Covid 19 Report

L. Baldridge

2025-05-29

1. Introduction to the Dataset

The data that was used for this analysis were imported from the Johns Hopkins University Center for Systems Science and Engineering's github repository (`https://github.com/CSSEGISandData/COVID-19`). The department themselves took on the task of aggregating information from an extensive and diverse collection of online sources (all of which are listed on the main readme document on the main page of the repository) and subsequently compiling the contents that would make up the entire database, which they have left available for anyone to access.

The specific csv files that are imported and analyzed in this report contain information on the number of new COVID deaths and cases per day for individual countries (for the global dataset) and individual counties within each US state (for the United States dataset) from January 22nd, 2020 to March 9th, 2023.

2. Import Libraries and Dataset

First we import the necessary libraries that we will need in order to clean, tidy, and format our data for further analysis. Afterwards we import the csv files directly from the Github repository itself.

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.4.3

## Warning: package 'ggplot2' was built under R version 4.4.3

## Warning: package 'tibble' was built under R version 4.4.3

## Warning: package 'tidyr' was built under R version 4.4.3

## Warning: package 'readr' was built under R version 4.4.3

## Warning: package 'dplyr' was built under R version 4.4.3

## Warning: package 'forcats' was built under R version 4.4.3

## Warning: package 'lubridate' was built under R version 4.4.3
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(glue)
```

```
url_in <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covi

file_names <- c("time_series_covid19_confirmed_global.csv",
                "time_series_covid19_deaths_global.csv",
                "time_series_covid19_confirmed_US.csv",
                "time_series_covid19_deaths_US.csv")

urls <- stringr::str_c(url_in, file_names)

gl_cases <- readr::read_csv(urls[1])
```

```
## 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.
```

```
gl_deaths <- readr::read_csv(urls[2])
```

```
## 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.
```

```
u_cases <- readr::read_csv(urls[3])
```

```
## 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.
```

```
u_deaths <- readr::read_csv(urls[4])
```

```
## 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.
```

3. Cleaning and Tidying Data

3.1 Pivot Data

Now we begin to clean and tidy our data, and the first task is to create functions that will pivot our dataset in order to turn dates (which initially are columns in the original csv files) to rows. As for the other columns, we will filter ones that we do not need, but keep the orientation for the ones that we want to keep. After creating the functions, we apply them to the csv files we imported to create new dataframes.

```
pivot_global = function(x, value_type) {
  pivoted <- x %>%
    tidyr::pivot_longer(
      cols = -c(`Province/State`, `Country/Region`, Lat, Long),
      names_to = "date",
      values_to = value_type
    ) %>%
    dplyr::select(-c(Lat, Long))
  return(pivoted)

}

pivot_us = function(x, value_type) {
  if (value_type == 'cases'){
    end_col = 'Combined_Key'
  }
  else{
    end_col = 'Population'
  }
  pivoted <- x %>%
  tidyr::pivot_longer(
    cols = -(UID:end_col),
    names_to = "date",
    values_to = value_type
  ) %>%
  dplyr::select(Admin2:value_type) %>%
  dplyr::mutate(date = lubridate::mdy(date)) %>%
```

```
  dplyr::select(-c(Lat, Long_))
  return(pivoted)
}

global_cases = pivot_global(gl_cases, 'cases')
global_deaths = pivot_global(gl_deaths, 'deaths')
us_cases = pivot_us(u_cases, 'cases')
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use ‘all_of()‘ or ‘any_of()‘ instead.
##   # Was:
##   data %>% select(end_col)
##
##   # Now:
##   data %>% select(all_of(end_col))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use ‘all_of()‘ or ‘any_of()‘ instead.
##   # Was:
##   data %>% select(value_type)
##
##   # Now:
##   data %>% select(all_of(value_type))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
us_deaths = pivot_us(u_deaths, 'deaths')
```

3.2 Joining Dataframes

Now that we have pivoted and formatted both the global and US datasets, we now join the deaths and cases dataframes with respect to their scope of information in order to get a unified dataframe.

```
global_data <- global_cases %>%
    dplyr::full_join(global_deaths) %>%
    dplyr::rename(
    Country_Region = `Country/Region`,
    Province_State = `Province/State`
    ) %>%
    dplyr::mutate(date = lubridate::mdy(date))
```

```
## Joining with ‘by = join_by(‘Province/State‘, ‘Country/Region‘, date)‘
```

```
us_data <- us_cases %>%
    dplyr::full_join(us_deaths)
```

```
## Joining with `by = join_by(Admin2, Province_State, Country_Region,
## Combined_Key, date)`
```

3.4 Verify Dataframes

Our last step in cleaning and tidying data is to verify the conntents by printing out the amount of missing cells and getting a summary of the entire dataframe. We see below that the global dataset has mostly missing information for the Province_State column while Admin2 for the US dataset has some missing information. This is fine for the type of analysis that we will be doing, which is primarily concerned with the aggregation of all the data, and can be seen further in this report.

