COVID-19 Data Analysis

Andrew Savala

2025-04-05

Overview

In this project I will be analyzing COVID-19 time series data from the the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University. The data set contains daily time series summary tables, including confirmed, deaths and recovered.

My goal is to better understand the COVID-19 pandemic and what factors contributed to deaths in the United States.

On March 10, 2023, the Johns Hopkins Corona Virus Resource Center ceased its collecting and reporting of global COVID-19 data.

Step 1: Import COVID-19 Data

```
# Read time series data from Johns Hopkins University GitHub repository
# Raw base URL
url_base <-
 ↪  "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_
file_names <- c("time_series_covid19_confirmed_US.csv",
                "time_series_covid19_deaths_US.csv")
urls <- str_c(url_base, file_names)

us_cases <- read_csv(urls[1], show_col_types = FALSE)
us_deaths <- read_csv(urls[2], show_col_types = FALSE)
```

Step 2: Tidy and Transform Data

Examine Our Data

Here's our raw data:

```
# Display our raw COVID-19 data
head(us_cases)
```

```
## # A tibble: 6 x 1,154
##        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 84001003 US    USA     840  1003 Baldwin Alabama        US              30.7
## 3 84001005 US    USA     840  1005 Barbour Alabama        US              31.9
## 4 84001007 US    USA     840  1007 Bibb    Alabama        US              33.0
## 5 84001009 US    USA     840  1009 Blount  Alabama        US              34.0
## 6 84001011 US    USA     840  1011 Bullock Alabama        US              32.1
## # i 1,145 more variables: Long_ <dbl>, Combined_Key <chr>, `1/22/20` <dbl>,
## #   `1/23/20` <dbl>, `1/24/20` <dbl>, `1/25/20` <dbl>, `1/26/20` <dbl>,
## #   `1/27/20` <dbl>, `1/28/20` <dbl>, `1/29/20` <dbl>, `1/30/20` <dbl>,
## #   `1/31/20` <dbl>, `2/1/20` <dbl>, `2/2/20` <dbl>, `2/3/20` <dbl>,
## #   `2/4/20` <dbl>, `2/5/20` <dbl>, `2/6/20` <dbl>, `2/7/20` <dbl>,
## #   `2/8/20` <dbl>, `2/9/20` <dbl>, `2/10/20` <dbl>, `2/11/20` <dbl>,
## #   `2/12/20` <dbl>, `2/13/20` <dbl>, `2/14/20` <dbl>, `2/15/20` <dbl>, ...
```

```
head(us_deaths)
```

```
## # A tibble: 6 x 1,155
##        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 84001003 US    USA     840  1003 Baldwin Alabama        US              30.7
## 3 84001005 US    USA     840  1005 Barbour Alabama        US              31.9
## 4 84001007 US    USA     840  1007 Bibb    Alabama        US              33.0
## 5 84001009 US    USA     840  1009 Blount  Alabama        US              34.0
## 6 84001011 US    USA     840  1011 Bullock Alabama        US              32.1
## # i 1,146 more variables: Long_ <dbl>, Combined_Key <chr>, Population <dbl>,
## #   `1/22/20` <dbl>, `1/23/20` <dbl>, `1/24/20` <dbl>, `1/25/20` <dbl>,
## #   `1/26/20` <dbl>, `1/27/20` <dbl>, `1/28/20` <dbl>, `1/29/20` <dbl>,
## #   `1/30/20` <dbl>, `1/31/20` <dbl>, `2/1/20` <dbl>, `2/2/20` <dbl>,
## #   `2/3/20` <dbl>, `2/4/20` <dbl>, `2/5/20` <dbl>, `2/6/20` <dbl>,
## #   `2/7/20` <dbl>, `2/8/20` <dbl>, `2/9/20` <dbl>, `2/10/20` <dbl>,
## #   `2/11/20` <dbl>, `2/12/20` <dbl>, `2/13/20` <dbl>, `2/14/20` <dbl>, ...
```

Lets continue with looking at our US data.

```
# Convert US cases wide to long format and drop columns we don't care about
us_cases_long <- 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_))

# Do the same with the US Deaths data
us_deaths_long <- us_deaths %>%
  pivot_longer(cols = -(UID:Population),
               names_to = "Date", values_to = "Deaths") %>%
  select(Admin2:Deaths) %>%
  mutate(Date = mdy(Date)) %>%
  select(-c(Lat, Long_))

head(us_cases_long)
```

```
## # A tibble: 6 x 6
```

```
##   Admin2  Province_State Country_Region Combined_Key         Date       Cases
##   <chr>   <chr>          <chr>          <chr>                <date>     <dbl>
## 1 Autauga Alabama        US             Autauga, Alabama, US 2020-01-22     0
## 2 Autauga Alabama        US             Autauga, Alabama, US 2020-01-23     0
## 3 Autauga Alabama        US             Autauga, Alabama, US 2020-01-24     0
## 4 Autauga Alabama        US             Autauga, Alabama, US 2020-01-25     0
## 5 Autauga Alabama        US             Autauga, Alabama, US 2020-01-26     0
## 6 Autauga Alabama        US             Autauga, Alabama, US 2020-01-27     0
```

```
head(us_deaths_long)
```

```
## # A tibble: 6 x 7
##   Admin2 Province_State Country_Region Combined_Key Population Date       Deaths
##   <chr>  <chr>          <chr>          <chr>              <dbl> <date>      <dbl>
## 1 Autau~ Alabama        US             Autauga, Al~       55869 2020-01-22      0
## 2 Autau~ Alabama        US             Autauga, Al~       55869 2020-01-23      0
## 3 Autau~ Alabama        US             Autauga, Al~       55869 2020-01-24      0
## 4 Autau~ Alabama        US             Autauga, Al~       55869 2020-01-25      0
## 5 Autau~ Alabama        US             Autauga, Al~       55869 2020-01-26      0
## 6 Autau~ Alabama        US             Autauga, Al~       55869 2020-01-27      0
```

We can combine our US data.

```
# Combine us cases and deaths data
us_data <- us_cases_long %>%
  full_join(us_deaths_long)
```

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

Lets filter our US data to only include states with cases greater than 0. This will help us with our analysis.

```
# Filter US data where the cases and population are positive
us_data <- us_data %>% filter(Cases > 0) %>% filter(Population > 0)
summary(us_data)
```

```
##     Admin2          Province_State     Country_Region     Combined_Key
##  Length:3424407     Length:3424407     Length:3424407     Length:3424407
##  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.   :      1   Min.   :      86   Min.   :    0.0
##  1st Qu.:2020-12-27   1st Qu.:    699   1st Qu.:   11710   1st Qu.:   10.0
##  Median :2021-09-20   Median :   2865   Median :   26830   Median :   47.0
##  Mean   :2021-09-19   Mean   :  15559   Mean   :  106024   Mean   :  204.2
##  3rd Qu.:2022-06-15   3rd Qu.:   9354   3rd Qu.:   69830   3rd Qu.:  138.0
##  Max.   :2023-03-09   Max.   :3710586   Max.   :10039107   Max.   :35545.0
```

