Final Project 2: Reproducible Report on COVID19 Data

F.M

2024-02-26

Objective

In this project, I examine COVID-19 cases in the United States. I aim to investigate whether the number of deaths per thousand is related to the number of cases per thousand. Can we predict the number of deaths based on the number of cases?

DATA IMPORT

Reading in the data from the four main csv files published on GitHub by Johns Hopkins University.

```
## Get current data in the four files
# they all begin the same way
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 <- str_c(url_in, file_names)
```

Loading each file into R.

```
global_cases <- read_csv(urls[1])
global_deaths <- read_csv(urls[2])
US_cases <- read_csv(urls[3])
US_deaths <- read_csv(urls[4])
```

Tidying the Data

Although we have imported both global and US data from GitHub, our focus is on the data for the United States. So, for now, we will ignore the global cases and tidy up the US cases.

Reshape US_cases data: use pivot_longer to convert the wide format to a long format, keeping only relevant columns (Admin2 to cases), parsing dates, and removing unnecessary columns (Lat, Long_).

```
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_))
```

Reshape the US_deaths data similarly to the US_cases.

```
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_))
```

Combine the reshaped data of US_cases and US_deaths using a full join based on common columns.

```
US <- US_cases %>% full_join(US_deaths)
```

Following the steps in the lecture, we group the data based on Province_State, Country_Region, and date. Next, we calculate the total cases and deaths while keeping the Population constant within each group. The variable deaths_per_mill is derived by dividing deaths by Population and multiplying the result by 1,000,000. Afterward, the code selects and reorders the necessary columns and removes the grouping structure.

```
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()
```

Create US_totals by aggregating data at the country and date level. Group the data by Country_Region and date, summarizing total cases and deaths while calculating the total population within each group. The variable deaths_per_mill is computed by dividing total deaths by the total population and multiplying the result by 1,000,000. After this, the relevant columns are selected and reordered. Finally, ungroup the data to remove the grouping structure.

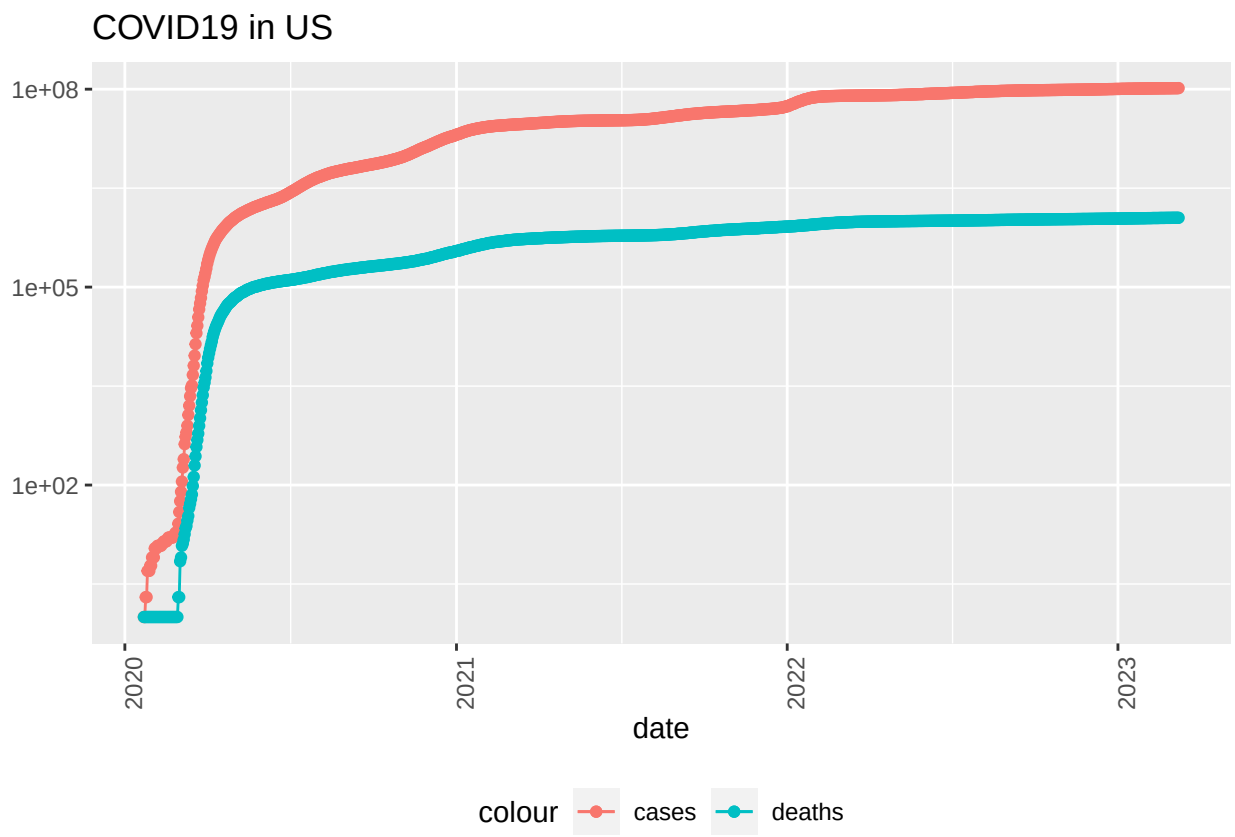
```
US_totals <- US_by_state %>%
  group_by(Country_Region, date) %>%
  summarize(cases = sum(cases), deaths = sum(deaths),
            Population = sum(Population)) %>%
  mutate(deaths_per_mill = deaths * 1000000 / Population) %>%
  select(Country_Region, date, cases, deaths, deaths_per_mill, Population) %>%
  ungroup()
```