```
print(colSums(is.na(global_data)))
```

```
## Province_State Country_Region           date          cases         deaths
##         226314              0              0              0              0
```

```
summary(global_data)
```

```
##  Province_State     Country_Region          date                cases
##  Length:330327      Length:330327      Min.   :2020-01-22   Min.   :        0
##  Class :character   Class :character   1st Qu.:2020-11-02   1st Qu.:      680
##  Mode  :character   Mode  :character   Median :2021-08-15   Median :    14429
##                                        Mean   :2021-08-15   Mean   :   959384
##                                        3rd Qu.:2022-05-28   3rd Qu.:   228517
##                                        Max.   :2023-03-09   Max.   :103802702
##      deaths
##  Min.   :      0
##  1st Qu.:      3
##  Median :    150
##  Mean   :  13380
##  3rd Qu.:   3032
##  Max.   :1123836
```

```
print(colSums(is.na(us_data)))
```

```
##         Admin2 Province_State Country_Region   Combined_Key           date
##           6858              0              0              0              0
##          cases     Population         deaths
##              0              0              0
```

```
summary(us_data)
```

```
##     Admin2          Province_State     Country_Region     Combined_Key
##  Length:3819906     Length:3819906     Length:3819906     Length:3819906
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

```
##
##
##
##       date                cases             Population           deaths
##  Min.   :2020-01-22   Min.   : -3073   Min.   :       0   Min.   : -82.0
##  1st Qu.:2020-11-02   1st Qu.:   330   1st Qu.:    9917   1st Qu.:   4.0
##  Median :2021-08-15   Median :  2272   Median :   24892   Median :  37.0
##  Mean   :2021-08-15   Mean   : 14088   Mean   :   99604   Mean   : 186.9
##  3rd Qu.:2022-05-28   3rd Qu.:  8159   3rd Qu.:   64979   3rd Qu.: 122.0
##  Max.   :2023-03-09   Max.   :3710586   Max.   :10039107   Max.   :35545.0
```

4. Transform Datasets in Preparation for Analysis

We move on to transforming our datasets into formats that will be useful for the type of analysis that we will be doing. We first do so with the US dataset, which will be used for the two main visualization and analysis segment.

4.1 Transform US Dataset

Here, we define a function that returns the total population by looking at all the distinct FIPS values from the original US deaths CVS file. We use FIPS as it is the main unique identifier for each county, and as we have the population data broken down by county in the unmodified US deaths table.

Next, we define a function that calculates and returns the monthly aggregates of cases and deaths. To do so, we must first find the daily totals by county, and then add up each day to find the monthly totals. We also include a line that appends the total population data from the previous function.

Afterwards we apply these functions to the pivoted US dataset and print the head in order to make sure that the output looks correct.

```
calculate_total_population <- function(x){
  total_population <- x %>%
    distinct(FIPS, .keep_all = TRUE) %>%
    summarize(total_pop = sum(Population)) %>%
    pull(total_pop)

  return(total_population)

}

create_monthly_df <- function(x, pop_value){
  daily_summary <- x %>%
    group_by(date) %>%
    summarize(
      daily_cumulative_cases = sum(cases, na.rm = TRUE),
      daily_cumulative_deaths = sum(deaths, na.rm = TRUE)
    ) %>%
    ungroup() %>%
    arrange(date)

  monthly_data <- daily_summary %>%
    mutate(year_month = floor_date(date, "month")) %>%
```

```
    group_by(year_month) %>%
    summarize(
      cumulative_cases_month_end = last(daily_cumulative_cases),
      cumulative_deaths_month_end = last(daily_cumulative_deaths),
      total_population = pop_value
    ) %>%
    ungroup() %>%
    arrange(year_month)

  return(monthly_data)
}

total_US_population_value <- calculate_total_population(u_deaths)

monthly_us_data = create_monthly_df(us_data,total_US_population_value)
head(monthly_us_data)
```

```
## # A tibble: 6 x 4
##   year_month cumulative_cases_month_end cumulative_deaths_mon~1 total_population
##   <date>                          <dbl>                   <dbl>            <dbl>
## 1 2020-01-01                          8                       1        332386194
## 2 2020-02-01                         26                       2        332386194
## 3 2020-03-01                     192079                    5359        332386194
## 4 2020-04-01                    1081150                   66636        332386194
## 5 2020-05-01                    1791547                  107857        332386194
## 6 2020-06-01                    2648797                  127432        332386194
## # i abbreviated name: 1: cumulative_deaths_month_end
```

4.2 Transform Global Dataset

We apply a similar transformation to our global dataset but the main difference here is that we do not use functions this time around, and that the amount of new monthly cases is calculated right away.

```
global_daily_cumulative <- global_data %>%
  group_by(date) %>%
  summarize(
    global_cumulative_cases_daily = sum(cases, na.rm = TRUE)
  ) %>%
  ungroup() %>%
  arrange(date)

global_daily_new <- global_daily_cumulative %>%
  mutate(
    global_new_cases_daily = global_cumulative_cases_daily - lag(global_cumulative_cases_daily, default
  ) %>%
  mutate(global_new_cases_daily = pmax(0, global_new_cases_daily))

global_monthly_new_cases_df <- global_daily_new %>%
  mutate(year_month = floor_date(date, "month")) %>%
  group_by(year_month) %>%
  summarize(global_new_monthly_cases = sum(global_new_cases_daily, na.rm = TRUE)) %>%
  ungroup() %>%
```

```
  arrange(year_month)

tail(global_monthly_new_cases_df)
```

```
## # A tibble: 6 x 2
##   year_month global_new_monthly_cases
##   <date>                         <dbl>
## 1 2022-10-01                  12795215
## 2 2022-11-01                  12388536
## 3 2022-12-01                  17219669
## 4 2023-01-01                  10270797
## 5 2023-02-01                   4587649
## 6 2023-03-01                   1247911
```

5. Visualizations and Analysis (New Monthly Cases vs Case Fatality Rate)

After the main preparations are finished, we can move on to the visualization and analysis segment of our report which will be focused on the US dataset.

