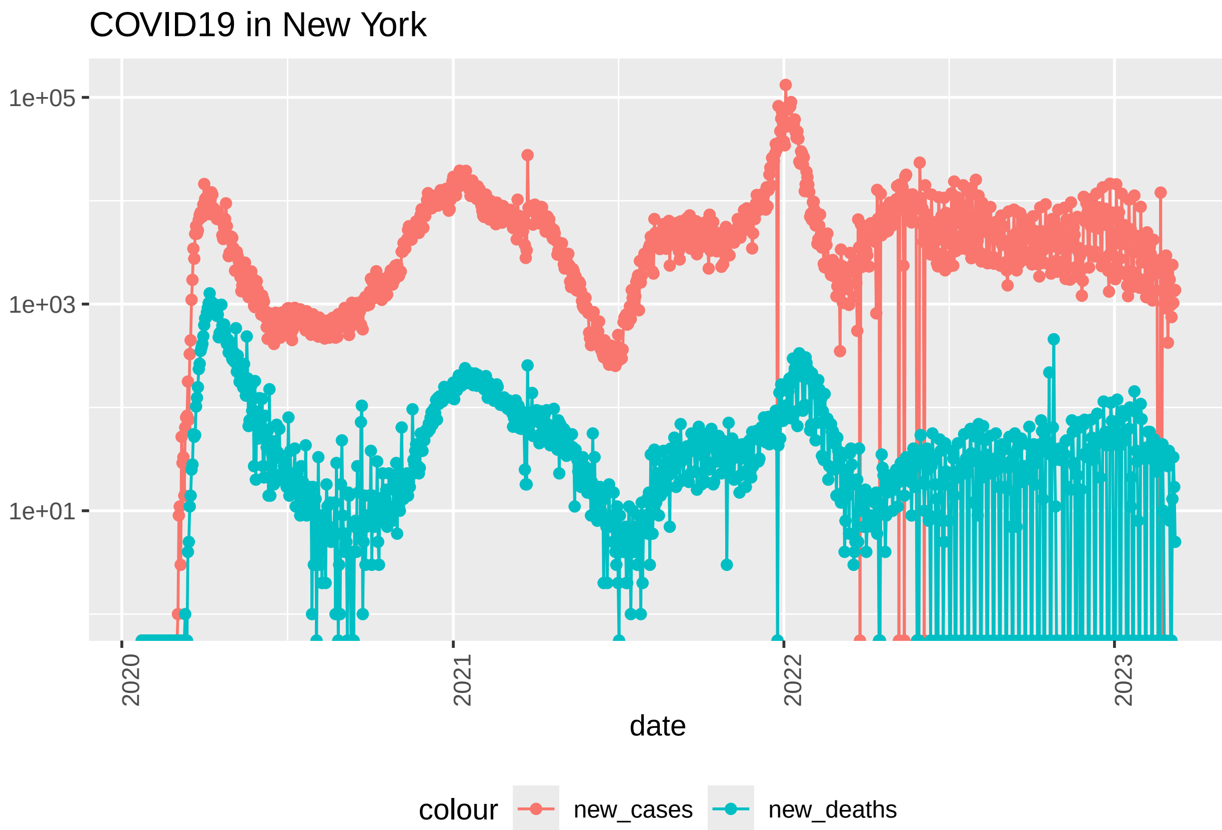
```r
  mutate(new_cases = cases - lag(cases),
         new_deaths = deaths - lag(deaths))
# Add new column named `new_cases` and `new_deaths` to the data set
US_totals <- US_totals %>%
  mutate(new_cases = cases - lag(cases),
         new_deaths = deaths - lag(deaths))

# Inspect the resulting data set
tail(US_totals %>% select(new_cases, new_deaths, everything()))
```

```
## # A tibble: 6 x 8
##   new_cases new_deaths Country_Region date          cases deaths deaths_per_mill
##       <dbl>      <dbl> <chr>          <date>        <dbl>  <dbl>           <dbl>
## 1      2147          7 US             2023-03-04   1.04e8 1.12e6           3371.
## 2     -3862        -38 US             2023-03-05   1.04e8 1.12e6           3371.
## 3      8564         47 US             2023-03-06   1.04e8 1.12e6           3371.
## 4     35371        335 US             2023-03-07   1.04e8 1.12e6           3372.
## 5     64861        730 US             2023-03-08   1.04e8 1.12e6           3374.
## 6     46931        590 US             2023-03-09   1.04e8 1.12e6           3376.
## # i 1 more variable: Population <dbl>
```