Visualizations

Plotting the graph of COVID-19 cases and deaths in the USA.

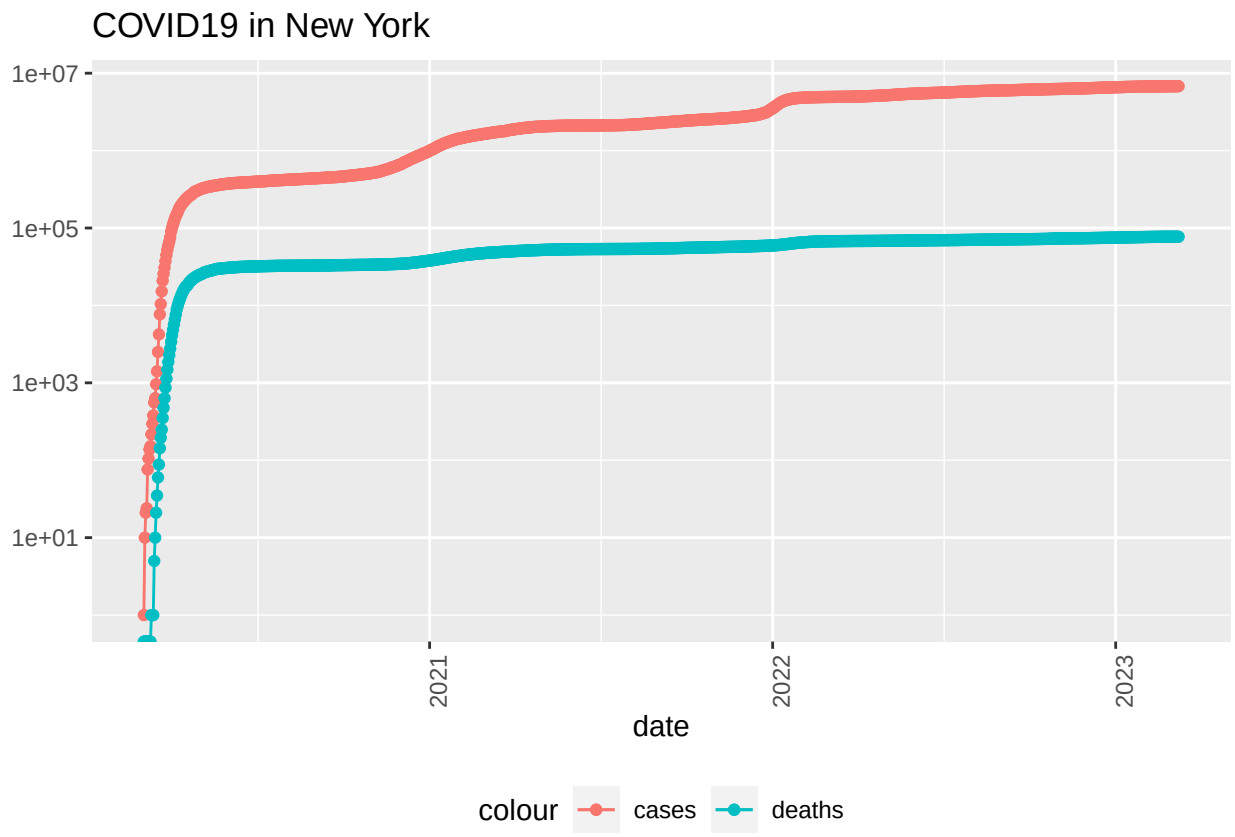
```
US_totals %>%
  filter(cases>0) %>%
```

```
  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)
```