There are two main statistics that we are concered with; one of which is the Rate of New Infections per Month, which as the name implies, is concerned with the changes regarding how many individuals are newly infected each month based on the amount of the population that is still susceptible to catching Covid. We make the assumption that once an individual is infected, that they cannot be infected again to simplify the calculation and due to the fact that we have no information regarding how many of the new cases are technically tied to individuals who have once recovered but are one again sick. To calculate this, we first have to create a column that is the lagged value of the new cases per month column and subtract this value against the total population in order to find the amount of the population that is susceptible per month. Finally, the rate of new infections per month is calculated by dividing the amount of new cases per month by the total amount of people that are susceptible for that month.

The other statistic that we calculate is the Case Fatality Rate (dubbed as CFR moving forward) per month. The CFR is a standard epidimiology statistic that "estimates [the] proportion of deaths among identified confirmed cases", as defined by the World Health Organization in this Covid-related article (`https://www.who.int/news-room/commentaries/detail/estimating-mortality-from-covid-19`). In order to relate this to our rate of new monthly cases statistic, we make a slight modification in how it is calculated. First, we once again begin by creating a lagged column using new deaths per month as the basis. The modifications referred to come in when we divide this amount of lagged new deaths each month by the amount of new cases per month. The outcome then represents the monthly CFR rate for this dataset.

5.1 US Dataset

5.1.1 Calculating Necessary Statistics

Below shows the function that is used in order to calculate the statistics referenced in the previous Section 5, again it shows how to create the lagged cases and lagged deaths columns, calculate the susceptible population statistic by subtracting the amount of cumulative cases for the past month from the total population, calculate the amount of new cases and deaths per month, and finally calculate the rates themselves (rate of new infections monthly and cfr monthly). To clean up our table, we filter out the columns that are not needed for the final result. Lastly we display the tail instead of the head of this dataframe as this allows us to better verify that the rates and aggregates were calculated correctly.

```r
calculate_monthly_df_statistics <- function(x){
  transformed_monthly_data <- x %>%
    mutate(
      cumulative_cases_previous_month_end = lag(cumulative_cases_month_end, default = 0),
      cumulative_deaths_previous_month_end = lag(cumulative_deaths_month_end, default = 0),
      susceptible_population_start_month = total_population - cumulative_cases_previous_month_end,
      new_cases_this_month = cumulative_cases_month_end - cumulative_cases_previous_month_end,
      rate_new_infections_monthly = ifelse(susceptible_population_start_month > 0,
                                           new_cases_this_month / susceptible_population_start_month,
                                           NA_real_),
      new_deaths_this_month = cumulative_deaths_month_end - cumulative_deaths_previous_month_end,
      cfr_monthly_period = ifelse(new_cases_this_month > 0,
                                  new_deaths_this_month / new_cases_this_month,
                                  NA_real_)
    )  %>%
    select(-c('cumulative_cases_previous_month_end', 'cumulative_deaths_previous_month_end', 'total_popu


  transformed_monthly_data <- transformed_monthly_data %>%
    mutate(
      cfr_cumulative_monthly = ifelse(cumulative_cases_month_end > 0,
                                  cumulative_deaths_month_end / cumulative_cases_month_end,
                                  NA_real_)
    )

  return(transformed_monthly_data)
}

transformed_monthly_us_data <- calculate_monthly_df_statistics(monthly_us_data)
tail(transformed_monthly_us_data)
```

```
## # A tibble: 6 x 9
##   year_month cumulative_cases_month_end cumulative_deaths_month_end
##   <date>                          <dbl>                       <dbl>
## 1 2022-10-01                   97491846                     1070376
## 2 2022-11-01                   98813400                     1080466
## 3 2022-12-01                  100765248                     1092764
## 4 2023-01-01                  102362900                     1108688
## 5 2023-02-01                  103443455                     1119917
## 6 2023-03-01                  103802702                     1123836
## # i 6 more variables: susceptible_population_start_month <dbl>,
## #   new_cases_this_month <dbl>, rate_new_infections_monthly <dbl>,
## #   new_deaths_this_month <dbl>, cfr_monthly_period <dbl>,
## #   cfr_cumulative_monthly <dbl>
```

5.1.2 Calculating Scaling Coefficient

As seen from the table above, rate_new_infections monthly and cfr_monthly_period are on different magnitudes, so in order for us to get a better visualization the overall trends during the pandemic, we calculate a coefficient that would display both rates in similar scales. This is done since we are specifically interested in how how the increasing and decreasing segments of each rate's cycle affect the other, and are not necessarily concerned with the absolute values of the rates themselves.

```r
calculate_coefficient <- function(x){
  max_infection_rate <- max(x$rate_new_infections_monthly, na.rm = TRUE)
  max_cfr <- max(x$cfr_monthly_period[-1], na.rm = TRUE)
  coeff <- max_cfr / max_infection_rate
  print(glue("Using scaling coefficient (coeff):{round(coeff,5)}" ))

  return(coeff)
}

us_coeff = calculate_coefficient(transformed_monthly_us_data)
```

```
## Using scaling coefficient (coeff):0.9404
```

