Johns Hopkins COVID 19 Report

DTSA 5301 Data Science as a Field

MS Data Science, University of Colorado Boulder

2024-08-20

Setup Knit Options

echo = true will display code chunks in the output

Load Libraries

```r
library(tidyverse)
```

```
## -- 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   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     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 error
```

```r
library(conflicted)
library(lubridate)
library(caret)
```

```
## Loading required package: lattice
```

```r
library(xgboost)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```r
library(PRROC)
library(MLmetrics)
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```
library(car)
```

```
## Loading required package: carData
```

```
library(smotefamily)
library(ROSE)
```

```
## Loaded ROSE 0.0-4
```

Read Datasets

Download COVID 19 data from Johns Hopkins University, for US and Global cases and deaths.

```
url <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_
us_cases <- read_csv(url)
```

```
## 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.
```

```
url <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_
global_cases <- read_csv(url)
```

```
## 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.
```

```
url <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_
us_deaths <- read_csv(url)
```

```
## 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.
```

```
url <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_
global_deaths <- read_csv(url)
```

```
## 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.
```

Inspect Data

Here we can see the first 5 rows of the data and can page to the right to see all the columns. We start with inspecting us cases.

```
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>, ...
```

Next we inspect us deaths.

```r
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>, ...
```

Next we inspect global cases.

```r
head(global_cases)
```

```
## # A tibble: 6 x 1,147
##   `Province/State` `Country/Region`   Lat  Long `1/22/20` `1/23/20` `1/24/20`
##   <chr>            <chr>            <dbl> <dbl>     <dbl>     <dbl>     <dbl>
## 1 <NA>             Afghanistan       33.9 67.7          0         0         0
## 2 <NA>             Albania           41.2 20.2          0         0         0
## 3 <NA>             Algeria           28.0  1.66         0         0         0
## 4 <NA>             Andorra           42.5  1.52         0         0         0
## 5 <NA>             Angola           -11.2 17.9          0         0         0
## 6 <NA>             Antarctica       -71.9 23.3          0         0         0
## # i 1,140 more variables: `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>, `2/16/20` <dbl>,
## #   `2/17/20` <dbl>, `2/18/20` <dbl>, `2/19/20` <dbl>, `2/20/20` <dbl>, ...
```

Next we inspect global deaths.

```r
head(global_deaths)
```

```
## # A tibble: 6 x 1,147
##   `Province/State` `Country/Region`   Lat  Long `1/22/20` `1/23/20` `1/24/20`
##   <chr>            <chr>            <dbl> <dbl>     <dbl>     <dbl>     <dbl>
## 1 <NA>             Afghanistan       33.9 67.7          0         0         0
## 2 <NA>             Albania           41.2 20.2          0         0         0
## 3 <NA>             Algeria           28.0  1.66         0         0         0
## 4 <NA>             Andorra           42.5  1.52         0         0         0
## 5 <NA>              Angola           -11.2 17.9          0         0         0
## 6 <NA>              Antarctica       -71.9 23.3          0         0         0
## # i 1,140 more variables: ‘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>, ‘2/16/20‘ <dbl>,
## #   ‘2/17/20‘ <dbl>, ‘2/18/20‘ <dbl>, ‘2/19/20‘ <dbl>, ‘2/20/20‘ <dbl>, ...
```

pivot data

Next we will pivot the data to make it easier to work with. This will transform the data from wide to long format. We start with us cases.

```
# pivot us cases
us_cases <- us_cases %>%
  pivot_longer(cols = -c(UID, iso2, iso3, code3, FIPS, Admin2, Province_State, Country_Region, Lat, Long
  select(-c(UID, iso2, iso3, code3, FIPS, Lat, Long_, Combined_Key))
us_cases
```

```
## # A tibble: 3,819,906 x 5
##    Admin2  Province_State Country_Region date    cases
##    <chr>   <chr>          <chr>          <chr>   <dbl>
##  1 Autauga Alabama        US             1/22/20     0
##  2 Autauga Alabama        US             1/23/20     0
##  3 Autauga Alabama        US             1/24/20     0
##  4 Autauga Alabama        US             1/25/20     0
##  5 Autauga Alabama        US             1/26/20     0
##  6 Autauga Alabama        US             1/27/20     0
##  7 Autauga Alabama        US             1/28/20     0
##  8 Autauga Alabama        US             1/29/20     0
##  9 Autauga Alabama        US             1/30/20     0
## 10 Autauga Alabama        US             1/31/20     0
## # i 3,819,896 more rows
```

We do the same for us deaths.

```
# pivot us deaths
us_deaths <- us_deaths %>%
  pivot_longer(cols = -c(UID, iso2, iso3, code3, FIPS, Admin2, Province_State, Country_Region, Lat, Long
  select(-c(UID, iso2, iso3, code3, FIPS, Lat, Long_))
us_deaths
```

```
## # A tibble: 3,819,906 x 7
##    Admin2  Province_State Country_Region Combined_Key    Population date  deaths
##    <chr>   <chr>          <chr>          <chr>                <dbl> <chr>  <dbl>
##  1 Autauga Alabama        US             Autauga, Alaba~      55869 1/22~      0
##  2 Autauga Alabama        US             Autauga, Alaba~      55869 1/23~      0
##  3 Autauga Alabama        US             Autauga, Alaba~      55869 1/24~      0
##  4 Autauga Alabama        US             Autauga, Alaba~      55869 1/25~      0
##  5 Autauga Alabama        US             Autauga, Alaba~      55869 1/26~      0
##  6 Autauga Alabama        US             Autauga, Alaba~      55869 1/27~      0
##  7 Autauga Alabama        US             Autauga, Alaba~      55869 1/28~      0
##  8 Autauga Alabama        US             Autauga, Alaba~      55869 1/29~      0
##  9 Autauga Alabama        US             Autauga, Alaba~      55869 1/30~      0
## 10 Autauga Alabama        US             Autauga, Alaba~      55869 1/31~      0
## # i 3,819,896 more rows
```

We do the same for global cases.

```
# pivot global cases
global_cases <- global_cases %>%
  pivot_longer(cols = -c("Province/State", "Country/Region", "Lat", "Long"), names_to = "date", values_
  select(-c(Lat, Long))
global_cases
```

```
## # A tibble: 330,327 x 4
##    `Province/State` `Country/Region` date    cases
##    <chr>            <chr>            <chr>   <dbl>
##  1 <NA>             Afghanistan      1/22/20     0
##  2 <NA>             Afghanistan      1/23/20     0
##  3 <NA>             Afghanistan      1/24/20     0
##  4 <NA>             Afghanistan      1/25/20     0
##  5 <NA>             Afghanistan      1/26/20     0
##  6 <NA>             Afghanistan      1/27/20     0
##  7 <NA>             Afghanistan      1/28/20     0
##  8 <NA>             Afghanistan      1/29/20     0
##  9 <NA>             Afghanistan      1/30/20     0
## 10 <NA>             Afghanistan      1/31/20     0
## # i 330,317 more rows
```