Plotting the graph of COVID-19 cases and deaths in one of the US states.

```
state <- "New York"
US_by_state %>%
  filter(Province_State == state) %>%
  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 = str_c("COVID19 in ",state),y=NULL)
```

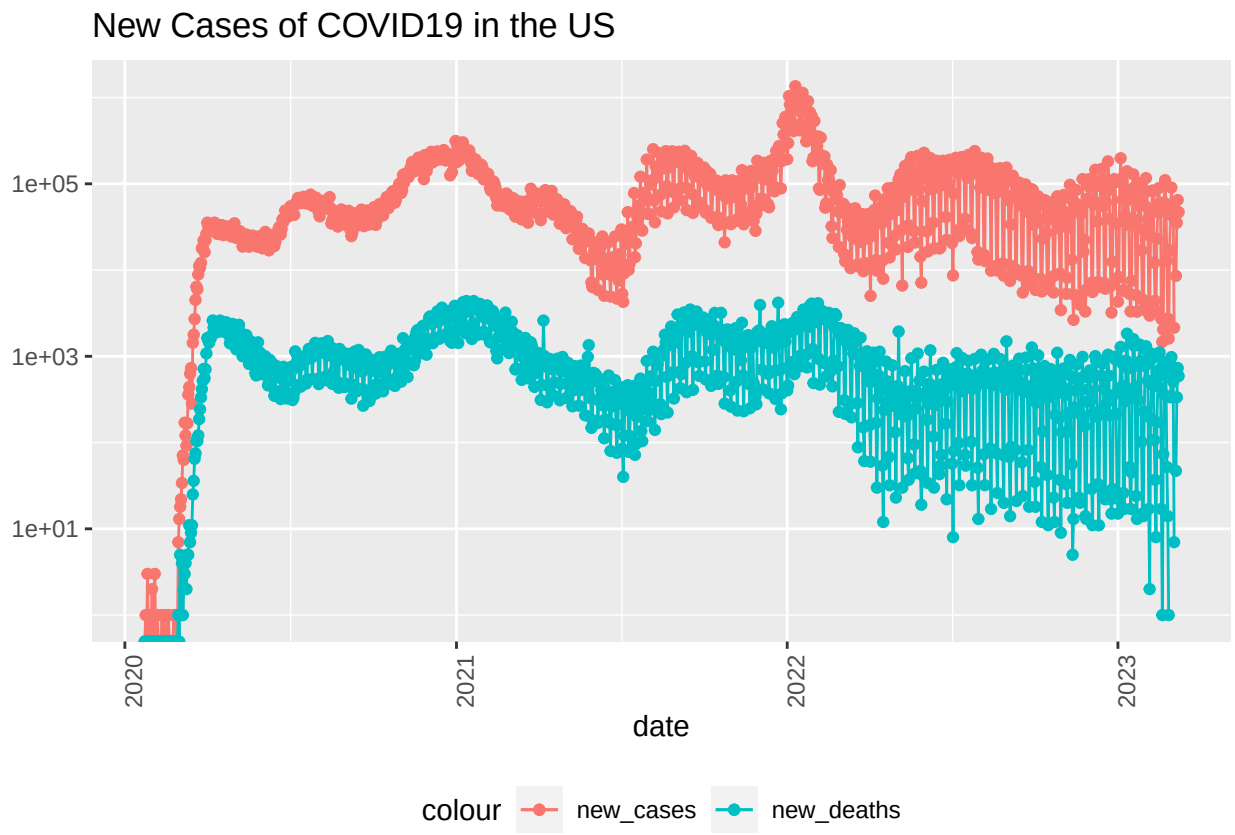


New columns for daily new cases and deaths are incorporated into both the US_by_state and US_totals datasets. The calculation involves subtracting the cases and deaths of the previous day from the corresponding values of the current day. This provides a concise way to track and analyze the daily changes in COVID-19 cases and deaths at both the state and total levels for the United States.

```