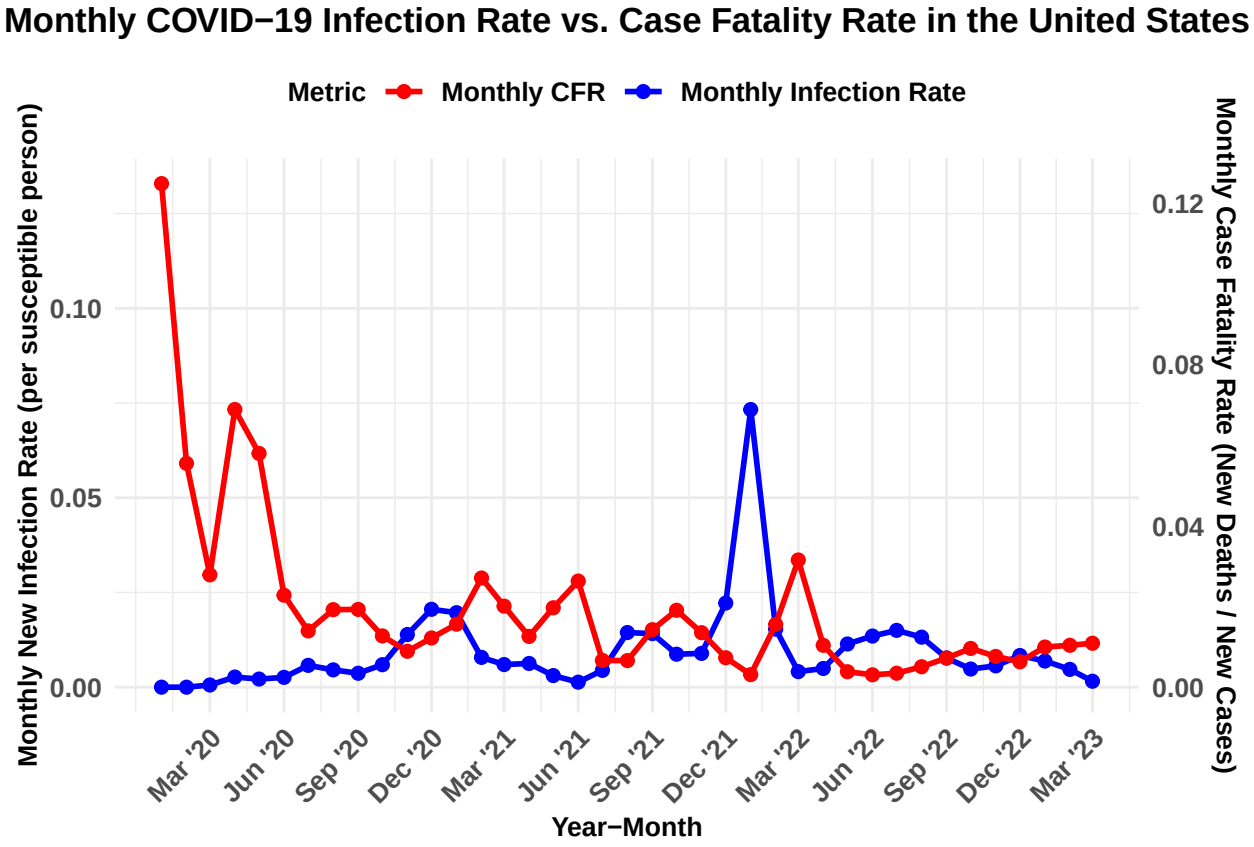
5.1.3 Creating the Graph of Trends

Now that we have all the necessary information to properly display our graph, we can turn towards creating the visual itself. First we set some options for the width and the height for better readability, then we create a list of specific quarterly months (March, June, September, and December) for all of the years which will be used to create even x-axis ticks on the graph itself. Finally we define a function that takes in the dataframe for visualization, the scaling coefficient needed, and string that represents the scope of it's data and outputs a dual-axis line graph that displays the trends we aim to analyze.

We then input the necessary parameters to this function to print the graph itself.

```r
options(repr.plot.width = 30, repr.plot.height = 20)

quarter_breaks <- transformed_monthly_us_data %>%
  filter(month(year_month) %in% c(3, 6, 9, 12)) %>%
  pull(year_month) %>%
  unique() %>%
  sort()

generate_plot <- function(x, coeff, location){
  dual_axis_plot <- ggplot(x, aes(x = year_month)) +
    geom_line(aes(y = rate_new_infections_monthly, color = "Monthly Infection Rate"), linewidth = 1) +
    geom_point(aes(y = rate_new_infections_monthly, color = "Monthly Infection Rate"), size = 2) +
    geom_line(aes(y = cfr_monthly_period / coeff, color = "Monthly CFR"), linewidth = 1) +
    geom_point(aes(y = cfr_monthly_period / coeff, color = "Monthly CFR"), size = 2) +
    scale_y_continuous(
      name = "Monthly New Infection Rate (per susceptible person)",
      sec.axis = sec_axis(
        ~ . * coeff,
        name = "Monthly Case Fatality Rate (New Deaths / New Cases)"
      )
    ) +
    scale_x_date(
      name = "Year-Month",
      breaks = quarter_breaks,
      date_labels = "%b '%y"
    ) +
    scale_color_manual(values = c("Monthly Infection Rate" = "blue", "Monthly CFR" = "red")) +
    labs(
      title = glue("Monthly COVID-19 Infection Rate vs. Case Fatality Rate in the {location}"),
```

```
      # x = "Year-Month", # X-axis label is now in scale_x_date
      color = "Metric"
    ) +
    theme_minimal() +
    theme(
      legend.position = "top",
      axis.title.x = element_text(size = 10, face = "bold"),
      axis.title.y = element_text(size = 10, face = "bold"),
      axis.text.x = element_text(size = 10, face = "bold", angle = 45, hjust = 1),
      axis.text.y = element_text(size = 10, face = "bold"),
      plot.title = element_text(size = 13, hjust = 0.5, face = 'bold'),
      legend.title = element_text(size = 10, face = "bold"),
      legend.text = element_text(size = 10, face = "bold")
    )

  print(dual_axis_plot)
}

generate_plot(transformed_monthly_us_data, us_coeff, "United States")
```




```
head(transformed_monthly_us_data)
```