We do the same for global deaths.

```
# pivot global deaths
global_deaths <- global_deaths %>%
  pivot_longer(cols = -c("Province/State", "Country/Region", "Lat", "Long"), names_to = "date", values_
  select(-c(Lat, Long))
global_deaths
```

```
## # A tibble: 330,327 x 4
##    `Province/State` `Country/Region` date    deaths
##    <chr>            <chr>            <chr>    <dbl>
##  1 <NA>             Afghanistan      1/22/20      0
##  2 <NA>             Afghanistan      1/23/20      0
##  3 <NA>             Afghanistan      1/24/20      0
##  4 <NA>             Afghanistan      1/25/20      0
##  5 <NA>             Afghanistan      1/26/20      0
##  6 <NA>             Afghanistan      1/27/20      0
##  7 <NA>             Afghanistan      1/28/20      0
##  8 <NA>             Afghanistan      1/29/20      0
##  9 <NA>             Afghanistan      1/30/20      0
## 10 <NA>             Afghanistan      1/31/20      0
## # i 330,317 more rows
```

combine cases and deaths, rename columns, convert dates

Next we will combine cases and deaths, rename columns, and convert dates. We can combine the us cases and deaths because they have columns in common to match on. Then we rename the column “Admin2” to “City” to be more descriptive and match the global data. Then we convert the date column to a date object. Finally we select the columns we want to keep. We will start with the us data.

```
us <- us_cases %>%
  full_join(us_deaths) %>%
  rename("City" = "Admin2") %>%
  mutate(date = mdy(date)) %>%
  select("City", "Province_State", "Country_Region", "date", "cases", "deaths", "Population", "Combined
```

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

```
us
```

```
## # A tibble: 3,819,906 x 8
##    City    Province_State Country_Region date       cases deaths Population
##    <chr>   <chr>          <chr>          <date>     <dbl>  <dbl>      <dbl>
##  1 Autauga Alabama        US             2020-01-22     0      0      55869
##  2 Autauga Alabama        US             2020-01-23     0      0      55869
##  3 Autauga Alabama        US             2020-01-24     0      0      55869
##  4 Autauga Alabama        US             2020-01-25     0      0      55869
##  5 Autauga Alabama        US             2020-01-26     0      0      55869
##  6 Autauga Alabama        US             2020-01-27     0      0      55869
##  7 Autauga Alabama        US             2020-01-28     0      0      55869
##  8 Autauga Alabama        US             2020-01-29     0      0      55869
##  9 Autauga Alabama        US             2020-01-30     0      0      55869
## 10 Autauga Alabama        US             2020-01-31     0      0      55869
## # i 3,819,896 more rows
## # i 1 more variable: Combined_Key <chr>
```

Next we do the same for the global data. We first combine the global cases with deaths. Then we rename the columns to match the us data. Then we convert the date column to a date object. Finally we select the columns we want to keep.

```
global <- global_cases %>%
  full_join(global_deaths) %>%
  rename("Province_State" = "Province/State", "Country_Region" = "Country/Region") %>%
  mutate(date = mdy(date))
```

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

```
global
```

```
## # A tibble: 330,327 x 5
##    Province_State Country_Region date       cases deaths
##    <chr>          <chr>          <date>     <dbl>  <dbl>
##  1 <NA>           Afghanistan    2020-01-22     0      0
##  2 <NA>           Afghanistan    2020-01-23     0      0
##   3 <NA>           Afghanistan    2020-01-24     0      0
##   4 <NA>           Afghanistan    2020-01-25     0      0
##   5 <NA>           Afghanistan    2020-01-26     0      0
##   6 <NA>           Afghanistan    2020-01-27     0      0
##   7 <NA>           Afghanistan    2020-01-28     0      0
##   8 <NA>           Afghanistan    2020-01-29     0      0
##   9 <NA>           Afghanistan    2020-01-30     0      0
##  10 <NA>           Afghanistan    2020-01-31     0      0
## # i 330,317 more rows
```

Summary Data

Next we will summarize the data to see the number of cases and deaths. If the features are numeric we get the mean, median, min, max, and quartiles. If the features are categorical we get the count of each category. We start with the us data.

```
summary(us)
```

```
##      City           Province_State     Country_Region          date           
##  Length:3819906     Length:3819906     Length:3819906     Min.   :2020-01-22  
##  Class :character   Class :character   Class :character   1st Qu.:2020-11-02  
##  Mode  :character   Mode  :character   Mode  :character   Median :2021-08-15  
##                                                           Mean   :2021-08-15  
##                                                           3rd Qu.:2022-05-28  
##                                                           Max.   :2023-03-09  
##      cases              deaths           Population       Combined_Key      
##  Min.   :  -3073   Min.   :  -82.0   Min.   :       0   Length:3819906     
##  1st Qu.:    330   1st Qu.:    4.0   1st Qu.:    9917   Class :character   
##  Median :   2272   Median :   37.0   Median :   24892   Mode  :character   
##  Mean   :  14088   Mean   :  186.9   Mean   :   99604                      
##  3rd Qu.:   8159   3rd Qu.:  122.0   3rd Qu.:   64979                      
##  Max.   :3710586   Max.   :35545.0   Max.   :10039107                      
```

Next we summarize the global data.

```
summary(global)
```

```
##  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  
```

Filter Data

Next we will filter the data to remove rows where cases are 0. We start with the us data.

```
us <- us %>% dplyr::filter(cases > 0)
summary(us)
```

```
##      City           Province_State     Country_Region          date
##  Length:3474292     Length:3474292     Length:3474292     Min.   :2020-01-22
##  Class :character   Class :character   Class :character   1st Qu.:2020-12-27
##  Mode  :character   Mode  :character   Mode  :character   Median :2021-09-20
##                                                           Mean   :2021-09-19
##                                                           3rd Qu.:2022-06-15
##                                                           Max.   :2023-03-09
##      cases              deaths          Population       Combined_Key
##  Min.   :      1   Min.   :    0.0   Min.   :       0   Length:3474292
##  1st Qu.:    687   1st Qu.:   10.0   1st Qu.:   10953   Class :character
##  Median :   2849   Median :   47.0   Median :   26248   Mode  :character
##  Mean   :  15489   Mean   :  205.1   Mean   :  104502
##  3rd Qu.:   9345   3rd Qu.:  137.0   3rd Qu.:   68098
##  Max.   :3710586   Max.   :35545.0   Max.   :10039107
```

Next we filter the global data.

```
global <- global %>% dplyr::filter(cases > 0)
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
```

Create Combined Key

Next we will create a combined key to uniquely identify each row. We will combine the “Province_State” and “Country_Region” columns to match the us data.

```
global <- global %>%
  unite("Combined_Key", c("Province_State", "Country_Region"), sep = ", ", na.rm = TRUE, remove = FALSE
global
```

```
## # A tibble: 306,827 x 6
##    Combined_Key Province_State Country_Region date       cases deaths
##    <chr>        <chr>          <chr>          <date>     <dbl>  <dbl>
##  1 Afghanistan  <NA>           Afghanistan    2020-02-24     5      0
##  2 Afghanistan  <NA>           Afghanistan    2020-02-25     5      0
##  3 Afghanistan  <NA>           Afghanistan    2020-02-26     5      0
##  4 Afghanistan  <NA>           Afghanistan    2020-02-27     5      0
##  5 Afghanistan  <NA>           Afghanistan    2020-02-28     5      0
##  6 Afghanistan  <NA>           Afghanistan    2020-02-29     5      0
##  7 Afghanistan  <NA>           Afghanistan    2020-03-01     5      0
##  8 Afghanistan  <NA>           Afghanistan    2020-03-02     5      0
##  9 Afghanistan  <NA>           Afghanistan    2020-03-03     5      0
## 10 Afghanistan  <NA>           Afghanistan    2020-03-04     5      0
## # i 306,817 more rows
```