US_by_state <- US_by_state %>%
  mutate(new_cases = cases - lag(cases),
         new_deaths = deaths - lag(deaths))
US_totals <- US_totals %>%
  mutate(new_cases = cases - lag(cases),
         new_deaths = deaths - lag(deaths))
```

Plotting the graph of new COVID-19 cases and deaths in the USA.

```
US_totals %>%
  filter(cases>0) %>%
  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 = "New Cases of COVID19 in the US",y=NULL)
```

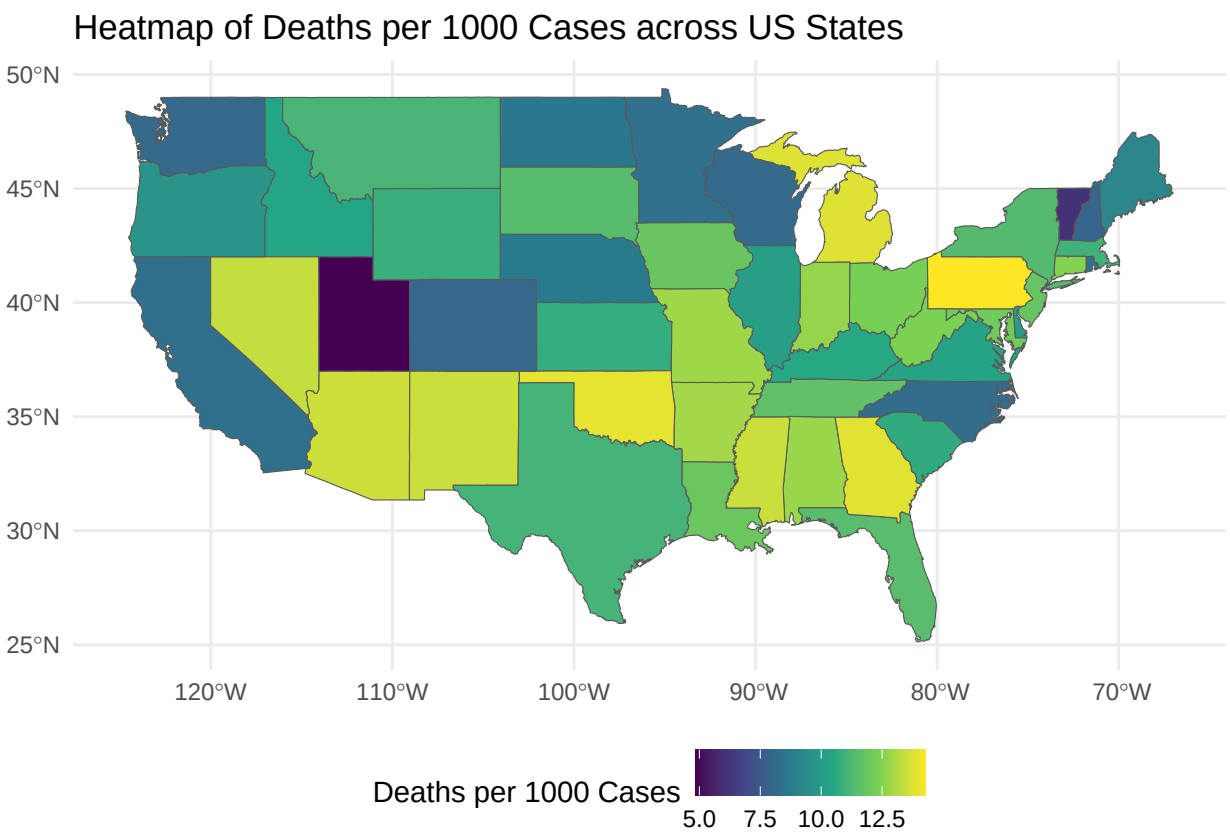


Plotting the heatmap of deaths per 1000 cases across US States

```
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,
            deaths_per_cases = 1000 * deaths / cases) %>%
  filter(cases > 0, population > 0)
```