5.1.4 Analysis on Rate of New Infections Monthly vs CFR Monthly for entire US

```
## # A tibble: 6 x 9
##   year_month cumulative_cases_month_end cumulative_deaths_month_end
##   <date>                          <dbl>                       <dbl>
## 1 2020-01-01                          8                           1
## 2 2020-02-01                         26                           2
## 3 2020-03-01                     192079                        5359
## 4 2020-04-01                    1081150                       66636
## 5 2020-05-01                    1791547                      107857
## 6 2020-06-01                    2648797                      127432
## # i 6 more variables: susceptible_population_start_month <dbl>,
## #   new_cases_this_month <dbl>, rate_new_infections_monthly <dbl>,
## #   new_deaths_this_month <dbl>, cfr_monthly_period <dbl>,
## #   cfr_cumulative_monthly <dbl>
```

The first thing to note is that while the CFR might look alarmingly high during the first few months, statistically it makes sense for this to be the case because not only has Covid only infected a small percentage of the US population at this point, the procedural and consistent practice of detecting and reporting cases was also just at its infancy during this period.

As such, newly reported deaths factored in heavily in the CFR calculation early on. For example, as seen in the table above, January 2020 had 1 new death and 8 new cases reported for a CFR of 1/8 (12.5%) while February had one more new death recorded and only 18 new cases for a CFR of 1/18 (5.5%). The first significant spike that would have caused some concern would have been the CFR number for April 2020, wherein the US experienced 11.4 times more new deaths than the previous month (5,357 vs 61,227) while new cases only saw about a 4.6x increase. Thankfully, the CFR did subsequently decline and eventually stabilized to a more manageable rate.

By the end of 2020, Covid has fully entrenched itself in the United States, infecting more than 20 million individuals. It is also at this point that we see a relationship take shape between the monthly new infection rate and the CFR. From this point on, we can see a trend develop where every new short term peak in new monthly infections inevitably leads to a spike in CFR soon after. Furthermore, it's interesting to note that the ratio of new deaths relative to new cases seem to mostly reach its peak on the exact period when new infections are declining rapidly. As such, the two rates end up forming a consistently cyclical but contrasting dynamic of accumulation and dispersion and provides insight on how populations navigate waves of infection with some ultimately succumbing to the virus.

The one final thing I'd like to point out is that while we do see this cyclical nature for both new cases and deaths, the manner in which it unfolds does seem to be slightly different. For new monthly infections, the peak and the troughs come about in a more gradual manner (we can see the connecting lines forming a more rounded curve) while the CFR cycle displays much sharper points of inflection on both upward and downward trajectory.

5.2 Pacific Northwest Dataset

5.2.1 Calculating Necessary Statistics

The second visualization and analysis follows a similar procedure to the steps taken above with a difference in scope. Here we put our focus on looking specifically at how the trends unfolded for the Pacific Northwest region, defined in code as the states of Oregon and Washington below. We use this list to filter the main us_data and original us_deaths csv in order to get the Pacific Northwest cases, deaths, and population numbers. We then reuse all of the functions that were previously defined to create the dataframes for the monthly aggregates and calculated rate statistics.

```r
pnw_states <- c("Washington", "Oregon")

pnw_data <- us_data %>%
  filter(Province_State %in% pnw_states)

pnw_pop_data <- u_deaths  %>%
    filter(Province_State %in% pnw_states)

total_pnw_population_value <- calculate_total_population(pnw_pop_data)

monthly_pnw_data <- create_monthly_df(pnw_data, total_pnw_population_value)
transformed_monthly_pnw_data <- calculate_monthly_df_statistics(monthly_pnw_data)
```