Fetch Population

Next we will fetch the population data from the UID_ISO_FIPS_LookUp_Table. We will join the population data to the global data only because the us data already has the population data.

```
url <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/UID_ISO_FIPS
uid <- read_csv(url) %>%
  select(-c(Lat, Long_, Combined_Key, code3, iso2, iso3, Admin2))
```

```
## Rows: 4321 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (7): iso2, iso3, FIPS, Admin2, Province_State, Country_Region, Combined_Key
## dbl (5): UID, code3, Lat, Long_, Population
##
## 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.
```

```
uid
```

```
## # A tibble: 4,321 x 5
##      UID FIPS  Province_State Country_Region      Population
##    <dbl> <chr> <chr>          <chr>                    <dbl>
##  1     4 <NA>  <NA>           Afghanistan           38928341
##  2     8 <NA>  <NA>           Albania                2877800
##  3    10 <NA>  <NA>           Antarctica                  NA
##  4    12 <NA>  <NA>           Algeria               43851043
##  5    20 <NA>  <NA>           Andorra                  77265
##  6    24 <NA>  <NA>           Angola                32866268
##  7    28 <NA>  <NA>           Antigua and Barbuda      97928
##  8    32 <NA>  <NA>           Argentina             45195777
##  9    51 <NA>  <NA>           Armenia                2963234
## 10    40 <NA>  <NA>           Austria                9006400
## # i 4,311 more rows
```

Next we join the population data to the global data.

```r
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")
global
```

```
## # A tibble: 306,827 x 7
##    Province_State Country_Region date       cases deaths Population Combined_Key
##    <chr>          <chr>          <date>     <dbl>  <dbl>      <dbl> <chr>
##  1 <NA>           Afghanistan    2020-02-24     5      0   38928341 Afghanistan
##  2 <NA>           Afghanistan    2020-02-25     5      0   38928341 Afghanistan
##  3 <NA>           Afghanistan    2020-02-26     5      0   38928341 Afghanistan
##  4 <NA>           Afghanistan    2020-02-27     5      0   38928341 Afghanistan
##  5 <NA>           Afghanistan    2020-02-28     5      0   38928341 Afghanistan
##  6 <NA>           Afghanistan    2020-02-29     5      0   38928341 Afghanistan
##  7 <NA>           Afghanistan    2020-03-01     5      0   38928341 Afghanistan
##  8 <NA>           Afghanistan    2020-03-02     5      0   38928341 Afghanistan
##  9 <NA>           Afghanistan    2020-03-03     5      0   38928341 Afghanistan
## 10 <NA>           Afghanistan    2020-03-04     5      0   38928341 Afghanistan
## # i 306,817 more rows
```

visualize data

Next we will visualize the data. We will start with the us data. We will plot the number of cases and deaths over time. We create new columns "cases_per_million" and "deaths_per_million" to normalize the data by population. We are also grouping the us data by Province_State, Country_Region, and date.

```r
us_by_state <- us %>%
  group_by(Province_State, Country_Region, date) %>%
  summarize(cases = sum(cases), deaths = sum(deaths), Population = sum(Population)) %>%
  mutate(cases_per_million = cases * 1000000 / Population, deaths_per_million = deaths * 1000000 / Popul
  select(Province_State, Country_Region, date, cases, deaths, cases_per_million, deaths_per_million, Po
  ungroup()
```

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

```r
us_by_state
```

```
## # A tibble: 63,216 x 8
##    Province_State Country_Region date       cases deaths cases_per_million
##    <chr>          <chr>          <date>     <dbl>  <dbl>             <dbl>
##  1 Alabama        US             2020-03-11     3      0              5.61
##  2 Alabama        US             2020-03-12     4      0              4.41
##  3 Alabama        US             2020-03-13     8      0              4.86
##  4 Alabama        US             2020-03-14    15      0              9.10
##  5 Alabama        US             2020-03-15    28      0             12.4
##  6 Alabama        US             2020-03-16    36      0             16.0
##  7 Alabama        US             2020-03-17    51      0             20.8
##  8 Alabama        US             2020-03-18    61      0             24.1
##  9 Alabama        US             2020-03-19    88      0             28.1
## 10 Alabama        US             2020-03-20   115      0             34.5
## # i 63,206 more rows
## # i 2 more variables: deaths_per_million <dbl>, Population <dbl>
```

Next we do something similar for the us totals. We take the us data by state which includes the new normalized metrics for cases per million and deaths per million, then group by Country_Region and date, and summarize the cases and deaths. Finally we select the columns we want to keep.

```
us_totals <- us_by_state %>%
  group_by(Country_Region, date) %>%
  summarize(cases = sum(cases), deaths = sum(deaths), Population = sum(Population)) %>%
  mutate(cases_per_million = cases * 1000000 / Population, deaths_per_million = deaths * 1000000 / Popul
  select(Country_Region, date, cases, deaths, cases_per_million, deaths_per_million, Population) %>%
  ungroup()
```

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

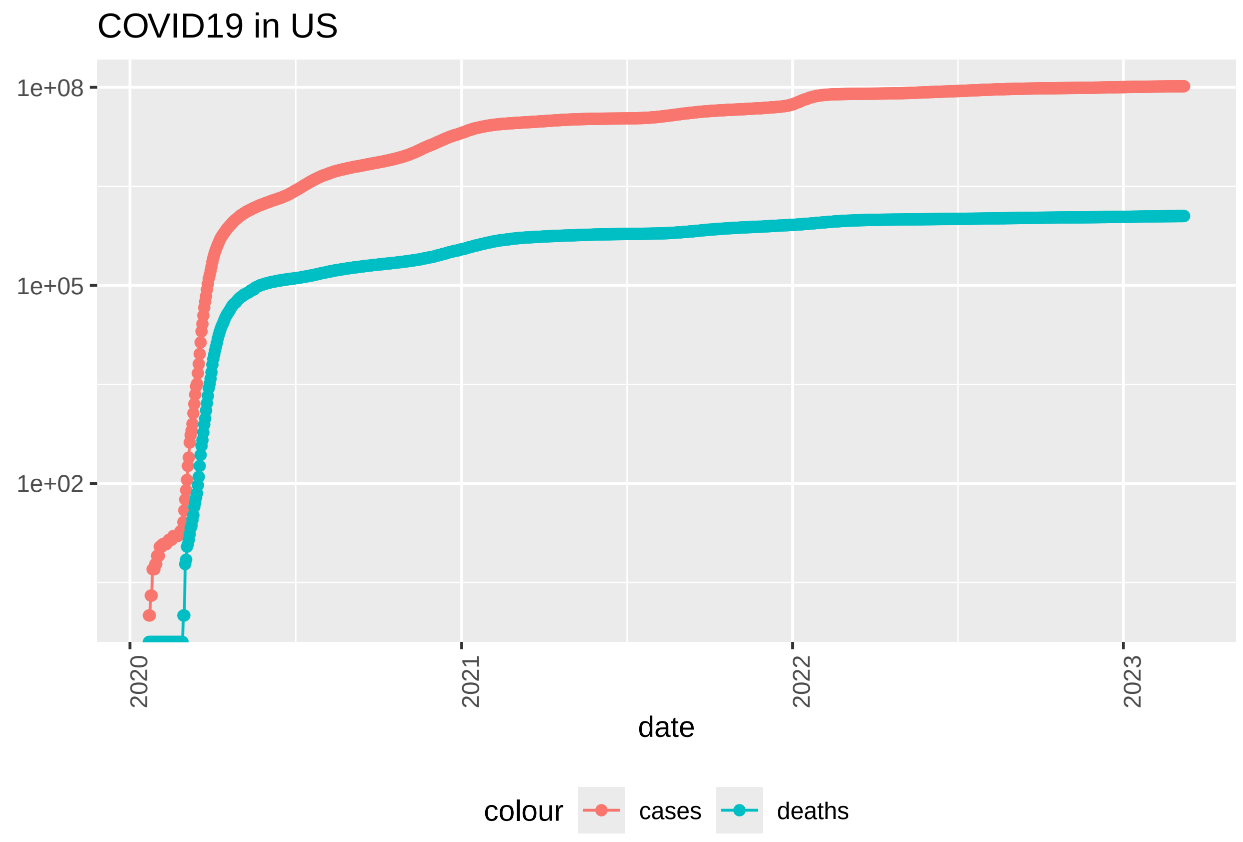
```
us_totals
```

```
## # A tibble: 1,143 x 7
##    Country_Region date       cases deaths cases_per_million deaths_per_million
##    <chr>          <date>     <dbl>  <dbl>             <dbl>              <dbl>
##  1 US             2020-01-22     1      0             0.444                  0
##  2 US             2020-01-23     1      0             0.444                  0
##  3 US             2020-01-24     2      0             0.270                  0
##  4 US             2020-01-25     2      0             0.270                  0
##  5 US             2020-01-26     5      0             0.199                  0
##  6 US             2020-01-27     5      0             0.199                  0
##  7 US             2020-01-28     5      0             0.199                  0
##  8 US             2020-01-29     6      0             0.232                  0
##  9 US             2020-01-30     6      0             0.232                  0
## 10 US             2020-01-31     8      0             0.287                  0
## # i 1,133 more rows
## # i 1 more variable: Population <dbl>
```