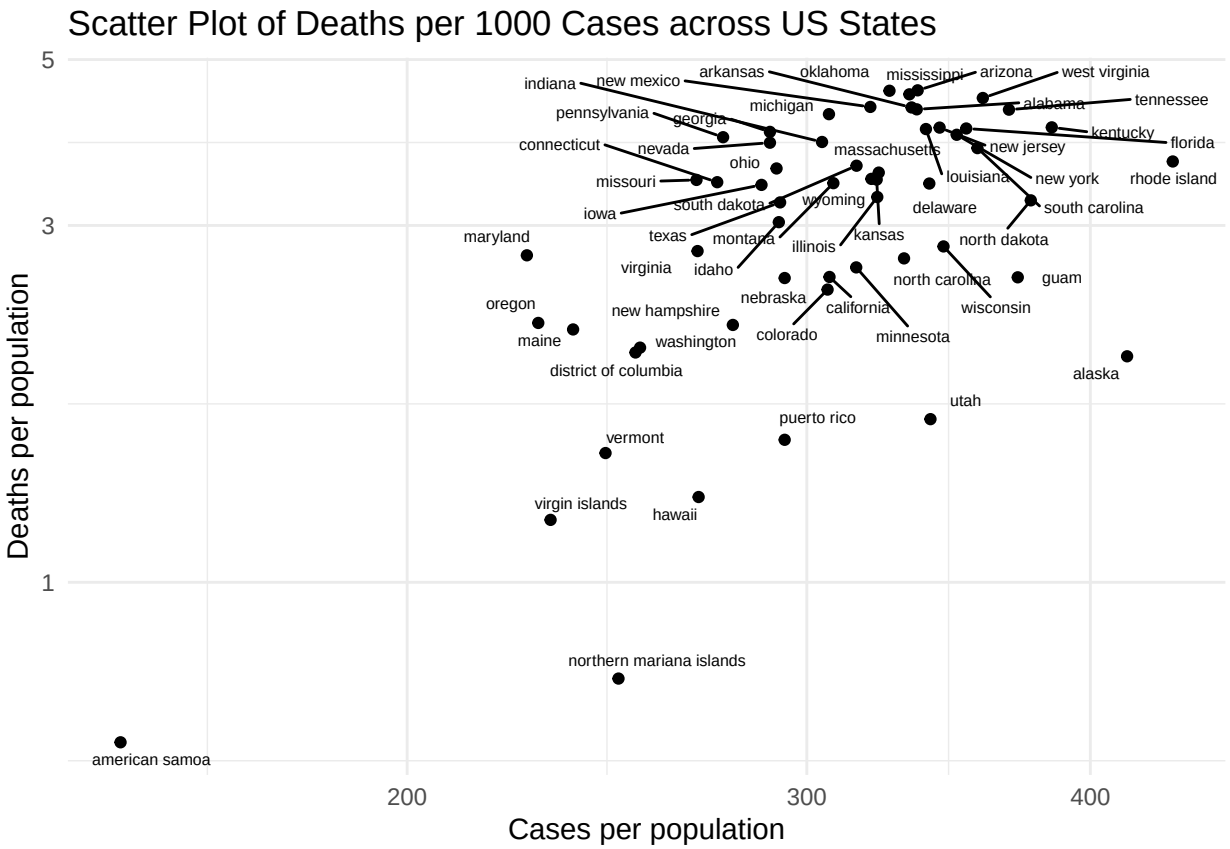
```
#Creating the map
us_states <- maps::map("state", plot = FALSE, fill = TRUE)
us_states <- sf::st_as_sf(us_states)
us_states$ID <- str_to_lower(us_states$ID)
US_state_totals$Province_State <- str_to_lower(US_state_totals$Province_State)

US_state_totals_sf <- left_join(us_states, US_state_totals, by = c("ID" = "Province_State"))

#heatmap
ggplot(data = US_state_totals_sf) +
  geom_sf(aes(fill = deaths_per_cases)) +
  scale_fill_viridis_c() +
  theme_minimal() +
```

```
  labs(title = "Heatmap of Deaths per 1000 Cases across US States",
       fill = "Deaths per 1000 Cases") +
  theme(legend.position = "bottom")
```