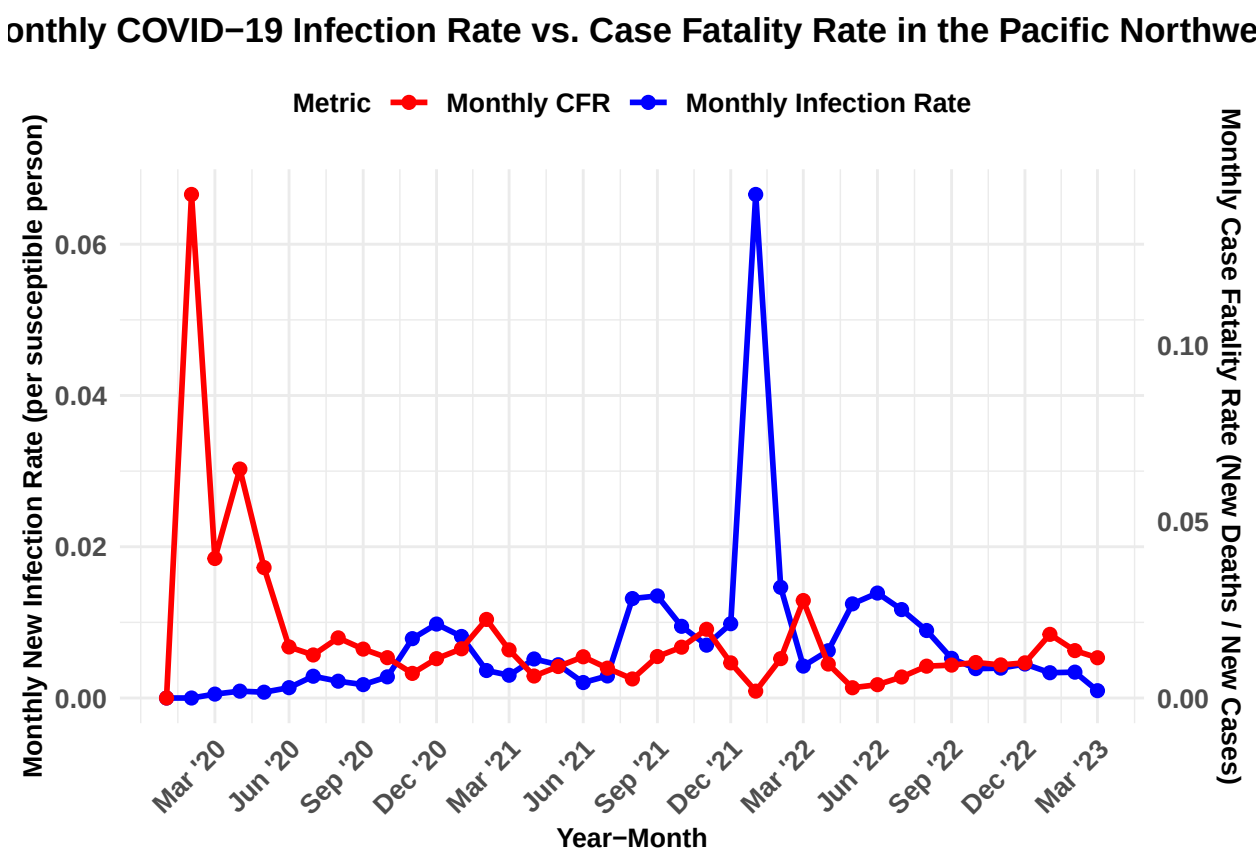
5.2.2 Calculating Scaling Coefficient and Generating Graph

We then also reuse the functions for finding the coefficient needed to scale the two rates and generating the plot, and the outcome can be seen below.

```r
options(repr.plot.width = 30, repr.plot.height = 20)
pnw_coeff <- calculate_coefficient(transformed_monthly_pnw_data)
```

```
## Using scaling coefficient (coeff):2.14553
```

```r
generate_plot(transformed_monthly_pnw_data, pnw_coeff, "Pacific Northwest")
```



5.2.3 Analysis on Rate of New Infections Monthly vs CFR Monthly for the Pacific Northwest Region Overall, the trends that appear on this graph of Monthly Infection Rate vs CFR for the Pacific Northwest end up closely resembling its national counterpart. We still see a more rounded cycle for the infection rate and a sharper cycle for CFR. The peaks and troughs for each rate in the Pacific Northwest graph also exist at the exact same points that they appeared in when we saw the National graph, except for one CFR spike (The US overall had a spike in October 2021 but the Pacific Northwest's spike for this period occured in November).

```
us_data_sliced <- transformed_monthly_us_data %>%
  slice(6:n())

mean_us_cfr <- colSums(us_data_sliced[,'cfr_monthly_period'])/length(us_data_sliced)


pnw_data_sliced <- transformed_monthly_pnw_data %>%
  slice(6:n())

mean_pnw_cfr <- colSums(pnw_data_sliced[,'cfr_monthly_period'])/length(pnw_data_sliced)

print(glue('Mean CFR for the entire US:{round(mean_us_cfr,5)} vs Mean CFR for the Pacific Northwest:{rou
```

```
## Mean CFR for the entire US:0.04848 vs Mean CFR for the Pacific Northwest:0.04304
```

Still, there are some key differences that need to be pointed out. First is that the CFR seemed to have a bit more variability, though at slightly lower levels for the Pacific Northwest while the National CFR was consistently peaking at around the same maximum rate. This has lead to a slightly lesser deaths for the Northwest Region, and can be seen quantitatively from the calculation of CFR means done in the previous cell (I chose to calculate everything starting from June in order to better reflect a rate that represents the CFR when it has settled closer to its real value).

As for the monthly infection rate, it shows that Oregon and Washington were able to manage the first couple of waves a bit better but were subsequently much more impacted by the later wave of cases, especially with regards to the sharp Omicron induced peak in January 2022. The Nation as a whole on the other hand, experienced gradually declining peaks in monthly infections rates outside of the afformentioned Omicron spike.

One final thing to note is the difference in coefficients used for creating the graphs (2.14 for the Pacific Northwest and 0.94 for the US overall), which while initially was only introduced as a way to scale the rates closer together for better visualization outputs, might imply that the Pacific Northwest experienced higher than national deaths relative to the peak infection periods for each wave, but proving that this is the case will require further analysis.

6. Creating a Linear Model

The final major analysis that is proposed for this report is centered around creating a linear model that finds the relationship between the total amount of new cases per month for the United States and total amount of new cases per month globally. We use the former as our independent variable to see if it is a good predictor for the latter.