Next we plot the us data. We plot the number of cases and deaths over time. We use a log scale for the y axis to better visualize the data. We can see that the number of cases and deaths are increasing over time and have a similar trend.

```
us_totals %>%
  dplyr::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)
```

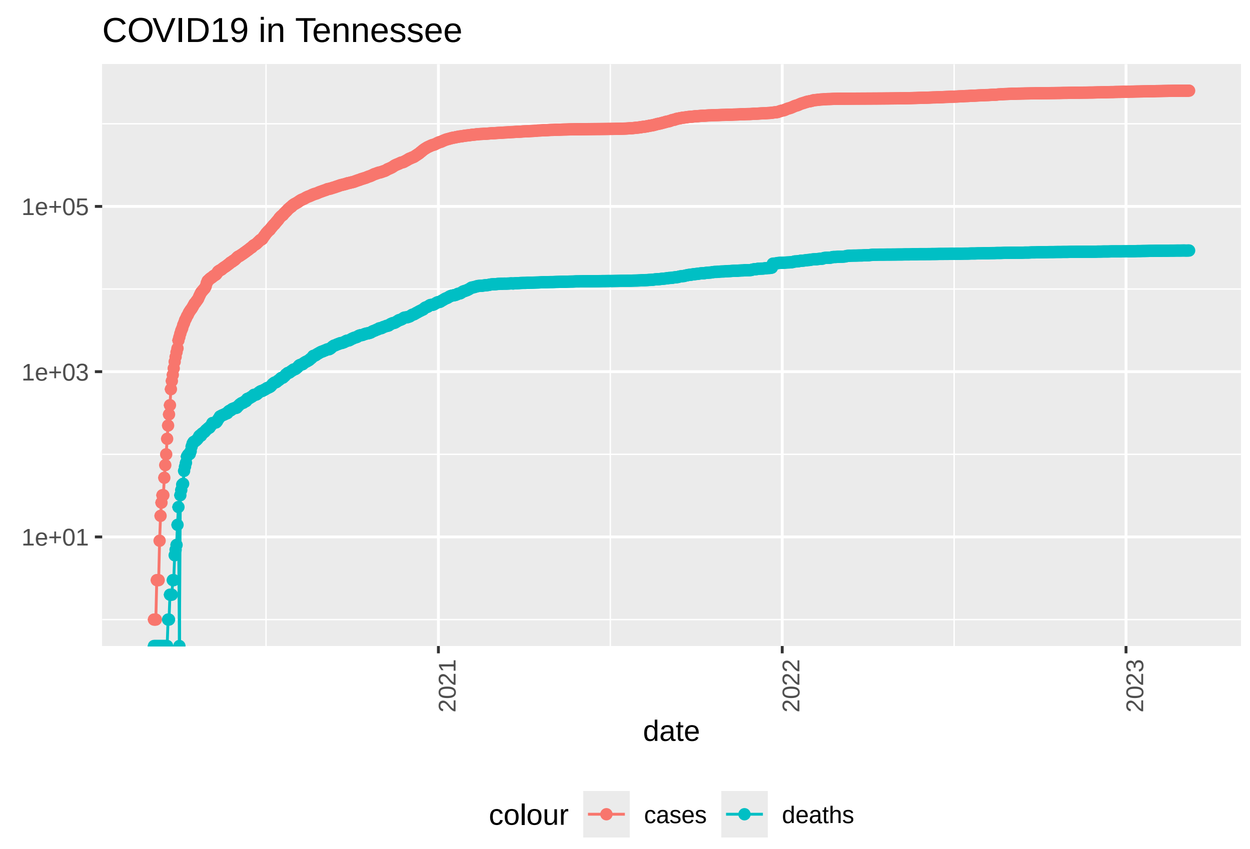
```
## Warning in scale_y_log10(): log-10 transformation introduced infinite values.
## log-10 transformation introduced infinite values.
```



Next we filter down to just one state Tennessee and plot the data. We can see that the number of cases and deaths are increasing over time and have a similar trend. We can also see the trends are similar to the us total data. This would be an indication that the state of Tennessee is following the same trend as the US as a whole and is not an outlier, either higher or lower in rates.

```r
state <- "Tennessee"
us_by_state %>%
  dplyr::filter(Province_State == state) %>%
  dplyr::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)
```

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



Here we check for the maximum date, cases, and deaths in the us data.

```
max(us_totals$date)
```

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

```
max(us_totals$cases)
```

```
## [1] 103802702
```

```
max(us_totals$deaths)
```

```
## [1] 1122724
```

Analyze Data

Next we will analyze the data. We will start by creating new features for new cases and new deaths. This will be the difference between the current and previous day.

```
us <- us %>%
  mutate(new_cases = cases - dplyr::lag(cases), new_deaths = deaths - dplyr::lag(deaths))
us_by_state <- us_by_state %>%
  mutate(new_cases = cases - dplyr::lag(cases), new_deaths = deaths - dplyr::lag(deaths))
us_totals <- us_totals %>%
  mutate(new_cases = cases - dplyr::lag(cases), new_deaths = deaths - dplyr::lag(deaths))
```

We can validate below that the new cases and new deaths are being calculated correctly for the us state data by city.

```
us
```

```
## # A tibble: 3,474,292 x 10
##    City    Province_State Country_Region date       cases deaths Population
##    <chr>   <chr>          <chr>          <date>     <dbl>  <dbl>      <dbl>
##  1 Autauga Alabama        US             2020-03-24     1      0      55869
##  2 Autauga Alabama        US             2020-03-25     5      0      55869
##  3 Autauga Alabama        US             2020-03-26     6      0      55869
##  4 Autauga Alabama        US             2020-03-27     6      0      55869
##  5 Autauga Alabama        US             2020-03-28     6      0      55869
##  6 Autauga Alabama        US             2020-03-29     6      0      55869
##  7 Autauga Alabama        US             2020-03-30     8      0      55869
##  8 Autauga Alabama        US             2020-03-31     8      0      55869
##  9 Autauga Alabama        US             2020-04-01    10      0      55869
## 10 Autauga Alabama        US             2020-04-02    12      0      55869
## # i 3,474,282 more rows
## # i 3 more variables: Combined_Key <chr>, new_cases <dbl>, new_deaths <dbl>
```