Visualize

```r
# Set the string to `New York` for the `state` variable
state <- "New York"
# Visualize the newly analyzed data of the number of new cases and new deaths in the state of New York
US_by_state %>%
  filter(Province_State == state) %>%
  ggplot(aes(x = date, y = new_cases)) +
  geom_line(aes(color = "new_cases")) +
  geom_point(aes(color = "new_cases")) +
  geom_line(aes(y = new_deaths, color = "new_deaths")) +
  geom_point(aes(y = new_deaths, color = "new_deaths")) +
  scale_y_log10() +
  theme(legend.position= "bottom",
        axis.text.x = element_text(angle = 90)) +
  labs(title = str_c("COVID19 in ", state), y = NULL)
```



Additional Analysis

Looking at this new visualization brings up a different set of questions to consider. One of the questions is which states have the highest and lowest numbers of cases and deaths. We analyze and transform the data by looking for the maximum number of cases and deaths by state.

We can also look at the data for the smallest number of cases and deaths per thousand.

Morover, we can predict the number of deaths by applying a linear model and then visualizing the points for both the existing and predicted cases when visualized in the next section.

```
# Transform the data once again by grouping data by state and finding the maximum number of deaths case
US_state_totals <- US_by_state %>%
  group_by(Province_State) %>%
  summarize(deaths = max(deaths), cases = max(cases),
                       population = max(Population),
                       cases_per_thou = 1000* cases / population,
                       deaths_per_thou = 1000* deaths / population) %>%
  filter(cases > 0, population > 0)

# Also look at the smallest number of cases and deaths per 1000
US_state_totals %>%
  slice_min(deaths_per_thou, n= 10)
```

```
## # A tibble: 10 x 6
##    Province_State         deaths  cases population cases_per_thou deaths_per_thou
##    <chr>                   <dbl>  <dbl>      <dbl>          <dbl>           <dbl>
##  1 American Samoa             34 8.32e3      55641           150.           0.611
##  2 Northern Mariana Isl~      41 1.37e4      55144           248.           0.744
```

```
##  3 Virgin Islands           130 2.48e4     107268           231.            1.21
##  4 Hawaii                  1841 3.81e5    1415872           269.            1.30
##  5 Vermont                  929 1.53e5     623989           245.            1.49
##  6 Puerto Rico             5823 1.10e6    3754939           293.            1.55
##  7 Utah                    5298 1.09e6    3205958           340.            1.65
##  8 Alaska                  1486 3.08e5     740995           415.            2.01
##  9 District of Columbia    1432 1.78e5     705749           252.            2.03
## 10 Washington             15683 1.93e6    7614893           253.            2.06
```

```
# Narrow down the data by selecting the columns `deaths_per_thou`, `cases_per_thou`, and `everthing()`
US_state_totals %>%
  slice_min(deaths_per_thou, n= 10) %>%
  select(deaths_per_thou, cases_per_thou, everything())
```

```
## # A tibble: 10 x 6
##    deaths_per_thou cases_per_thou Province_State        deaths  cases population
##              <dbl>          <dbl> <chr>                  <dbl>  <dbl>      <dbl>
##  1           0.611           150. American Samoa            34 8.32e3      55641
##  2           0.744           248. Northern Mariana Isl~     41 1.37e4      55144
##  3           1.21            231. Virgin Islands           130 2.48e4     107268
##  4           1.30            269. Hawaii                  1841 3.81e5    1415872
##  5           1.49            245. Vermont                  929 1.53e5     623989
##  6           1.55            293. Puerto Rico             5823 1.10e6    3754939
##  7           1.65            340. Utah                    5298 1.09e6    3205958
##  8           2.01            415. Alaska                  1486 3.08e5     740995
##  9           2.03            252. District of Columbia    1432 1.78e5     705749
## 10           2.06            253. Washington             15683 1.93e6    7614893
```