6.1 Preparing the Data

Most of the data needed for the proposed linear model has already been prepared previously. The global dataset that we will be using has been cleaned, tidied and transformed in section 4.2 and the same has

been accomplished for the US dataset in section 5.1.1. For the final steps, we organize the US table by standardizing the name of the column for new monthly cases and selecting only this and the year_month column before joining this table with its global counterpart.

```
us_monthly_new_cases_df <- transformed_monthly_us_data  %>%
    rename(us_new_monthly_cases = 'new_cases_this_month')  %>%
    select(c('year_month', 'us_new_monthly_cases'))

model_data <- inner_join(us_monthly_new_cases_df, global_monthly_new_cases_df, by = "year_month")

tail(model_data)
```

```
## # A tibble: 6 x 3
##   year_month us_new_monthly_cases global_new_monthly_cases
##   <date>                    <dbl>                    <dbl>
## 1 2022-10-01              1130168                 12795215
## 2 2022-11-01              1321554                 12388536
## 3 2022-12-01              1951848                 17219669
## 4 2023-01-01              1597652                 10270797
## 5 2023-02-01              1080555                  4587649
## 6 2023-03-01               359247                  1247911
```

6.2 Modeling and Plotting Actual vs Expected Graph

We can now create a linear model, using new US monthly cases as our independent variable and new global monthly cases as our independent variable. We also print the summary that is analyzed further below. Lastly, we append a prediction column to our main linear model dataframe, and plot the graph of actual vs predicted using a scatter plot for the former and a line graph for the latter.

```
lm_global_vs_us <- lm(global_new_monthly_cases ~ us_new_monthly_cases, data = model_data)
print("Model Summary:")
```

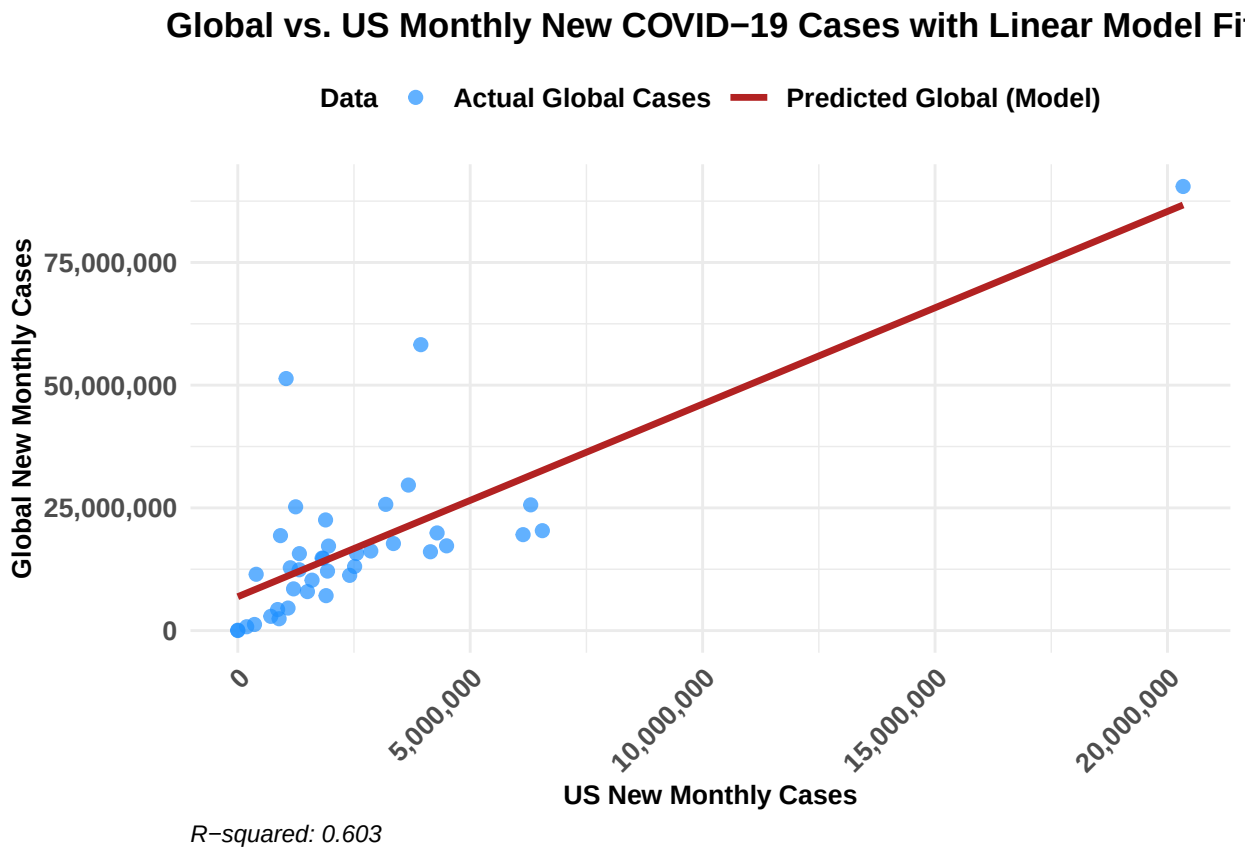
```
## [1] "Model Summary:"
```

```
print(summary(lm_global_vs_us))
```

```
##
## Call:
## lm(formula = global_new_monthly_cases ~ us_new_monthly_cases,
##     data = model_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -12259213  -6807860  -2901679   2836810  40364675
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          6.903e+06  2.229e+06   3.097  0.00372 **
## us_new_monthly_cases 3.925e+00  5.239e-01   7.491 6.34e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 10860000 on 37 degrees of freedom
## Multiple R-squared:  0.6027, Adjusted R-squared:  0.5919
## F-statistic: 56.12 on 1 and 37 DF,  p-value: 6.34e-09
```

```r
model_data_with_predictions <- model_data %>%
  mutate(predicted_global_cases = predict(lm_global_vs_us, newdata = model_data))

options(repr.plot.width = 30, repr.plot.height = 20)
model_vs_actual_plot <- ggplot(model_data_with_predictions, aes(x = us_new_monthly_cases)) +
  geom_point(aes(y = global_new_monthly_cases, color = "Actual Global Cases"), alpha = 0.7, size = 2) +
  geom_line(aes(y = predicted_global_cases, color = "Predicted Global (Model)"), linewidth = 1.2) +
  scale_x_continuous(labels = scales::comma) +
  scale_y_continuous(labels = scales::comma) +
  scale_color_manual(
    name = "Data",
    values = c("Actual Global Cases" = "dodgerblue", "Predicted Global (Model)" = "firebrick")
  ) +
  labs(
    title = "Global vs. US Monthly New COVID-19 Cases with Linear Model Fit",
    x = "US New Monthly Cases",
    y = "Global New Monthly Cases",
    caption = glue("R-squared: { round(summary(lm_global_vs_us)$r.squared, 3)}")
  ) +
  theme_minimal() +
  theme(
    legend.position = "top",
    plot.caption = element_text(hjust = 0, face = "italic"),
    axis.title.x = element_text(size = 10, face = "bold"),
    axis.title.y = element_text(size = 10, face = "bold"),
    axis.text.x = element_text(size = 10, face = "bold", angle = 45, hjust = 1),
    axis.text.y = element_text(size = 10, face = "bold"),
    plot.title = element_text(size = 13, hjust = 0.5, face = 'bold'),
    legend.title = element_text(size = 10, face = "bold"),
    legend.text = element_text(size = 10, face = "bold")
  )

print(model_vs_actual_plot)
```