We can validate below that the new cases and new deaths are being calculated correctly for the us state totals.

```
us_by_state
```

```
## # A tibble: 63,216 x 10
##    Province_State Country_Region date       cases deaths cases_per_million
##    <chr>          <chr>          <date>     <dbl>  <dbl>             <dbl>
##  1 Alabama        US             2020-03-11     3      0              5.61
##  2 Alabama        US             2020-03-12     4      0              4.41
##  3 Alabama        US             2020-03-13     8      0              4.86
##  4 Alabama        US             2020-03-14    15      0              9.10
##  5 Alabama        US             2020-03-15    28      0             12.4
##  6 Alabama        US             2020-03-16    36      0             16.0
##  7 Alabama        US             2020-03-17    51      0             20.8
##  8 Alabama        US             2020-03-18    61      0             24.1
##  9 Alabama        US             2020-03-19    88      0             28.1
## 10 Alabama        US             2020-03-20   115      0             34.5
## # i 63,206 more rows
## # i 4 more variables: deaths_per_million <dbl>, Population <dbl>,
## #   new_cases <dbl>, new_deaths <dbl>
```

We can validate below that the new cases and new deaths are being calculated correctly for the us totals.

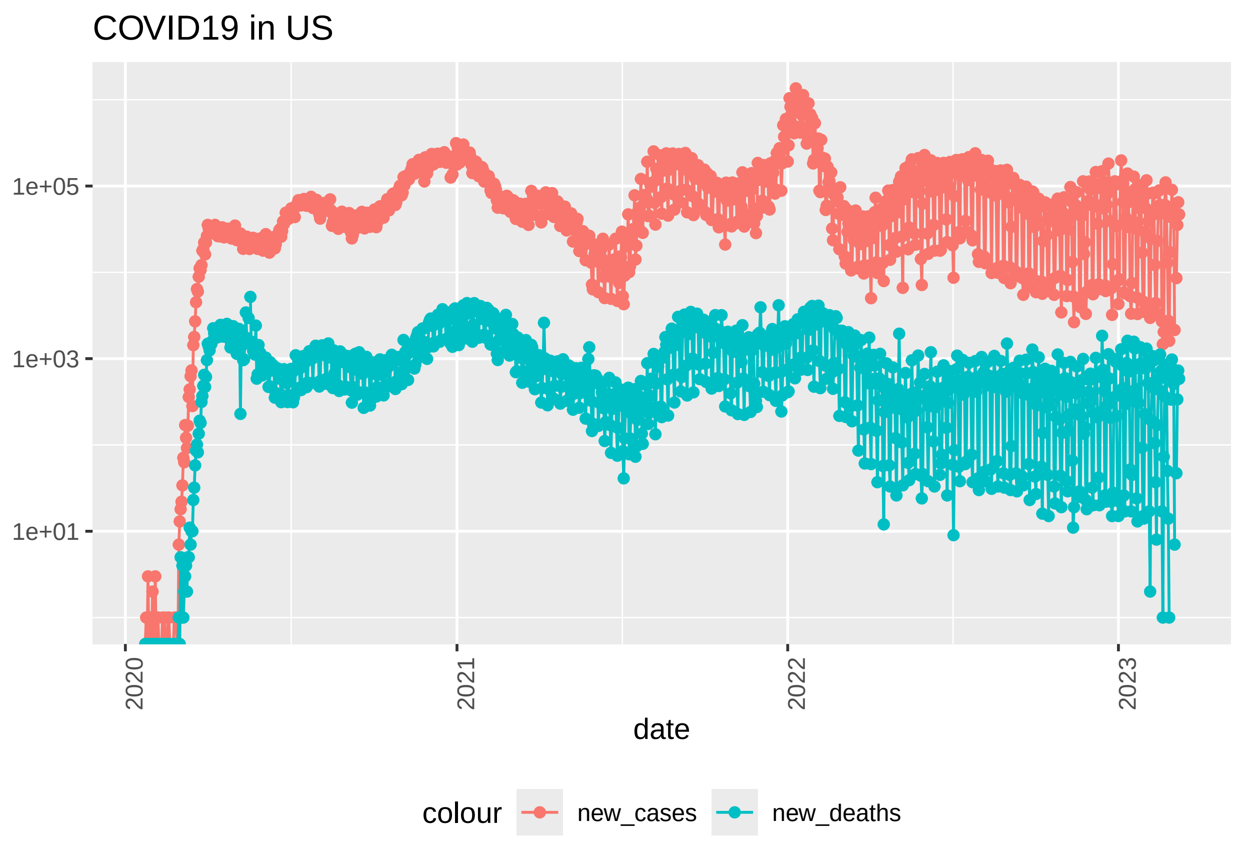
```
us_totals
```

```
## # A tibble: 1,143 x 9
##    Country_Region date       cases deaths cases_per_million deaths_per_million
##    <chr>          <date>     <dbl>  <dbl>             <dbl>              <dbl>
##  1 US             2020-01-22     1      0             0.444                  0
##  2 US             2020-01-23     1      0             0.444                  0
##  3 US             2020-01-24     2      0             0.270                  0
##  4 US             2020-01-25     2      0             0.270                  0
##  5 US             2020-01-26     5      0             0.199                  0
##  6 US             2020-01-27     5      0             0.199                  0
##  7 US             2020-01-28     5      0             0.199                  0
##  8 US             2020-01-29     6      0             0.232                  0
##  9 US             2020-01-30     6      0             0.232                  0
## 10 US             2020-01-31     8      0             0.287                  0
## # i 1,133 more rows
## # i 3 more variables: Population <dbl>, new_cases <dbl>, new_deaths <dbl>
```

Next we will calculate the new cases and new deaths per thousand for the us totals.

```
us_totals %>%
  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 = "COVID19 in US", y = NULL)
```