Plotting the scatter plot of deaths per 1000 cases across US States

```
#Scatter Plot
US_state_totals %>%
  ggplot(aes(x = cases_per_thou, y = deaths_per_thou, label = Province_State)) +
  geom_point() +
  geom_text_repel(
    box.padding = 0.2,
    point.padding = 0.2,
    force = 20,   # Increase force for better label placement
    size = 2,     # Increase label size
    max.overlaps = Inf
  ) +
  labs(title = "Scatter Plot of Deaths per 1000 Cases across US States", y = "Deaths per population", x
  theme_minimal()+scale_x_log10()+scale_y_log10()
```



Linear Regression Model and Correlation Analysis

We fitted a linear regression model to examine the relationship between deaths per thousand and cases per thousand. The summary of the model is presented below:

```
# Fit a linear regression model
model <- lm(deaths_per_thou ~ cases_per_thou, data = US_state_totals)

# Print the summary
summary(model)
```

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

```
# Calculate the correlation coefficient
correlation_coefficient <- cor(US_state_totals$deaths_per_thou, US_state_totals$cases_per_thou)

# Print the correlation coefficient
print(paste("Correlation Coefficient between Deaths per Thousand and Cases per Thousand: ", correlation_
```

[1] "Correlation Coefficient between Deaths per Thousand and Cases per Thousand: 0.553268495260889"

Conclusion

The analysis found that as the number of COVID-19 cases increases in the U.S., the number of deaths also tends to go up. The model used for this prediction suggests that for every thousand cases, about 0.01133 deaths might occur. This means we can estimate deaths based on the number of observed cases. However, it's crucial to know that this predictive power is limited to the factors we considered in the model and might not cover all the things affecting how many people pass away. Also, just because there's a connection between cases and deaths doesn't mean one directly causes the other.