Visualizations and Analysis

Lets start with some visualizations of the US data. We will create a plot to show the trends in cases and deaths over time in the US and then focus in on my home state of California.

```
# US totals by state
us_by_state <- us_data %>%
  group_by(Province_State, Country_Region, Date) %>%
  summarise(Cases = sum(Cases), Deaths = sum(Deaths), Population = sum(Population)) %>%
  mutate(Deaths_Per_Mil = Deaths * 1000000 / Population,
         Cases_Per_Mil = Cases * 1000000 / Population) %>%
  select(Province_State, Country_Region, Date, Cases, Deaths, Population, Deaths_Per_Mil,
   ↪  Cases_Per_Mil) %>%
  ungroup()
```

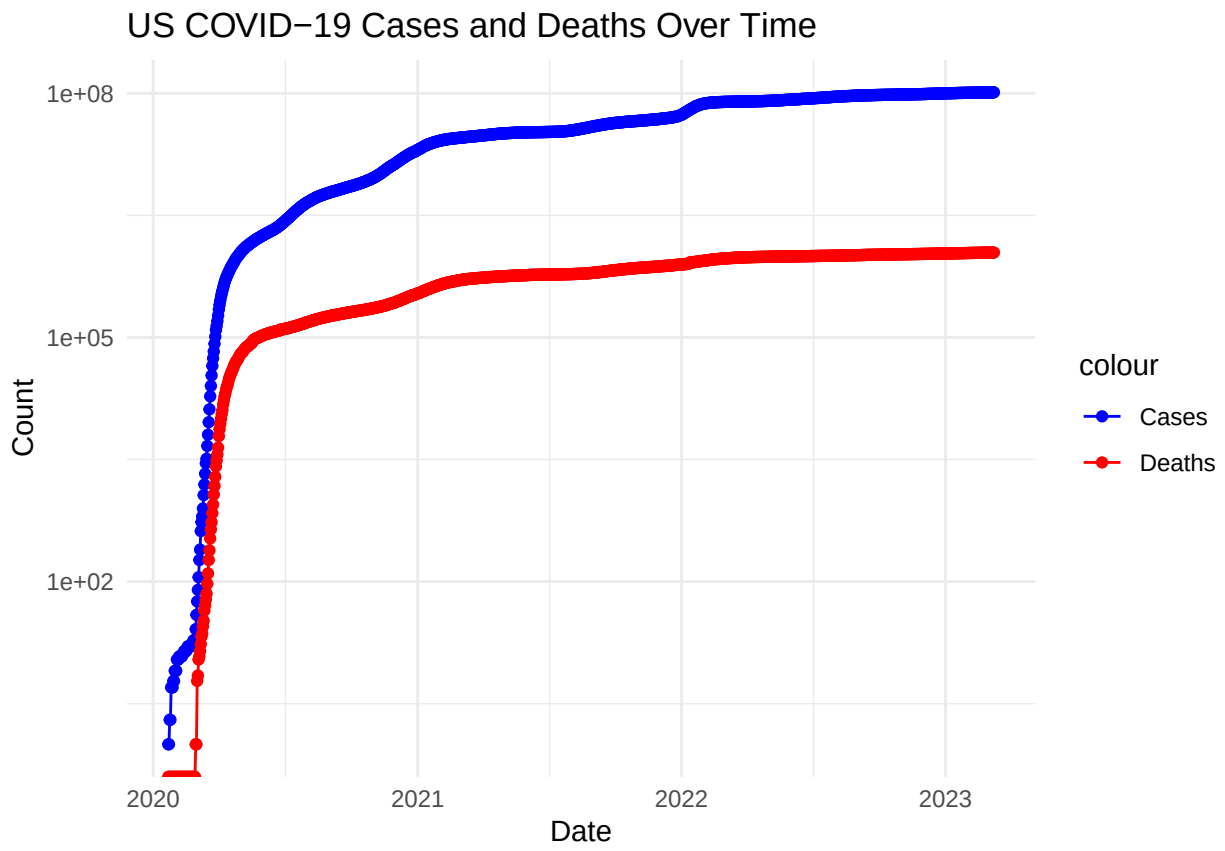
```
## `summarise()` has grouped output by 'Province_State', 'Country_Region'. You can
## override using the `.groups` argument.
```

```
# US totals
us_totals <- us_data %>%
  group_by(Country_Region, Date) %>%
  summarise(Cases = sum(Cases), Deaths = sum(Deaths), Population = sum(Population)) %>%
  mutate(Deaths_Per_Mil = Deaths * 1000000 / Population,
         Cases_Per_Mil = Cases * 1000000 / Population) %>%
  select(Country_Region, Date, Cases, Deaths, Population, Deaths_Per_Mil, Cases_Per_Mil)
   ↪  %>%
  ungroup()
```