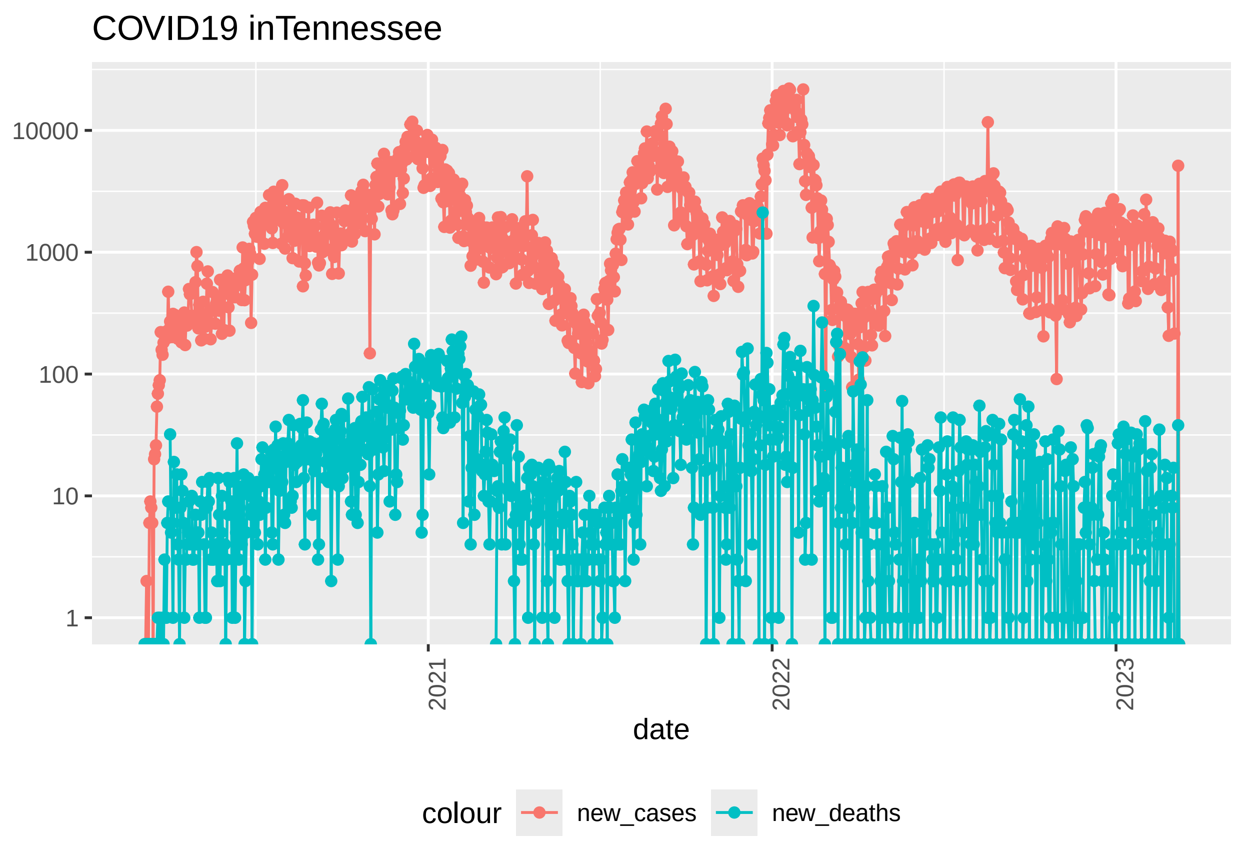
```
## Warning in transformation$transform(x): NaNs produced

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

## Warning in transformation$transform(x): NaNs produced

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

## Warning in transformation$transform(x): NaNs produced

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

## Warning in transformation$transform(x): NaNs produced

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

## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).

## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_point()`).
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).

## Warning: Removed 7 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



We do the same for us by state and filter for Tennessee.

```
state <- "Tennessee"
us_by_state %>%
  dplyr::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)
```

```
## Warning in transformation$transform(x): NaNs produced

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

## Warning in transformation$transform(x): NaNs produced
## Warning in scale_y_log10(): log-10 transformation introduced infinite values.

## Warning in transformation$transform(x): NaNs produced

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

## Warning in transformation$transform(x): NaNs produced

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

## Warning: Removed 1 row containing missing values or values outside the scale range
## (‘geom_line()‘).

## Warning: Removed 2 rows containing missing values or values outside the scale range
## (‘geom_point()‘).

## Warning: Removed 1 row containing missing values or values outside the scale range
## (‘geom_line()‘).

## Warning: Removed 14 rows containing missing values or values outside the scale range
## (‘geom_point()‘).
```



Next we will summarize the us state totals, and filter for only cases and population greater than 0.

```r
us_state_totals <- us_by_state %>%
  group_by(Province_State) %>%
  summarize(cases = max(cases), deaths = max(deaths), Population = max(Population), cases_per_thousand =
  dplyr::filter(cases > 0, Population > 0) %>%
  ungroup()
```

Here we inspect the top 10 min deaths per thousand.

```r
us_state_totals %>%
  slice_min(deaths_per_thousand, n = 10)
```

```
## # A tibble: 10 x 6
##    Province_State             cases deaths Population cases_per_thousand
##    <chr>                      <dbl>  <dbl>      <dbl>              <dbl>
##  1 American Samoa              8320     34      55641               150.
##  2 Northern Mariana Islands   13666     41      55144               248.
##  3 Virgin Islands             24813    130     107268               231.
##  4 Hawaii                    380608   1841    1415872               269.
##  5 Vermont                   152618    929     623989               245.
##  6 Puerto Rico              1101469   5823    3754939               293.
##  7 Utah                     1090346   5298    2785478               391.
##  8 District of Columbia      177945   1432     705749               252.
##  9 Alaska                    307655   1486     728809               422.
## 10 Washington               1928913  15683    7614893               253.
## # i 1 more variable: deaths_per_thousand <dbl>
```

Here we inspect the top 10 min deaths per thousand and select the columns we want to keep.

```r
us_state_totals %>%
  slice_min(deaths_per_thousand, n = 10) %>%
  select(deaths_per_thousand, cases_per_thousand, everything())
```

```
## # A tibble: 10 x 6
##    deaths_per_thousand cases_per_thousand Province_State            cases deaths
##                  <dbl>              <dbl> <chr>                     <dbl>  <dbl>
##  1               0.611               150. American Samoa           8.32e3     34
##  2               0.744               248. Northern Mariana Islands 1.37e4     41
##  3               1.21                231. Virgin Islands           2.48e4    130
##  4               1.30                269. Hawaii                   3.81e5   1841
##  5               1.49                245. Vermont                  1.53e5    929
##  6               1.55                293. Puerto Rico              1.10e6   5823
##  7               1.90                391. Utah                     1.09e6   5298
##  8               2.03                252. District of Columbia     1.78e5   1432
##  9               2.04                422. Alaska                   3.08e5   1486
## 10               2.06                253. Washington               1.93e6  15683
## # i 1 more variable: Population <dbl>
```

Here we inspect the top 10 max deaths per thousand.

```r
us_state_totals %>%
  slice_max(deaths_per_thousand, n = 10)
```

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

Here we inspect the top 10 max deaths per thousand and select the columns we want to keep.

```r
us_state_totals %>%
  slice_max(deaths_per_thousand, n = 10) %>%
  select(deaths_per_thousand, cases_per_thousand, everything())
```

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

Model Data

Next we will model the data. We will start by creating a linear regression model to predict deaths per thousand based on cases per thousand. We will use the us state totals data for this analysis. Lastly we will summarize the model's performance.

```r
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
## -2.2394 -0.6114  0.1965  0.6413  1.2413
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -0.02599    0.72442  -0.036    0.972
## cases_per_thousand  0.01020    0.00231   4.414 4.89e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8803 on 54 degrees of freedom
## Multiple R-squared:  0.2652, Adjusted R-squared:  0.2516
## F-statistic: 19.49 on 1 and 54 DF,  p-value: 4.894e-05
```

Next we will predict the deaths per thousand based on the cases per thousand.