On another note, there's a chance our analysis might not catch all the important factors affecting outcomes because we specifically chose certain variables for our model. The model simplifies the relationship between deaths and cases, potentially missing some of the complexities of COVID-19. Additionally, there's a risk of bias because of missing data during the process of putting together and organizing the datasets.

```
session_info()
```

```
## - Session info ---------------------------------------------------------------
##  setting  value
##  version  R version 4.3.2 (2023-10-31)
##  os       macOS Sonoma 14.3.1
##  system   aarch64, darwin20
##  ui       X11
##  language (EN)
##  collate  en_US.UTF-8
##  ctype    en_US.UTF-8
##  tz       America/Los_Angeles
##  date     2024-03-01
##  pandoc   3.1.6.2 @ /opt/homebrew/bin/ (via rmarkdown)
##
## - Packages -------------------------------------------------------------------
##  package     * version date (UTC) lib source
##  bit           4.0.5   2022-11-15 [1] CRAN (R 4.3.0)
##  bit64         4.0.5   2020-08-30 [1] CRAN (R 4.3.0)
##  class         7.3-22  2023-05-03 [1] CRAN (R 4.3.2)
##  classInt      0.4-10  2023-09-05 [1] CRAN (R 4.3.0)
##  cli           3.6.1   2023-03-23 [1] CRAN (R 4.3.0)
##  colorspace    2.1-0   2023-01-23 [1] CRAN (R 4.3.0)
##  crayon        1.5.2   2022-09-29 [1] CRAN (R 4.3.0)
##  curl          5.0.1   2023-06-07 [1] CRAN (R 4.3.0)
##  DBI           1.1.3   2022-06-18 [1] CRAN (R 4.3.0)
##  digest        0.6.31  2022-12-11 [1] CRAN (R 4.3.0)
##  dplyr       * 1.1.2   2023-04-20 [1] CRAN (R 4.3.0)
##  e1071         1.7-14  2023-12-06 [1] CRAN (R 4.3.1)
##  evaluate      0.21    2023-05-05 [1] CRAN (R 4.3.0)
##  fansi         1.0.4   2023-01-22 [1] CRAN (R 4.3.0)
##  farver        2.1.1   2022-07-06 [1] CRAN (R 4.3.0)
##  fastmap       1.1.1   2023-02-24 [1] CRAN (R 4.3.0)
##  forcats     * 1.0.0   2023-01-29 [1] CRAN (R 4.3.0)
##  generics      0.1.3   2022-07-05 [1] CRAN (R 4.3.0)
##  ggplot2     * 3.4.2   2023-04-03 [1] CRAN (R 4.3.0)
##  ggrepel     * 0.9.5   2024-01-10 [1] CRAN (R 4.3.1)
```

```
##  glue          1.6.2   2022-02-24 [1] CRAN (R 4.3.0)
##  gtable        0.3.3   2023-03-21 [1] CRAN (R 4.3.0)
##  highr         0.10    2022-12-22 [1] CRAN (R 4.3.0)
##  hms           1.1.3   2023-03-21 [1] CRAN (R 4.3.0)
##  htmltools     0.5.5   2023-03-23 [1] CRAN (R 4.3.0)
##  KernSmooth    2.23-22 2023-07-10 [1] CRAN (R 4.3.2)
##  knitr         1.43    2023-05-25 [1] CRAN (R 4.3.0)
##  labeling      0.4.2   2020-10-20 [1] CRAN (R 4.3.0)
##  lifecycle     1.0.3   2022-10-07 [1] CRAN (R 4.3.0)
##  lubridate   * 1.9.2   2023-02-10 [1] CRAN (R 4.3.0)
##  magrittr      2.0.3   2022-03-30 [1] CRAN (R 4.3.0)
##  maps        * 3.4.2   2023-12-15 [1] CRAN (R 4.3.1)
##  munsell       0.5.0   2018-06-12 [1] CRAN (R 4.3.0)
##  pillar        1.9.0   2023-03-22 [1] CRAN (R 4.3.0)
##  pkgconfig     2.0.3   2019-09-22 [1] CRAN (R 4.3.0)
##  proxy         0.4-27  2022-06-09 [1] CRAN (R 4.3.0)
##  purrr       * 1.0.1   2023-01-10 [1] CRAN (R 4.3.0)
##  R6            2.5.1   2021-08-19 [1] CRAN (R 4.3.0)
##  Rcpp          1.0.11  2023-07-06 [1] CRAN (R 4.3.0)
##  readr       * 2.1.4   2023-02-10 [1] CRAN (R 4.3.0)
##  rlang         1.1.1   2023-04-28 [1] CRAN (R 4.3.0)
##  rmarkdown     2.22    2023-06-01 [1] CRAN (R 4.3.0)
##  rstudioapi    0.14    2022-08-22 [1] CRAN (R 4.3.0)
##  scales        1.2.1   2022-08-20 [1] CRAN (R 4.3.0)
##  sessioninfo * 1.2.2   2021-12-06 [1] CRAN (R 4.3.0)
##  sf          * 1.0-15  2023-12-18 [1] CRAN (R 4.3.1)
##  stringi       1.7.12  2023-01-11 [1] CRAN (R 4.3.0)
##  stringr     * 1.5.0   2022-12-02 [1] CRAN (R 4.3.0)
##  tibble      * 3.2.1   2023-03-20 [1] CRAN (R 4.3.0)
##  tidyr       * 1.3.0   2023-01-24 [1] CRAN (R 4.3.0)
##  tidyselect    1.2.0   2022-10-10 [1] CRAN (R 4.3.0)
##  tidyverse   * 2.0.0   2023-02-22 [1] CRAN (R 4.3.0)
##  timechange    0.2.0   2023-01-11 [1] CRAN (R 4.3.0)
##  tzdb          0.4.0   2023-05-12 [1] CRAN (R 4.3.0)
##  units         0.8-5   2023-11-28 [1] CRAN (R 4.3.1)
##  utf8          1.2.3   2023-01-31 [1] CRAN (R 4.3.0)
##  vctrs         0.6.3   2023-06-14 [1] CRAN (R 4.3.0)
##  viridisLite   0.4.2   2023-05-02 [1] CRAN (R 4.3.0)
##  vroom         1.6.3   2023-04-28 [1] CRAN (R 4.3.0)
##  withr         2.5.0   2022-03-03 [1] CRAN (R 4.3.0)
##  xfun          0.39    2023-04-20 [1] CRAN (R 4.3.0)
##  yaml          2.3.7   2023-01-23 [1] CRAN (R 4.3.0)
##
##  [1] /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library
##
## ------------------------------------------------------------------------------
```