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

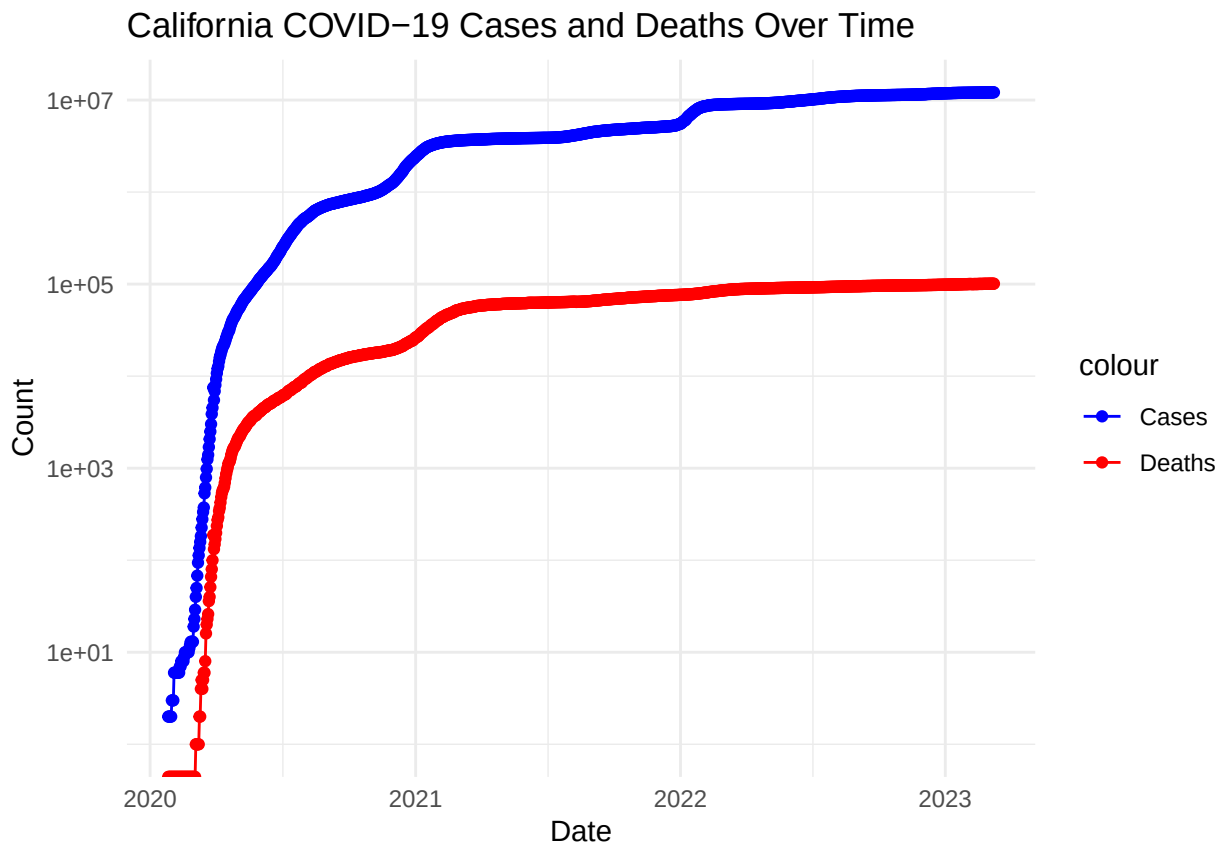
```
# Plot US totals for cases and deaths over time and filter to only show cases > 0.  Lets
 ↪  also scale Y so we can compare the trends.
us_totals %>%
  filter(Cases > 0) %>%
  ggplot(aes(x = Date, y = Cases)) +
  geom_line(aes(y = Cases, color = "Cases")) +
  geom_point(aes(y = Cases, color = "Cases")) +
  geom_line(aes(y = Deaths, color = "Deaths")) +
  geom_point(aes(y = Deaths, color = "Deaths")) +
  scale_y_log10() +
  labs(title = "US COVID-19 Cases and Deaths Over Time",
       x = "Date",
       y = "Count") +
  scale_color_manual(values = c("Cases" = "blue", "Deaths" = "red")) +
  theme_minimal()
```

```
## Warning in scale_y_log10(): log-10 transformation introduced infinite values.
## log-10 transformation introduced infinite values.
```




```
# Now lets do the same thing, but just for the state of California
us_by_state %>%
  filter(Cases > 0) %>%
  filter(Province_State == "California") %>%
  ggplot(aes(x = Date, y = Cases)) +
  geom_line(aes(y = Cases, color = "Cases")) +
  geom_point(aes(y = Cases, color = "Cases")) +
  geom_line(aes(y = Deaths, color = "Deaths")) +
  geom_point(aes(y = Deaths, color = "Deaths")) +
  scale_y_log10() +
  labs(title = "California COVID-19 Cases and Deaths Over Time",
       x = "Date",
       y = "Count") +
  scale_color_manual(values = c("Cases" = "blue", "Deaths" = "red")) +
  theme_minimal()
```

```
## Warning in scale_y_log10(): log-10 transformation introduced infinite values.
## log-10 transformation introduced infinite values.
```

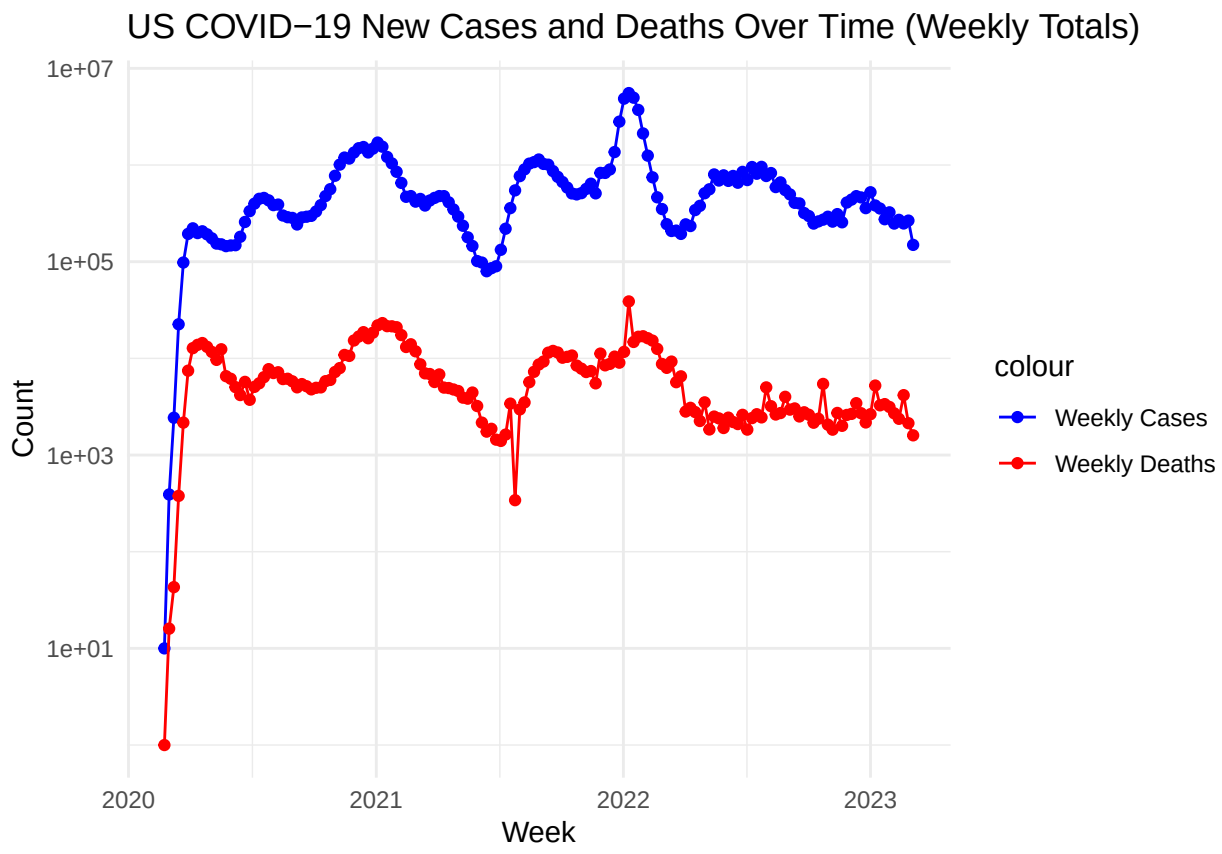


US Analysis

Now lets do some analysis. We will look at the new cases and deaths over time by week.