```
us_state_totals %>% mutate(pred = predict(model))
```

```
## # A tibble: 56 x 7
##    Province_State          cases deaths Population cases_per_thousand
##    <chr>                   <dbl>  <dbl>      <dbl>              <dbl>
##  1 Alabama               1644533  21032    4903185               335.
##  2 Alaska                 307655   1486     728809               422.
##  3 American Samoa           8320     34      55641               150.
##  4 Arizona               2443514  33102    7278717               336.
##  5 Arkansas              1006883  13020    3017804               334.
##  6 California           12129699 101159   39512223               307.
##  7 Colorado              1764401  14181    5758736               306.
##  8 Connecticut            976657  12220    3565287               274.
##  9 Delaware               330793   3324     973764               340.
## 10 District of Columbia   177945   1432     705749               252.
## # i 46 more rows
## # i 2 more variables: deaths_per_thousand <dbl>, pred <dbl>
```

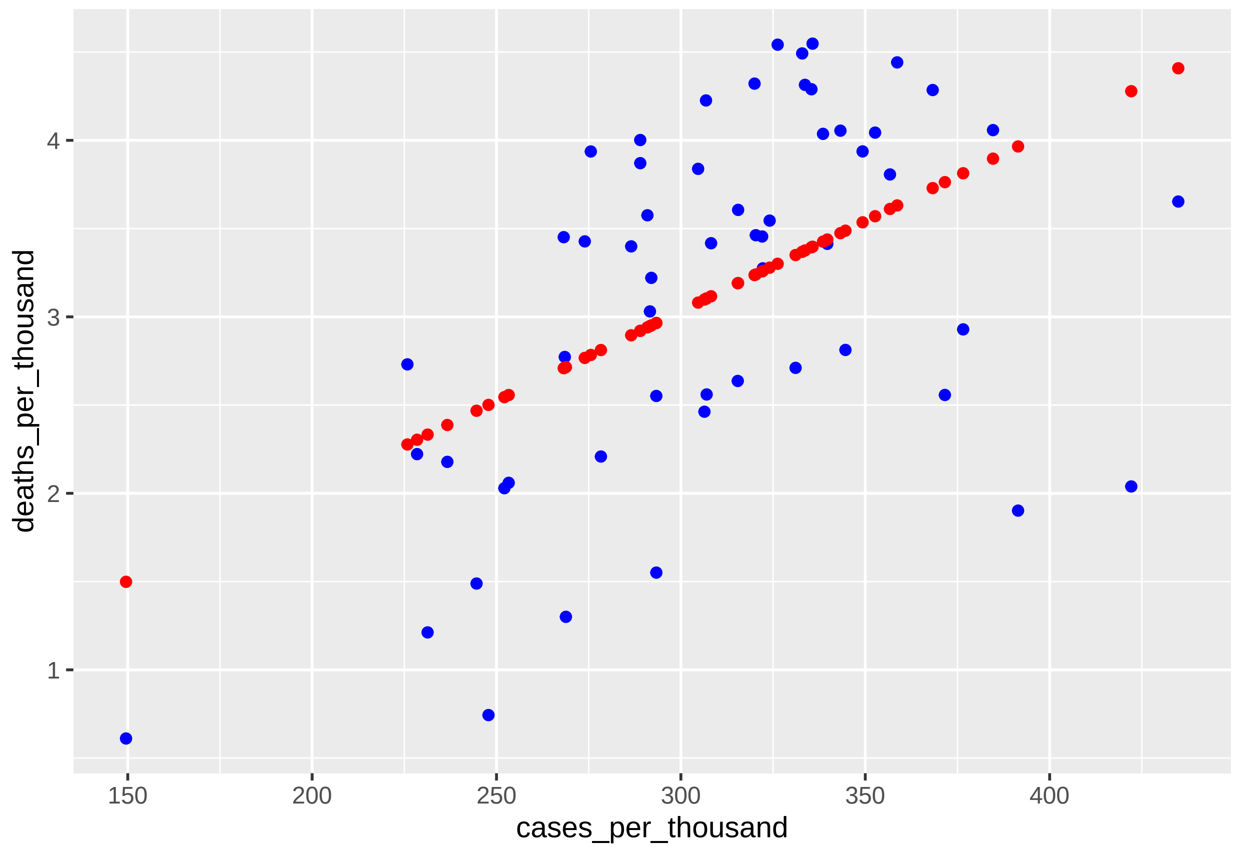
Next we will add the predicted deaths per thousand to the us state totals data.

```
us_totals_w_pred <- us_state_totals %>% mutate(pred = predict(model))
us_totals_w_pred
```

```
## # A tibble: 56 x 7
##    Province_State          cases deaths Population cases_per_thousand
##    <chr>                   <dbl>  <dbl>      <dbl>              <dbl>
##  1 Alabama               1644533  21032    4903185               335.
##  2 Alaska                 307655   1486     728809               422.
##  3 American Samoa           8320     34      55641               150.
##  4 Arizona               2443514  33102    7278717               336.
##  5 Arkansas              1006883  13020    3017804               334.
##  6 California           12129699 101159   39512223               307.
##  7 Colorado              1764401  14181    5758736               306.
##  8 Connecticut            976657  12220    3565287               274.
##  9 Delaware               330793   3324     973764               340.
## 10 District of Columbia   177945   1432     705749               252.
## # i 46 more rows
## # i 2 more variables: deaths_per_thousand <dbl>, pred <dbl>
```

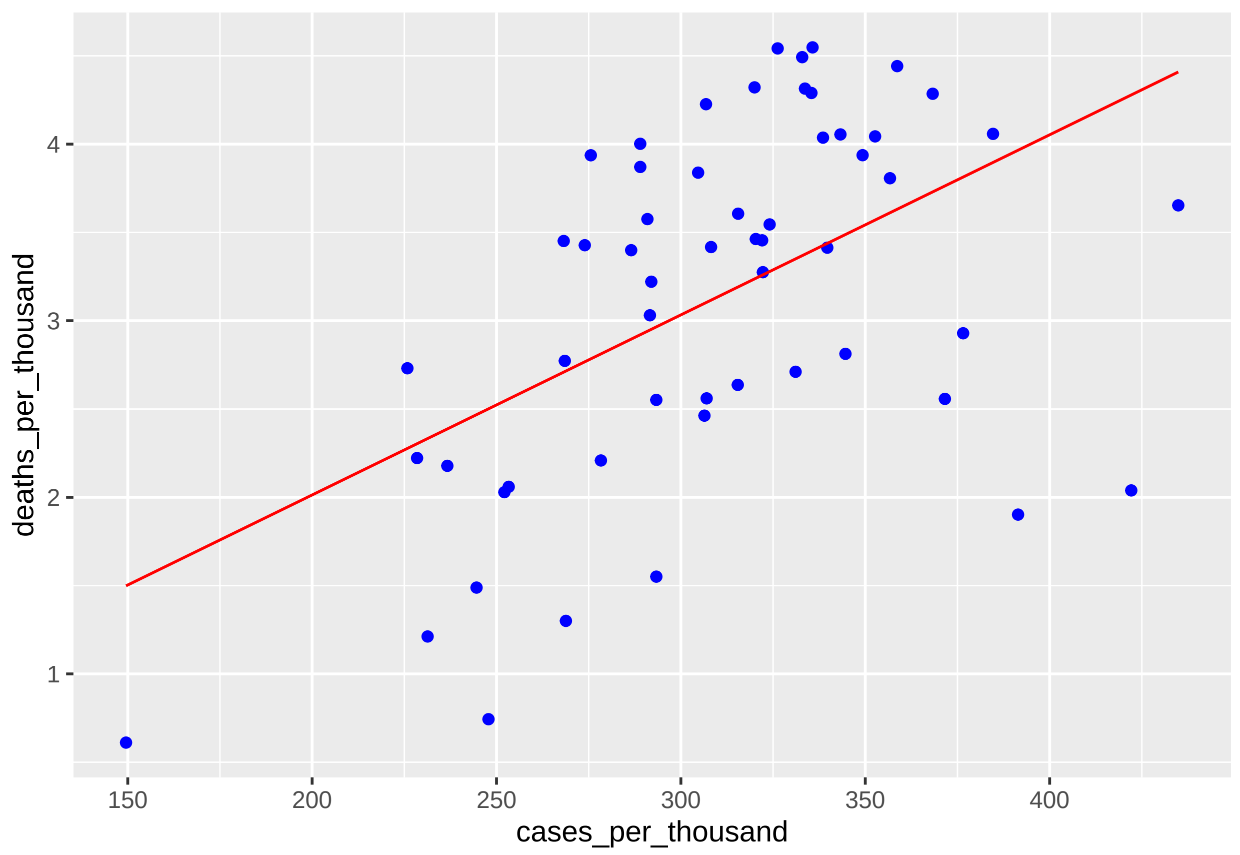
Next we will plot the actual deaths per thousand vs the predicted deaths per thousand. We can see that the predicted deaths per thousand are close to the actual deaths per thousand. The red points are the predicted deaths per thousand and the blue points are the actual deaths per thousand. The blue dots do appear to have a linear relationship with the red dots.