```
# Look at the data for the states with the highest cases
US_state_totals %>%
  slice_max(deaths_per_thou, n = 10) %>%
  select(deaths_per_thou, cases_per_thou, everything())
```

```
## # A tibble: 10 x 6
##    deaths_per_thou cases_per_thou Province_State deaths   cases population
##              <dbl>          <dbl> <chr>           <dbl>   <dbl>      <dbl>
##  1            4.55           336. Arizona         33102 2443514    7278717
##  2            4.54           326. Oklahoma        17972 1290929    3956971
##  3            4.49           333. Mississippi     13370  990756    2976149
##  4            4.44           359. West Virginia    7960  642760    1792147
##  5            4.32           320. New Mexico       9061  670929    2096829
##  6            4.31           334. Arkansas        13020 1006883    3017804
##  7            4.29           335. Alabama         21032 1644533    4903185
##  8            4.28           368. Tennessee       29263 2515130    6829174
##  9            4.23           307. Michigan        42205 3064125    9986857
## 10            4.06           385. Kentucky        18130 1718471    4467673
```

```
# Create a linear model where the deaths_per_thou as being a function of the cases_per_thou
mod <- lm(deaths_per_thou ~ cases_per_thou, data = US_state_totals)
# Look at a summary of the linear model
summary(mod)
```

```
##
## Call:
## lm(formula = deaths_per_thou ~ cases_per_thou, data = US_state_totals)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.3352 -0.5978  0.1491  0.6535  1.2086
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -0.36167    0.72480  -0.499     0.62
## cases_per_thou  0.01133    0.00232   4.881 9.76e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8615 on 54 degrees of freedom
## Multiple R-squared:  0.3061, Adjusted R-squared:  0.2933
## F-statistic: 23.82 on 1 and 54 DF,  p-value: 9.763e-06
```

```
# See the smallest cases per thousand
US_state_totals %>% slice_min(cases_per_thou)
```

```
## # A tibble: 1 x 6
##   Province_State deaths cases population cases_per_thou deaths_per_thou
##   <chr>           <dbl> <dbl>      <dbl>          <dbl>           <dbl>
## 1 American Samoa     34  8320      55641           150.           0.611
```

```
# See the largest cases per thousand
US_state_totals %>% slice_max(cases_per_thou)
```

```
## # A tibble: 1 x 6
##   Province_State deaths  cases population cases_per_thou deaths_per_thou
##   <chr>           <dbl>  <dbl>      <dbl>          <dbl>           <dbl>
## 1 Rhode Island     3870 460697    1059361           435.            3.65
```

```
# Model the data containing the US states with predictions and add a new column to reflect the predicti
US_state_totals %>% mutate(pred = predict(mod))
```

```
## # A tibble: 56 x 7
##    Province_State  deaths  cases population cases_per_thou deaths_per_thou  pred
##    <chr>            <dbl>  <dbl>      <dbl>          <dbl>           <dbl> <dbl>
##  1 Alabama          21032 1.64e6    4903185           335.           4.29   3.44
##  2 Alaska            1486 3.08e5     740995           415.           2.01   4.34
##  3 American Samoa      34 8.32e3      55641           150.           0.611  1.33
##  4 Arizona          33102 2.44e6    7278717           336.           4.55   3.44
##  5 Arkansas         13020 1.01e6    3017804           334.           4.31   3.42
##  6 California      101159 1.21e7   39512223           307.           2.56   3.12
##  7 Colorado         14181 1.76e6    5758736           306.           2.46   3.11
##  8 Connecticut      12220 9.77e5    3565287           274.           3.43   2.74
##  9 Delaware          3324 3.31e5     973764           340.           3.41   3.49
## 10 District of Co~   1432 1.78e5     705749           252.           2.03   2.49
## # i 46 more rows
```

```
# Add the predicted value to US_state_totals and store the result in a variable named `US_tot_w_pred`
US_tot_w_pred <- US_state_totals %>% mutate(pred = predict(mod))
US_tot_w_pred
```