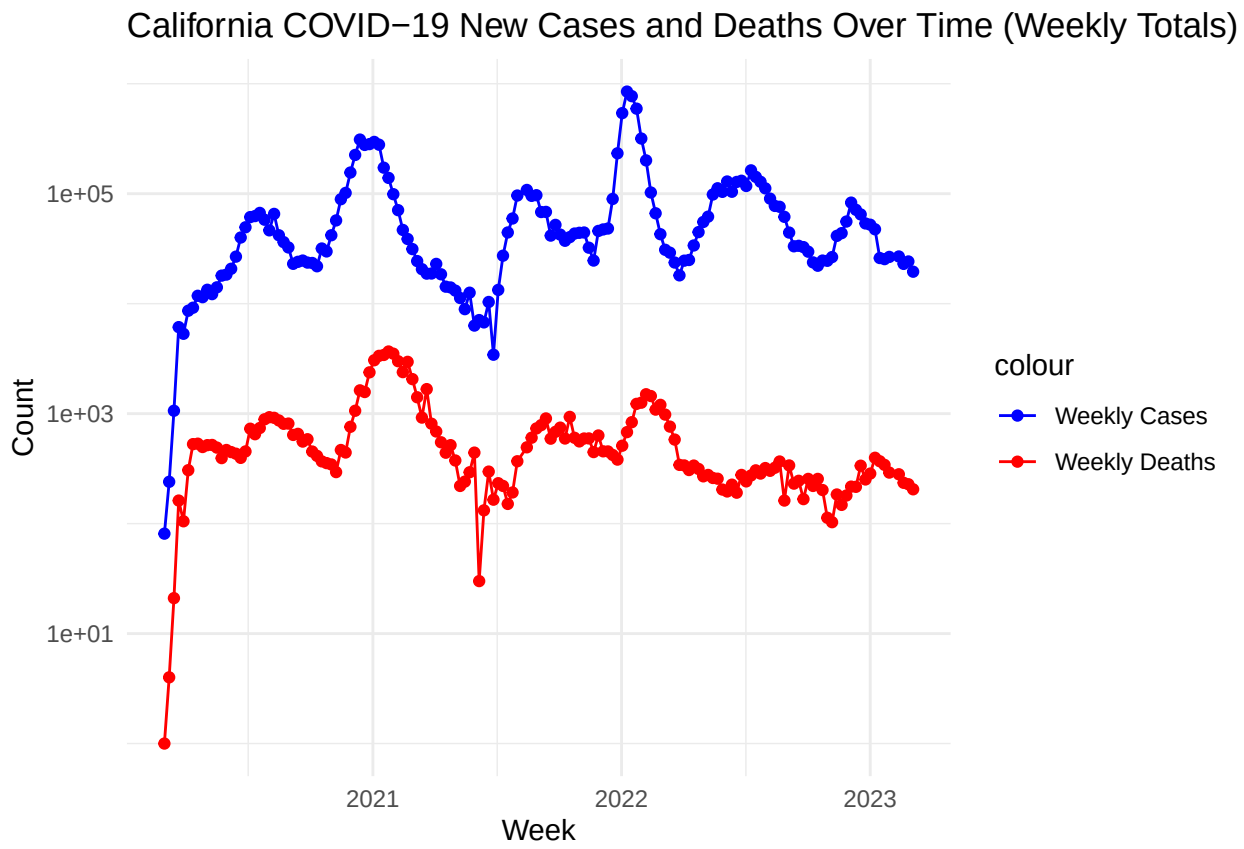
```
us_totals <- us_totals %>%
  mutate(New_Cases = Cases - lag(Cases, default = 0),
         New_Deaths = Deaths - lag(Deaths, default = 0))

us_totals_weekly <- us_totals %>%
  mutate(Week = floor_date(Date, "week")) %>%
  group_by(Week) %>%
  summarise(
    Weekly_Cases = sum(New_Cases, na.rm = TRUE),
    Weekly_Deaths = sum(New_Deaths, na.rm = TRUE)
  )

us_totals_weekly %>%
  filter(Weekly_Cases > 0, Weekly_Deaths > 0) %>%
  ggplot(aes(x = Week, y = Weekly_Cases)) +
  geom_line(aes(color = "Weekly Cases")) +
  geom_point(aes(color = "Weekly Cases")) +
  geom_line(aes(y = Weekly_Deaths, color = "Weekly Deaths")) +
  geom_point(aes(y = Weekly_Deaths, color = "Weekly Deaths")) +
  scale_y_log10() +
  labs(
```

```
    title = "US COVID-19 New Cases and Deaths Over Time (Weekly Totals)",
    x = "Week",
    y = "Count"
  ) +
  scale_color_manual(values = c("Weekly Cases" = "blue", "Weekly Deaths" = "red")) +
  theme_minimal()
```




```
us_by_state <- us_by_state %>%
  mutate(New_Cases = Cases - lag(Cases, default = 0),
         New_Deaths = Deaths - lag(Deaths, default = 0))

ca_totals_weekly <- us_by_state %>%
  filter(Province_State == "California") %>%
  mutate(Week = floor_date(Date, "week")) %>%
  group_by(Week) %>%
  summarise(
    Weekly_Cases = sum(New_Cases, na.rm = TRUE),
    Weekly_Deaths = sum(New_Deaths, na.rm = TRUE)
  )

ca_totals_weekly %>%
  filter(Weekly_Cases > 0, Weekly_Deaths > 0) %>%
  ggplot(aes(x = Week, y = Weekly_Cases)) +
  geom_line(aes(color = "Weekly Cases")) +
  geom_point(aes(color = "Weekly Cases")) +
```