```r
us_totals_w_pred %>% ggplot() +
  geom_point(aes(x = cases_per_thousand, y = deaths_per_thousand), color = "blue") +
  geom_point(aes(x = cases_per_thousand, y = pred), color = "red")
```



Next we create another plot to visualize the model.

```r
us_totals_w_pred %>% ggplot() +
  geom_point(aes(x = cases_per_thousand, y = deaths_per_thousand), color = "blue") +
  geom_line(aes(x = cases_per_thousand, y = pred), color = "red")
```



Next we summarize the model's performance using RMSE and R2.

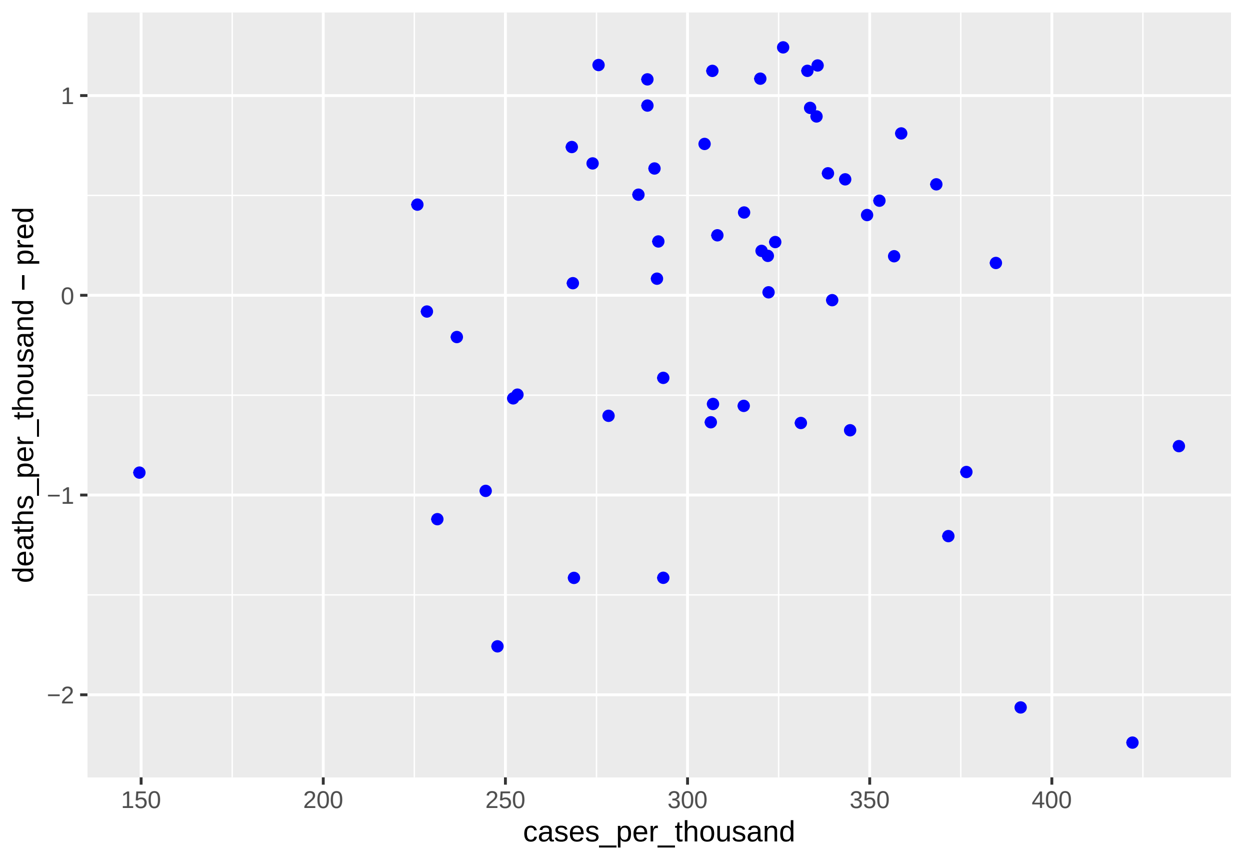
```
us_totals_w_pred %>% summarize(rmse = sqrt(mean((deaths_per_thousand - pred)^2)), r2 = cor(deaths_per_tl
```

```
## # A tibble: 1 x 2
##    rmse    r2
##   <dbl> <dbl>
## 1 0.864 0.265
```

Above we can see that the RMSE is 0.8644585 and the R2 is 0.2651695. The RMSE is the square root of the mean of the squared differences between the actual and predicted deaths per thousand. The R2 is the square of the correlation between the actual and predicted deaths per thousand. The RMSE is a measure of the model's accuracy and the R2 is a measure of the model's goodness of fit. An RMSE of 0.8644585 means that the model's predictions are on average 0.8644585 deaths per thousand away from the actual deaths per thousand. An R2 of 0.2651695 means that the model explains 26.52% of the variance in the deaths per thousand.

Here we plot the residuals which are the difference between the actual deaths per thousand and the predicted deaths per thousand. We can see that the residuals are randomly distributed and there is no clear pattern. This is a good indication that the model is a good fit for the data.

```
us_totals_w_pred %>% ggplot() +
  geom_point(aes(x = cases_per_thousand, y = deaths_per_thousand - pred), color = "blue")
```



Conclusion

In conclusion, we have analyzed the COVID 19 data from Johns Hopkins University. We have cleaned the data, visualized the data, created new features, and modeled the data. We then created a linear regression model to predict deaths per thousand based on cases per thousand. We evaluated the model using RMSE which is a measure of the model's accuracy and with R2 which is a measure of the model's goodness of fit. An RMSE of 0.8644585 means that the model's predictions are on average 0.8644585 deaths per thousand away from the actual deaths per thousand. An R2 of 0.2651695 means that the model explains 26.52% of the variance in the deaths per thousand. The residuals are also randomly distributed and there is no clear pattern, which is a good indication that the model is a good fit for the data. Some bias that could be present is in the data itself, as the data is from Johns Hopkins University and may not be representative of the entire population. We also do not have enough information to determine how positive cases and positive deaths are being reported, or under reported. Therefore, we should be cautious in interpreting the results and making predictions until we have more information. However, this model seems to be a good fit for the data is making accurate predictions for deaths per thousand based on cases per thousand in the US state totals data.