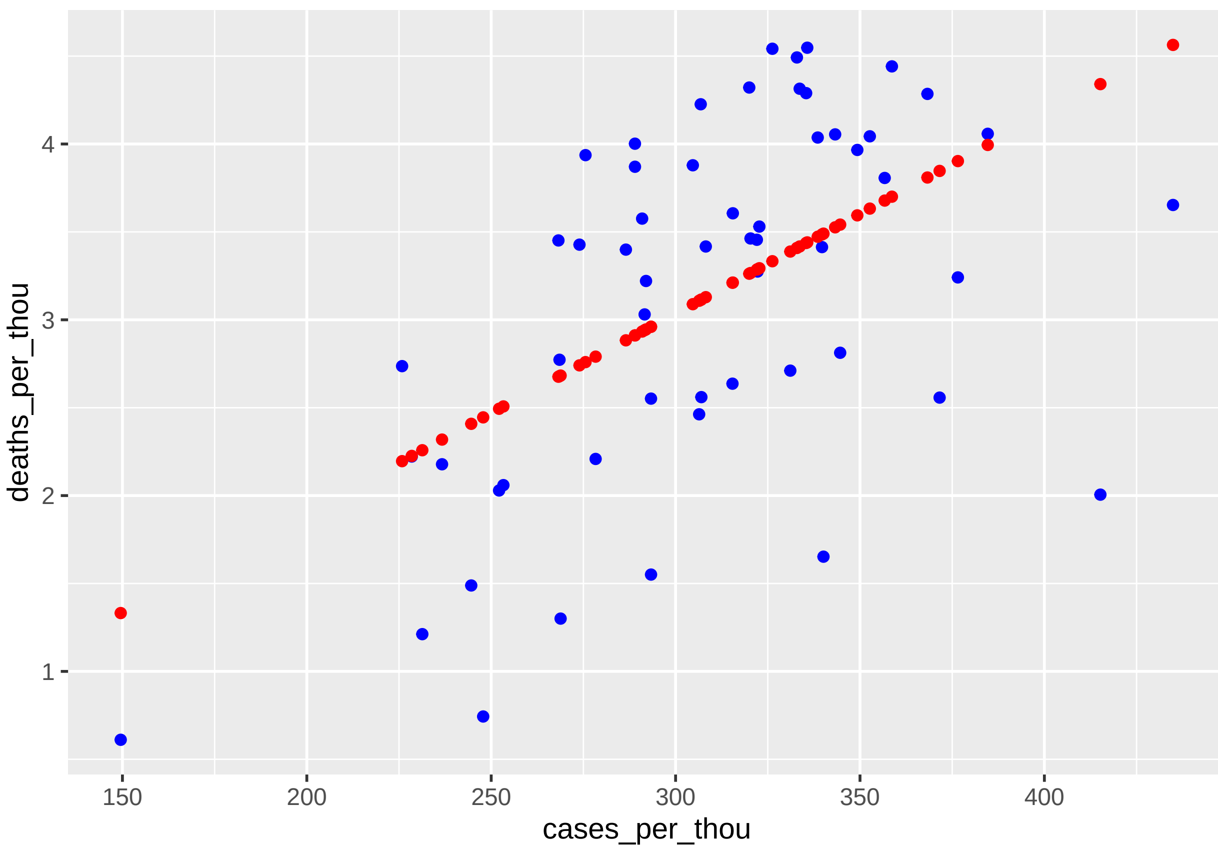
```
## # A tibble: 56 x 7
##    Province_State  deaths  cases population cases_per_thou deaths_per_thou  pred
##    <chr>            <dbl>  <dbl>      <dbl>          <dbl>           <dbl> <dbl>
##  1 Alabama          21032 1.64e6    4903185           335.           4.29   3.44
```

```
## 2 Alaska 1486 3.08e5 740995 415. 2.01 4.34
## 3 American Samoa 34 8.32e3 55641 150. 0.611 1.33
## 4 Arizona 33102 2.44e6 7278717 336. 4.55 3.44
## 5 Arkansas 13020 1.01e6 3017804 334. 4.31 3.42
## 6 California 101159 1.21e7 39512223 307. 2.56 3.12
## 7 Colorado 14181 1.76e6 5758736 306. 2.46 3.11
## 8 Connecticut 12220 9.77e5 3565287 274. 3.43 2.74
## 9 Delaware 3324 3.31e5 973764 340. 3.41 3.49
## 10 District of Co~ 1432 1.78e5 705749 252. 2.03 2.49
## # i 46 more rows
```