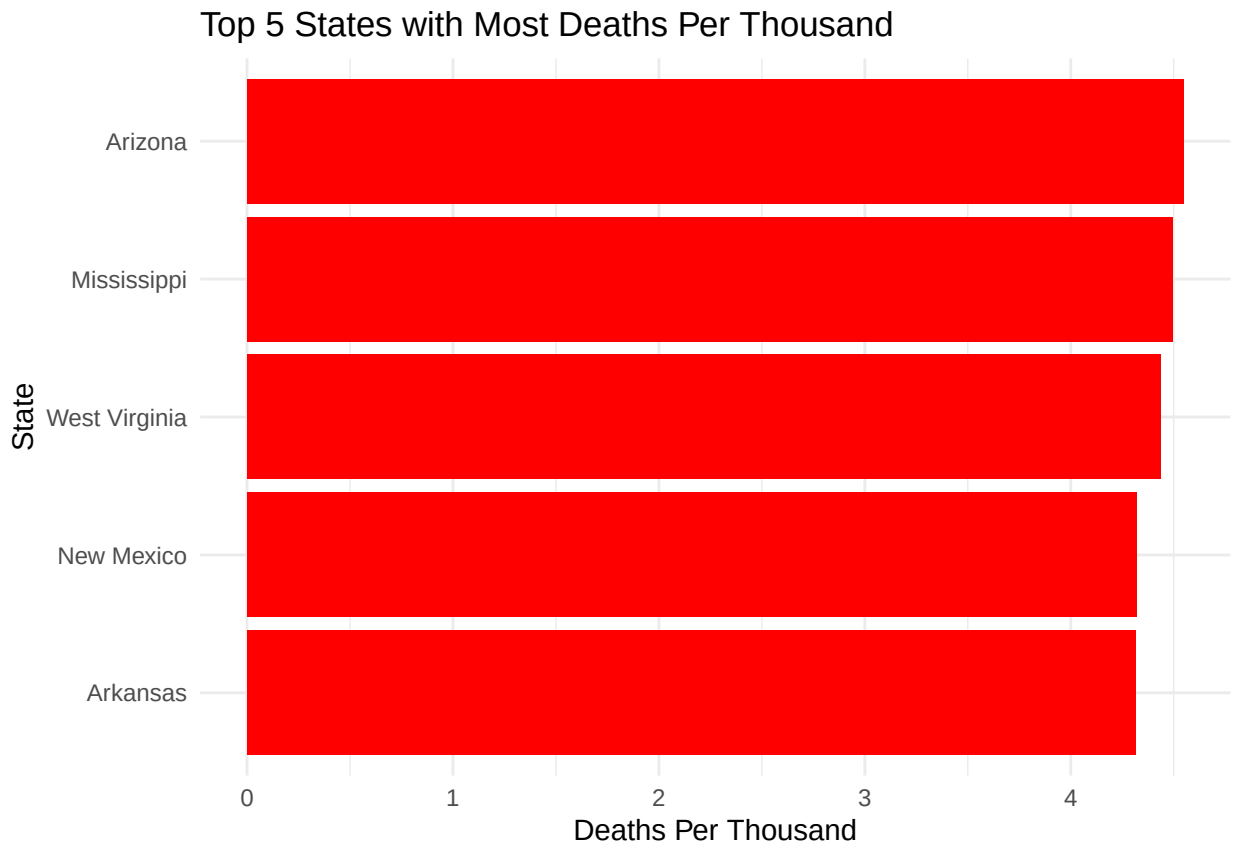
```
  geom_line(aes(y = Weekly_Deaths, color = "Weekly Deaths")) +
  geom_point(aes(y = Weekly_Deaths, color = "Weekly Deaths")) +
  scale_y_log10() +
  labs(
    title = "California COVID-19 New Cases and Deaths Over Time (Weekly Totals)",
    x = "Week",
    y = "Count"
  ) +
  scale_color_manual(values = c("Weekly Cases" = "blue", "Weekly Deaths" = "red")) +
  theme_minimal()
```



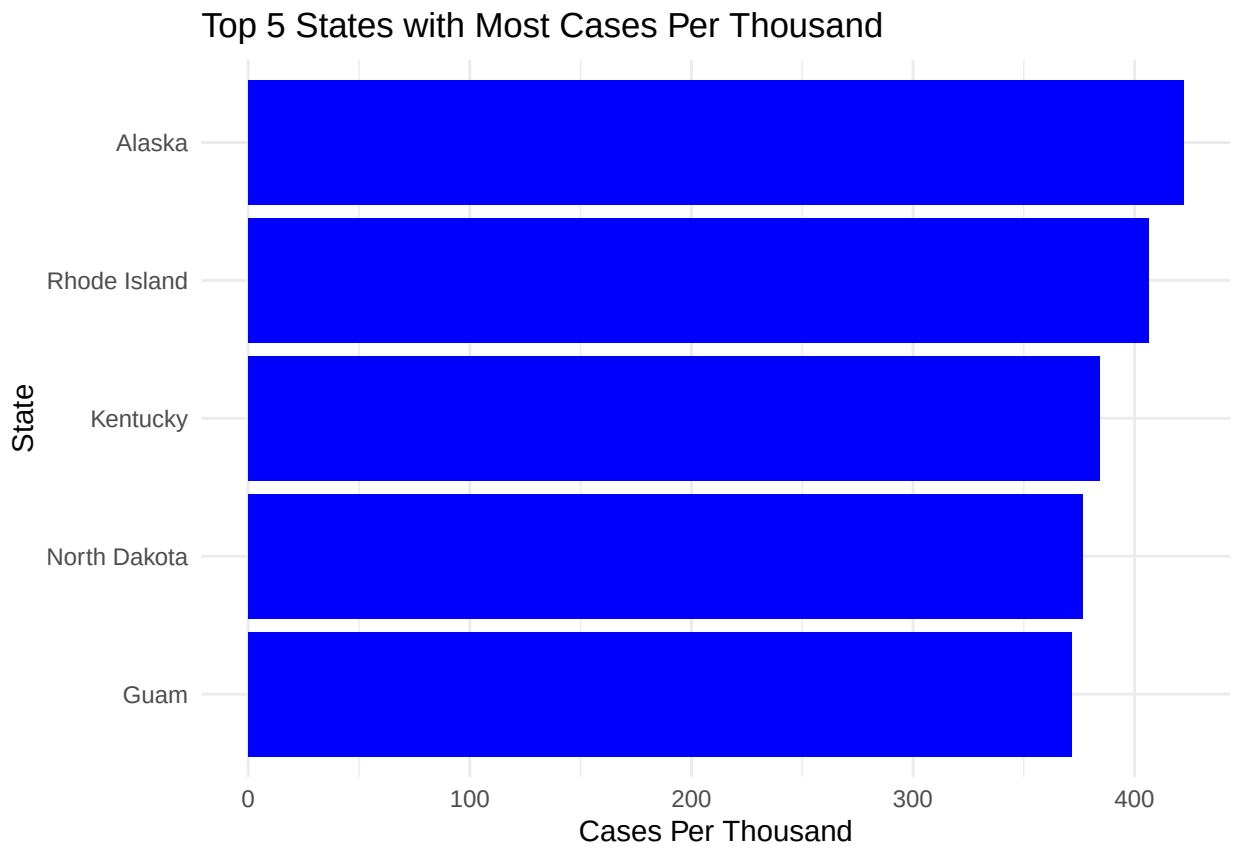
Now lets see which states had the most deaths and cases per thousand.

```
# Calculate cases and deaths per thousand grouped by state
us_state_totals <- us_by_state %>%
  group_by(Province_State) %>%
  summarise(Cases = max(Cases), Deaths = max(Deaths),
            Population = max(Population),
            Cases_Per_Thousand = 1000 * Cases / Population,
            Deaths_Per_Thousand = 1000 * Deaths / Population) %>%
  filter(Cases > 0, Population > 0)

# Find top 5 states with most Deaths_Per_Thousand
us_state_totals %>%
  arrange(desc(Deaths_Per_Thousand)) %>%
```

```
  slice(1:5) %>%
  ggplot(aes(x = reorder(Province_State, Deaths_Per_Thousand), y = Deaths_Per_Thousand))
   ↪  +
  geom_bar(stat = "identity", fill = "red") +
  coord_flip() +
  labs(title = "Top 5 States with Most Deaths Per Thousand",
       x = "State",
       y = "Deaths Per Thousand") +
  theme_minimal()
```




```
# Find top 5 states with most Cases_Per_Thousand
us_state_totals %>%
  arrange(desc(Cases_Per_Thousand)) %>%
  slice(1:5) %>%
  ggplot(aes(x = reorder(Province_State, Cases_Per_Thousand), y = Cases_Per_Thousand)) +
  geom_bar(stat = "identity", fill = "blue") +
  coord_flip() +
  labs(title = "Top 5 States with Most Cases Per Thousand",
       x = "State",
       y = "Cases Per Thousand") +
  theme_minimal()
```