6.3 Analysis on Linear Model and Graph of Predicted vs Actual

Looking at the summary of our model, the first thing to note are the very small p-values for our intercept, our dependent variable, and for our f-statistic. The small p-value for the last two items on that list indicates that not only are new monthly US cases a significant predictor of new global cases, it also tells us that the model itself as a whole is statistically significant. As for the intercept's p-value, the value of 0.00372 tells us that the intercept of 6.9e6 is statistically significant from a baseline of zero.

The R-squared value of 60.2% represents the variability in new mothly global cases that can be explained by new US monthly cases and indicates a good linear relationship between the two variables. This is made apparent on the graph itself, wherein we see a moderately clustered set of points in a linear fashion centered around our line of prediction.

7. Discussion on Bias

On the topic of general bias within the datasets, it is highly likely that bias has had an impact on the integrity of both global and US Covid Reports. One way in which this bias might be present is if more focus were put on collecting and verifying data for countries that are considered more prestigious or have a wider global impact thus ensuring that the data present in the global report is much more accurate compared to other developing nations that might not have had the same level of support or resources. Bias can also come about through underreporting, as some countries could have manipulated their case and death counts in order to show that their nation performed better than others at managing the pandemic, which could serve to benefit their socio-economic standing in some shape or form.

As for my own personal biases, it was definitely a factor in choosing to focus on the Pacific Northwest as a region in my deeper analysis, as these are the two states I call home myself. Seeing as how I lived through the pandemic almost entirely within the bounds of both states, I was highly interested in seeing how this region fared in comparison to the country as a whole, as I had the preconceived notion that both states were dealing with the crisis better than most states and wanted to see if such was the case. Still, I did my analysis in an objective manner and ultimately I saw that the region as a whole experienced similar monthly case and death rates - which just shows how pandemics such as Covid and its impact on our communities, can only be managed to a certain degree and that the effects it had on populations were all-encompassing.

```
sessionInfo()
```

```
## R version 4.4.2 (2024-10-31 ucrt)
## Platform: x86_64-w64-mingw32/x64
## Running under: Windows 11 x64 (build 26100)
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=English_United States.utf8
## [2] LC_CTYPE=English_United States.utf8
## [3] LC_MONETARY=English_United States.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.utf8
##
## time zone: America/Los_Angeles
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] glue_1.8.0      lubridate_1.9.4 forcats_1.0.0   stringr_1.5.1
##  [5] dplyr_1.1.4     purrr_1.0.2     readr_2.1.5     tidyr_1.3.1
##  [9] tibble_3.2.1    ggplot2_3.5.2   tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] bit_4.6.0          gtable_0.3.6       crayon_1.5.3       compiler_4.4.2
##  [5] tidyselect_1.2.1   parallel_4.4.2     scales_1.4.0       yaml_2.3.10
##  [9] fastmap_1.2.0      R6_2.5.1           labeling_0.4.3     generics_0.1.4
## [13] curl_6.2.2         knitr_1.49         pillar_1.10.0      RColorBrewer_1.1-3
## [17] tzdb_0.5.0         rlang_1.1.4        stringi_1.8.4      xfun_0.49
## [21] bit64_4.6.0-1      timechange_0.3.0   cli_3.6.3          withr_3.0.2
## [25] magrittr_2.0.3     digest_0.6.37      grid_4.4.2         vroom_1.6.5
## [29] rstudioapi_0.17.1  hms_1.1.3          lifecycle_1.0.4    vctrs_0.6.5
## [33] evaluate_1.0.1     farver_2.1.2       rmarkdown_2.29     tools_4.4.2
## [37] pkgconfig_2.0.3    htmltools_0.5.8.1
```