Visualize the new analysis so we see the existing data and the predicted outcomes

Using the existing data and predicted results from earlier, we can visualize the analysis on a chart with the existing cases with the blue dots and the predicted cases resulting from the linear model with the red dots.

```r
# Visualize the existing and predicted data by plotting the existing data in blue and predicted data in
US_tot_w_pred %>% ggplot() +
  geom_point(aes(x = cases_per_thou, y = deaths_per_thou), color = "blue") +
  geom_point(aes(x = cases_per_thou, y = pred), color = "red")
```



Bias Identification

The analysis was fairly impartial but there were outliers in the data that were not taken into consideration. For example, we saw outliers in the plot containing the linear model showing the predictions. However, the outlying data were not considered for further analysis. This is not to say it was not identified and discussed verbally. Instead, it is a source of bias simply because it was not addressed.

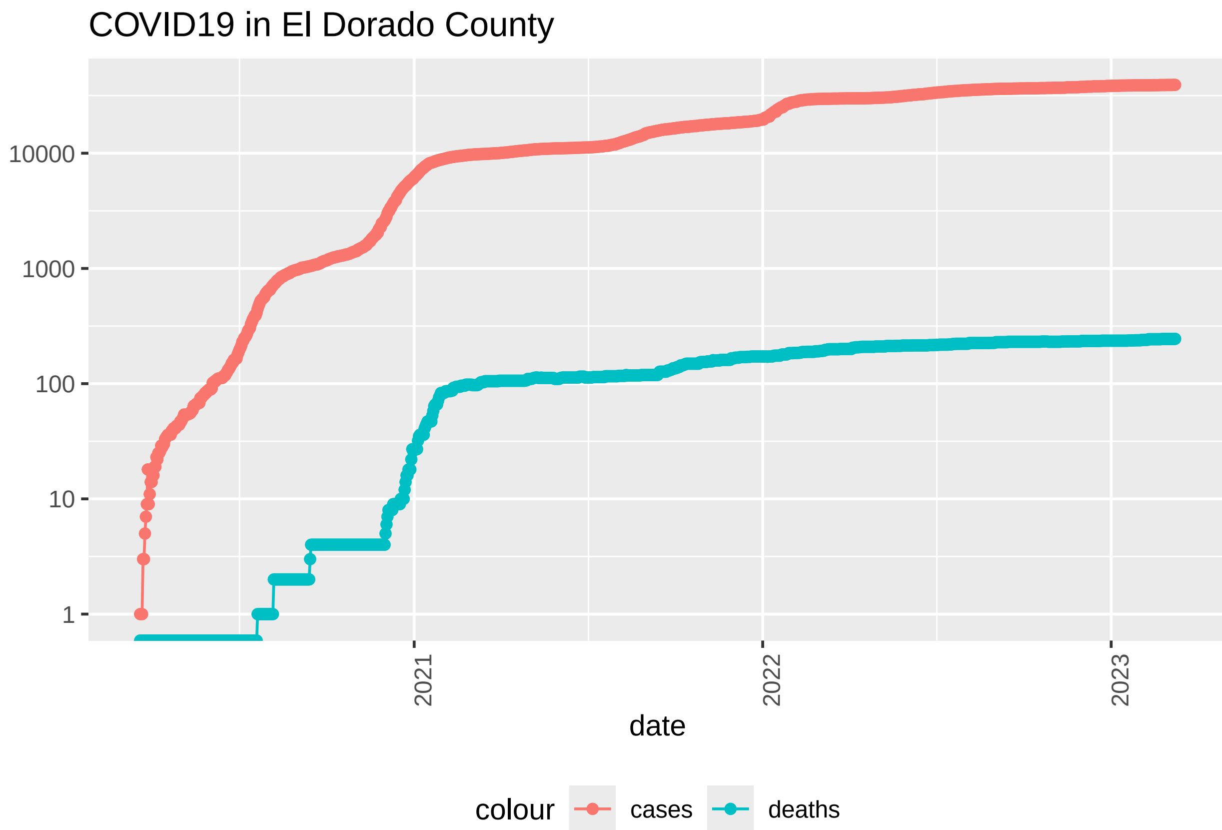
My Additional Analysis

As part of the project, I present my analysis and at least two visualizations. I live in El Dorado County, California and I would like to know more about the cases and deaths where I live. Much like the original analysis performed on the state of New York, I see that the number of cases and deaths flattening so it raises questions about why that might be.

First, I organize my data by county and visualize the number of cases vs the number of deaths.

```
# Use the `US` data set to group by `Admin2` which is to say by county then `Province_State`, `Country_
US_by_county <- US %>%
  group_by(Admin2, Province_State, Country_Region, date) %>%
  summarize(cases = sum(cases), deaths = sum(deaths),
            Population = sum(Population)) %>%
  select(Admin2, Province_State,
         Country_Region, date, cases, deaths, Population) %>%
  ungroup()
```

```
# Set the string to `New York` for the `state` variable
my_state <- "California"
my_county <- "El Dorado"
# Visualize the newly analyzed data of the number of new cases and new deaths in the state of New York
US_by_county %>%
  filter(Admin2 == my_county, Province_State == my_state, cases > 0) %>%
  ggplot(aes(x = date, y = cases)) +
  geom_line(aes(color = "cases")) +
  geom_point(aes(color = "cases")) +
  geom_line(aes(y = deaths, color = "deaths")) +
  geom_point(aes(y = deaths, color = "deaths")) +
  scale_y_log10() +
  theme(legend.position= "bottom",
        axis.text.x = element_text(angle = 90)) +
  labs(title = str_c("COVID19 in ", my_county, " County"), y = NULL)
```