Model

Lets train a linear regression model to predict the number of deaths per thousand based on the number of cases per thousand.

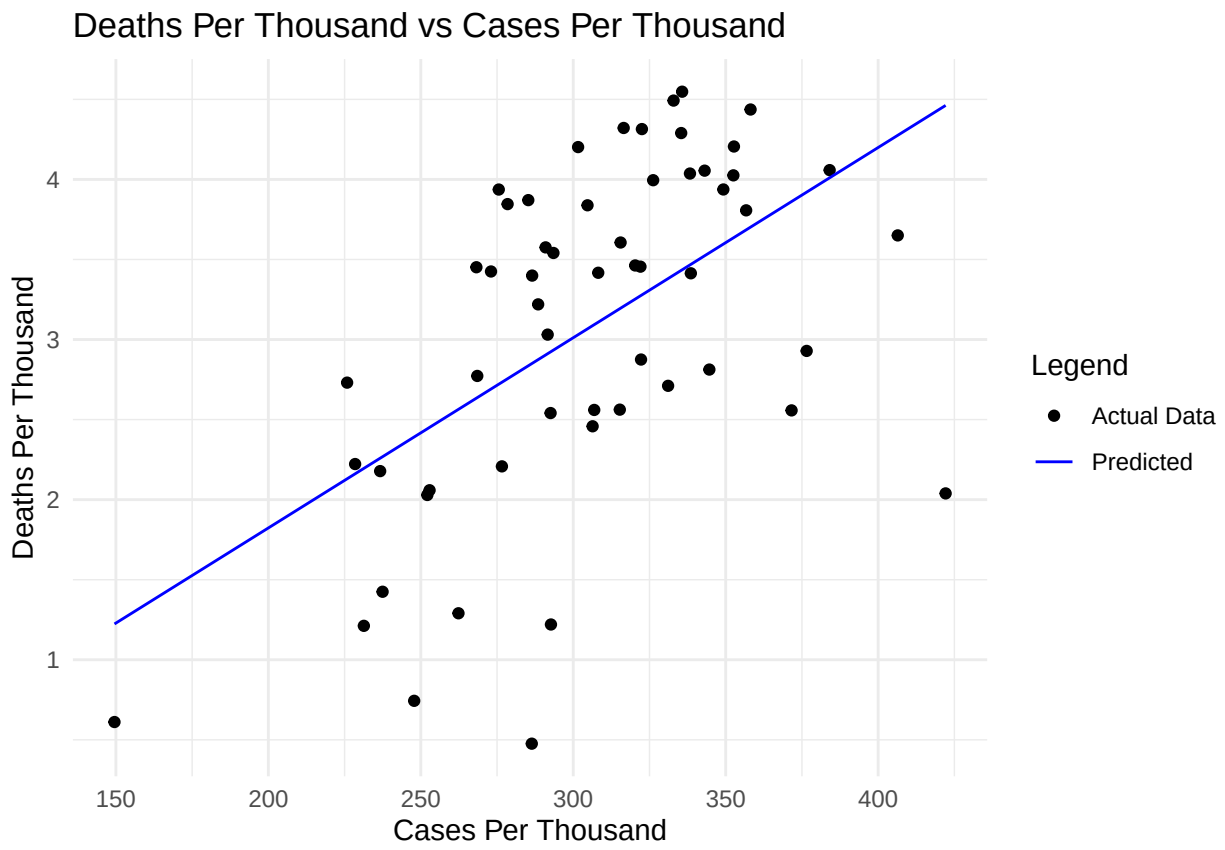
Initial Prediction

```
# Create a linear regression model to predict deaths per thousand based on cases per
 ↪  thousand
model <- lm(Deaths_Per_Thousand ~ Cases_Per_Thousand, data = us_state_totals)
summary(model)
```

```
##
## Call:
## lm(formula = Deaths_Per_Thousand ~ Cases_Per_Thousand, data = us_state_totals)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.4231 -0.6158  0.1588  0.6722  1.2157
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)        -0.552774   0.762354  -0.725    0.472
## Cases_Per_Thousand  0.011880   0.002467   4.816 1.23e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8967 on 54 degrees of freedom
## Multiple R-squared:  0.3004, Adjusted R-squared:  0.2875
## F-statistic: 23.19 on 1 and 54 DF,  p-value: 1.226e-05
```

```r
# Lets add our predicted deaths per thousand to our data as a new column
us_state_totals <- us_state_totals %>%
  mutate(Predicted_Deaths_Per_Thousand = predict(model))

us_state_totals %>%
  ggplot(aes(x = Cases_Per_Thousand, y = Deaths_Per_Thousand)) +
  geom_point(aes(color = "Actual Data")) +
  geom_line(aes(y = Predicted_Deaths_Per_Thousand, color = "Predicted")) +
  labs(title = "Deaths Per Thousand vs Cases Per Thousand",
       x = "Cases Per Thousand",
       y = "Deaths Per Thousand",
       color = "Legend") +
  scale_color_manual(values = c("Actual Data" = "black", "Predicted" = "blue")) +
  theme_minimal()
```



Based on the Multiple R-squared, we can see that our model explains about 30% of the variation in deaths per thousand. This is not a very good model. We can try to improve it by adding more features to our

model.