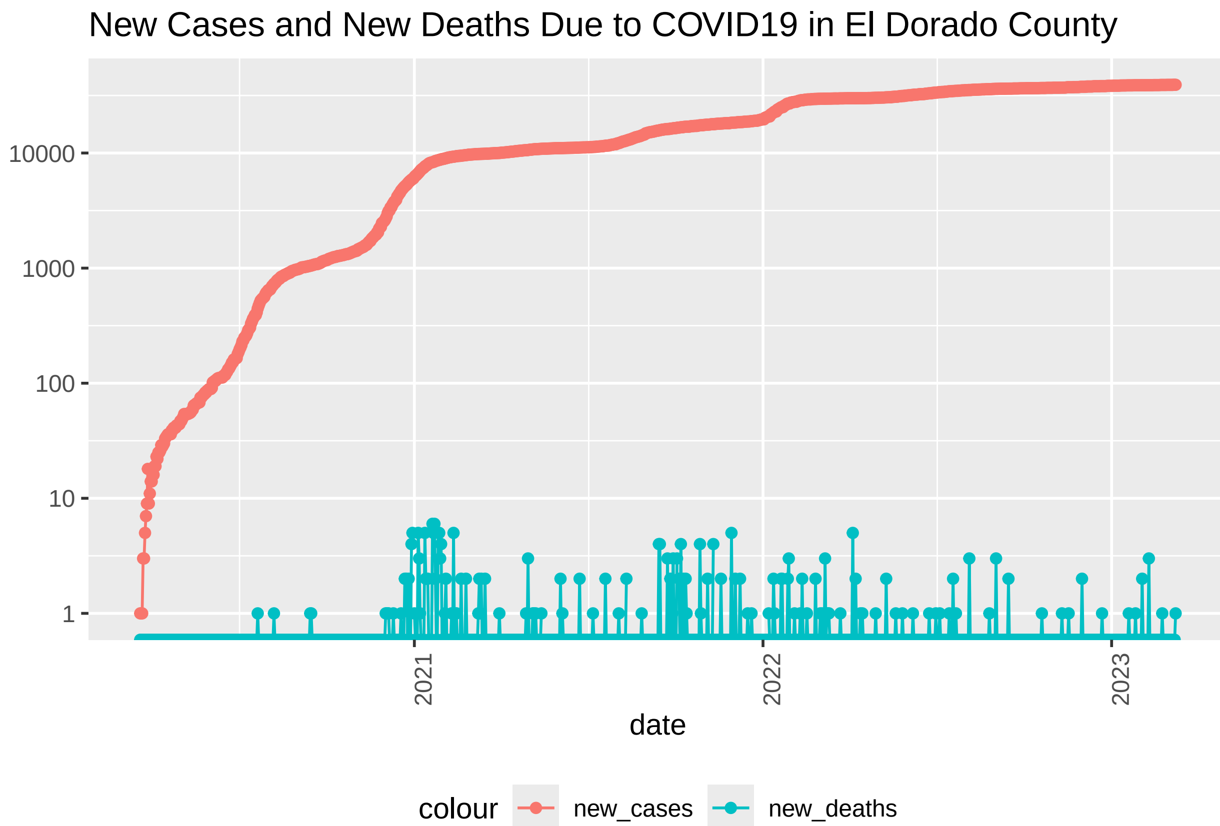


My Further Analysis

We see the flattening of the curve so I would like to know more about the new cases and deaths I create new columns for `new_cases` and `new_deaths` and add them to the `US_by_county` variable. Then I visualize the results by plotting `new_cases` and `new_deaths` and see that new deaths don't seem as leveled off and severe as the first visualization suggests.

```r
# Add new column named `new_cases` and `new_deaths` to the data set
US_by_county <- US_by_county %>%
  mutate(new_cases = cases - lag(cases),
         new_deaths = deaths - lag(deaths))
# Inspect the resulting data set
#tail(US_by_county %>% select(new_cases, new_deaths, everything()))
```

```r
# Visualize the newly analyzed data of the number of new cases and new deaths in the state of New York
US_by_county %>%
  filter(Admin2 == my_county, Province_State == my_state, cases > 0) %>%
  ggplot(aes(x = date, y = cases)) +
  geom_line(aes(color = "new_cases")) +
  geom_point(aes(color = "new_cases")) +
  geom_line(aes(y = new_deaths, color = "new_deaths")) +
  geom_point(aes(y = new_deaths, color = "new_deaths")) +
  scale_y_log10() +
  theme(legend.position= "bottom",
        axis.text.x = element_text(angle = 90)) +
  labs(title = str_c("New Cases and New Deaths Due to COVID19 in ", my_county, " County"), y = NULL)
```



My Implicit Bias

Analyzing and visualizing data for my state and county has implicit bias since I live here. While my curiosity about my county answers questions I may not have had before, it lacks data from surrounding counties or adjacent states that may contribute to the data within my county. As such, to mitigate my bias from the analysis, I could include data from adjacent states and counties to form more neutral conclusions.

Conclusion

This report analyzes and visualizes 4 data sets in iterations of the data science process in an effort to better understand cases and deaths due to the COVID 19 virus. The different steps in the data science process are performed and the outcomes presented for consideration including repeating steps as needed in effort to answer questions that surface data is visualized in a graph or when organized and revealed in the summary of tables. Questions are addressed by repeating the data gathering, analyzing, modeling, and visualizing processes to better understand the data.

We must also consider biases with our approach to our analysis. Biases are identified and mitigation is discussed. For example, towards the end of this project report, my willingness to narrow down the data set to the state and county I live in is a form of bias. Doing so, removes several considerations outside of my county and state that could potentially skew my analysis of my data. To mitigate the bias, I could be open to including surrounding counties or the entire state to be more inclusive of factors surrounding my county.