Additional Features

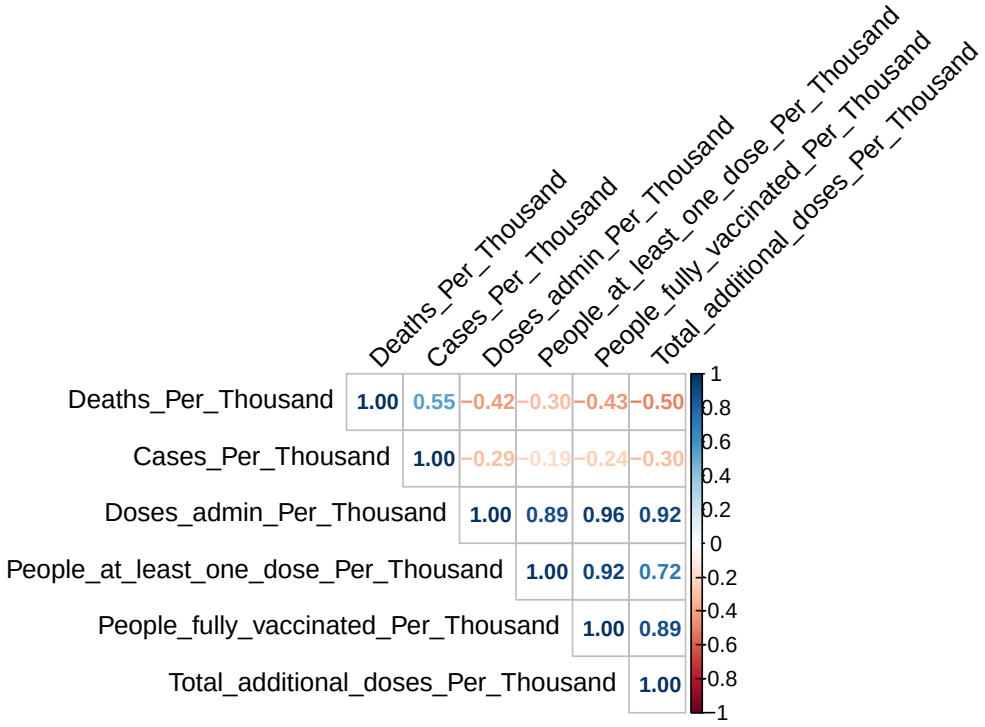
Lets try adding vaccination data to our model and see how that impacts our predictions. We will use the vaccination data from GovEx GitHub repository.

```
# Read vaccination data from GovEx GitHub repository
vaccine_data_url <-
 ↪  "https://raw.githubusercontent.com/govex/COVID-19/refs/heads/master/data_tables/vaccine_data/us_data

# Lets add the columns
 ↪  Doses_admin,People_at_least_one_dose,People_fully_vaccinated,Total_additional_doses
vaccine_data <- read_csv(vaccine_data_url, show_col_types = FALSE) %>%
  select(Date, Province_State, Country_Region, Doses_admin, People_at_least_one_dose,
   ↪  People_fully_vaccinated, Total_additional_doses)

# Find the latest vaccination totals
us_vaccine_totals <- vaccine_data %>%
  group_by(Province_State) %>%
  summarise(
    Doses_admin = max(Doses_admin, na.rm = TRUE),
    People_at_least_one_dose = max(People_at_least_one_dose, na.rm = TRUE),
    People_fully_vaccinated = max(People_fully_vaccinated, na.rm = TRUE),
    Total_additional_doses = max(Total_additional_doses, na.rm = TRUE)
  )

# Join vaccine data to us state totals data
us_state_totals <- us_state_totals %>%
  left_join(us_vaccine_totals, by = "Province_State")

# Make a per thousand column for the vaccine data
us_state_totals <- us_state_totals %>%
  mutate(
    Doses_admin_Per_Thousand = 1000 * Doses_admin / Population,
    People_at_least_one_dose_Per_Thousand = 1000 * People_at_least_one_dose / Population,
    People_fully_vaccinated_Per_Thousand = 1000 * People_fully_vaccinated / Population,
    Total_additional_doses_Per_Thousand = 1000 * Total_additional_doses / Population
  )
```

Looking at our new vaccination data, we have several features to choose from. They're likely highly correlated. Lets see how they see how they correlate with our deaths per thousand and one another.

```
# Correlation matrix for deaths per thousand and vaccination data
vaccine_data <- us_state_totals %>%
  select(Deaths_Per_Thousand, Cases_Per_Thousand, Doses_admin_Per_Thousand,
   ↪  People_at_least_one_dose_Per_Thousand, People_fully_vaccinated_Per_Thousand,
   ↪  Total_additional_doses_Per_Thousand)
correlation_matrix <- cor(vaccine_data, use = "pairwise.complete.obs")
# Plot the correlation matrix
# corrplot(correlation_matrix, method = "circle", type = "upper", tl.col = "black",
 ↪  tl.srt = 45, title = "Correlation Matrix for Deaths Per Thousand and Vaccination
 ↪  Data")
```

```r
corrplot(
  correlation_matrix,
  method = "number",         # Show correlation coefficients
  type = "upper",            # Only show the upper half
  tl.col = "black",          # Text label color
  tl.srt = 45,               # Rotate text labels
  title = "Correlation Matrix for Deaths Per Thousand and Vaccination Data",
  mar = c(0, 0, 1, 0),       # Adjust margins (optional)
  number.cex = 0.8           # Control size of the correlation coefficients
)
```



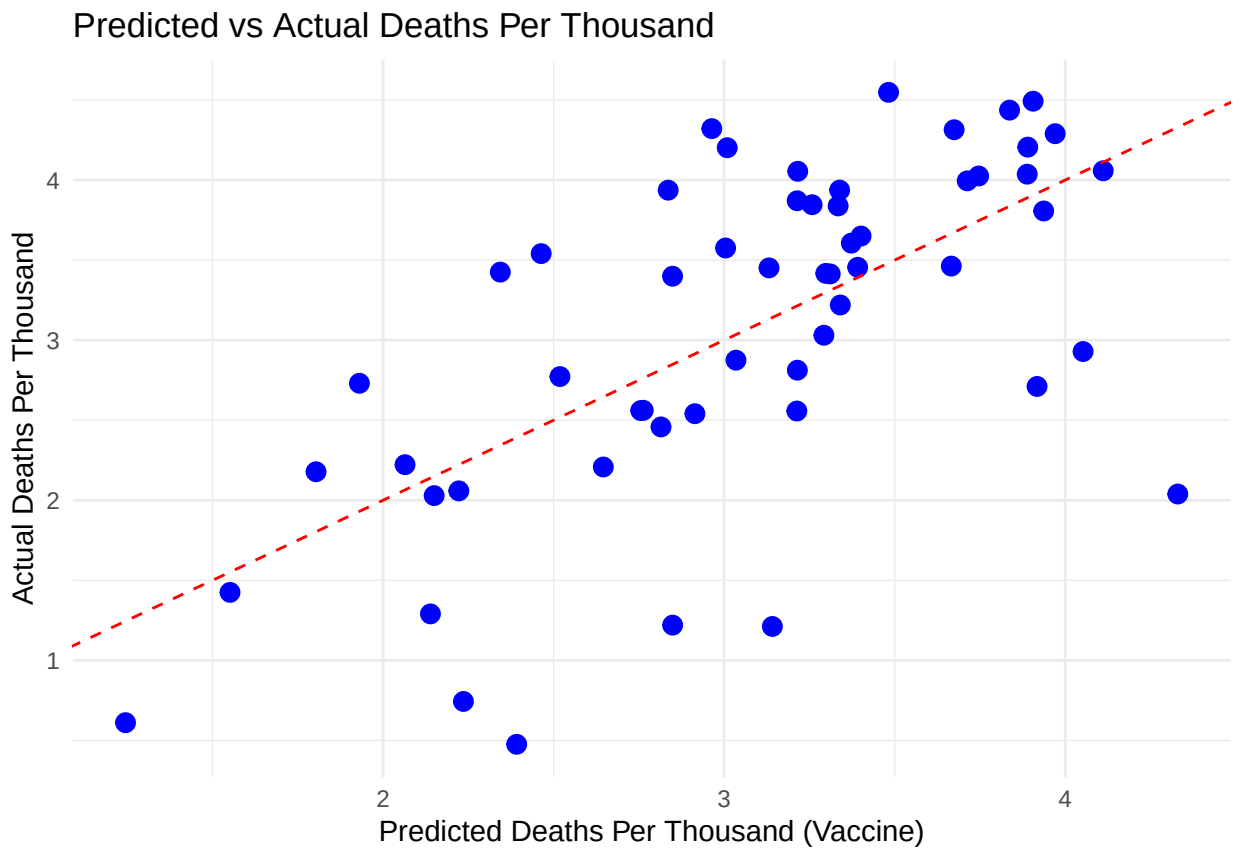
Additional doses per thousand seems to have the strongest correlation with deaths per thousand. Lets try training a new model and add the additional doses per thousand feature.

```r
# Create a linear regression model to predict deaths per thousand based on cases per
 ↪  thousand and additional doses per thousand
model_vaccine <- lm(Deaths_Per_Thousand ~ Cases_Per_Thousand +
 ↪  Total_additional_doses_Per_Thousand, data = us_state_totals)
summary(model_vaccine)
```

```
##
## Call:
## lm(formula = Deaths_Per_Thousand ~ Cases_Per_Thousand + Total_additional_doses_Per_Thousand,
```

```
##     data = us_state_totals)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.2908 -0.2854  0.1337  0.5752  1.3575
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                        1.661495   0.968965   1.715 0.092241 .
## Cases_Per_Thousand                 0.009493   0.002379   3.990 0.000204 ***
## Total_additional_doses_Per_Thousand -0.004141   0.001252  -3.308 0.001693 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8241 on 53 degrees of freedom
## Multiple R-squared:  0.4202, Adjusted R-squared:  0.3983
## F-statistic:  19.2 on 2 and 53 DF,  p-value: 5.342e-07
```

```r
# Now lets compare our predicted deaths per thousand to our actual deaths per thousand
us_state_totals <- us_state_totals %>%
  mutate(Predicted_Deaths_Per_Thousand_Vaccine = predict(model_vaccine))

# Now lets plot our Predicted_Deaths_Per_Thousand_Vaccine vs the actual
 ↪  Deaths_Per_Thousand
ggplot(us_state_totals, aes(x = Predicted_Deaths_Per_Thousand_Vaccine, y =
 ↪  Deaths_Per_Thousand)) +
  geom_point(color = "blue", size = 3) +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  labs(x = "Predicted Deaths Per Thousand (Vaccine)",
       y = "Actual Deaths Per Thousand",
       title = "Predicted vs Actual Deaths Per Thousand") +
  theme_minimal()
```



We're going in the right direction because Multiple R-squared indicates our model explains about 42% of the variation in deaths per thousand. This is a significant improvement over our previous model.

Bias Identification and Conclusion

Regarding bias, it's important to recognize that different states may have different criteria for attributing deaths to COVID-19. Some may report more conservatively, while others more liberally. This could lead to bias in our model and our predictions. Additionally, the data is not perfect and there may be missing or inaccurate data points. This could also lead to bias in our model and our predictions.

In conclusion, we were able to train a model which was able to explain roughly 42% of the variation in deaths per thousand. We were able to do this by adding additional features to our model, specifically vaccination data. This is a significant improvement over our previous model which only explained about 30% of the variation in deaths per thousand.

We can see that the model is not perfect, but it does provide some insight into the relationship between cases per thousand and deaths per thousand.

Session Info

Record our session info for reproducibility.

```
sessionInfo()
```

```
## R version 4.4.3 (2025-02-28 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] forcats_1.0.0   stringr_1.5.1   dplyr_1.1.4     purrr_1.0.4
##  [5] readr_2.1.5     tidyr_1.3.1     tibble_3.2.1    ggplot2_3.5.1
##  [9] tidyverse_2.0.0 corrplot_0.95   lubridate_1.9.4
##
## loaded via a namespace (and not attached):
##  [1] bit_4.6.0         gtable_0.3.6      crayon_1.5.3      compiler_4.4.3
##  [5] tidyselect_1.2.1  parallel_4.4.3    scales_1.3.0      yaml_2.3.10
##  [9] fastmap_1.2.0     R6_2.6.1          labeling_0.4.3    generics_0.1.3
## [13] curl_6.2.1        knitr_1.49        munsell_0.5.1     pillar_1.10.1
## [17] tzdb_0.4.0        rlang_1.1.5       utf8_1.2.4        stringi_1.8.4
## [21] xfun_0.51         bit64_4.6.0-1     timechange_0.3.0  cli_3.6.4
## [25] withr_3.0.2       magrittr_2.0.3    digest_0.6.37     grid_4.4.3
## [29] vroom_1.6.5       rstudioapi_0.17.1 hms_1.1.3         lifecycle_1.0.4
## [33] vctrs_0.6.5       evaluate_1.0.3    glue_1.8.0        farver_2.1.2
## [37] colorspace_2.1-1  rmarkdown_2.29    tools_4.4.3       pkgconfig_2.0.3
## [41] htmltools_0.5.8.1
```