# Covid-19 Report

Saikat Sengupta

2023-05-02

## Introduction

Our task for the DTSA 5301: Data Science as a Field course is to showcase our proficiency in executing all stages of the data science process. We will achieve this by generating a replicable report based on the COVID19 dataset obtained from the John Hopkins GitHub repository.

### Questions asked in this project

What is the Wisconsin county with the highest COVID19 mortality rate, and what is the Wisconsin county with the lowest COVID19 mortality rate? Additionally, can we employ a Linear Regression Model to forecast future COVID19 cases and deaths in Wisconsin?

## Step 0: Import Libraries

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(lubridate)
library(ggplot2)
library(dplyr)
```

## Step 2:Import and Describe the Dataset

```r
# All files begin with this string.
url_in <- ('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_co
# Vector containing four file names.
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)


global_cases <- read_csv(urls[1])
```

```
## Rows: 289 Columns: 1147
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr    (2): Province/State, Country/Region
## dbl (1145): Lat, Long, 1/22/20, 1/23/20, 1/24/20, 1/25/20, 1/26/20, 1/27/20,...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
global_deaths <- read_csv(urls[2])
```

```
## Rows: 289 Columns: 1147
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr    (2): Province/State, Country/Region
## dbl (1145): Lat, Long, 1/22/20, 1/23/20, 1/24/20, 1/25/20, 1/26/20, 1/27/20,...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
US_cases <- read_csv(urls[3])
```

```
## Rows: 3342 Columns: 1154
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr    (6): iso2, iso3, Admin2, Province_State, Country_Region, Combined_Key
## dbl (1148): UID, code3, FIPS, Lat, Long_, 1/22/20, 1/23/20, 1/24/20, 1/25/20...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
US_deaths <- read_csv(urls[4])
```

```
## Rows: 3342 Columns: 1155
## -- Column specification --------------------------------------------------------
## Delimiter: ","
```

```
## chr    (6): iso2, iso3, Admin2, Province_State, Country_Region, Combined_Key
## dbl (1149): UID, code3, FIPS, Lat, Long_, Population, 1/22/20, 1/23/20, 1/24...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

## Step 3: Tidy and Transform the Dataset

```
global_cases <- global_cases %>%
  pivot_longer(cols =
                 -c('Province/State',
                    'Country/Region', Lat, Long),
               names_to = "date",
               values_to = "cases")
global_deaths <- global_deaths %>%
  pivot_longer(cols =
                 -c('Province/State',
                    'Country/Region', Lat, Long),
               names_to = "date",
               values_to = "deaths")
 global <- global_cases %>%
  full_join(global_deaths) %>%
    rename(Country_Region = 'Country/Region',
       Province_State = 'Province/State') %>%
  mutate(date = mdy(date))
```

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

```
summary(global)
```

```
##  Province_State     Country_Region          Lat               Long
##  Length:330327      Length:330327      Min.   :-71.950   Min.   :-178.12
##  Class :character   Class :character   1st Qu.:  3.934   1st Qu.: -42.60
##  Mode  :character   Mode  :character   Median : 21.513   Median :  20.94
##                                        Mean   : 19.719   Mean   :  22.18
##                                        3rd Qu.: 40.464   3rd Qu.:  90.36
##                                        Max.   : 71.707   Max.   : 178.06
##                                        NA's   :2286      NA's   :2286
##       date                cases               deaths
##  Min.   :2020-01-22   Min.   :        0   Min.   :      0
##  1st Qu.:2020-11-02   1st Qu.:      680   1st Qu.:      3
##  Median :2021-08-15   Median :    14429   Median :    150
##  Mean   :2021-08-15   Mean   :   959384   Mean   :  13380
##  3rd Qu.:2022-05-28   3rd Qu.:   228517   3rd Qu.:   3032
##  Max.   :2023-03-09   Max.   :103802702   Max.   :1123836
##
```

```
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_))

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_))

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

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

```
summary(US)
```

```
##     Admin2          Province_State     Country_Region     Combined_Key
##  Length:3819906     Length:3819906     Length:3819906     Length:3819906
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##       date                cases           Population          deaths
##  Min.   :2020-01-22   Min.   : -3073   Min.   :       0   Min.   :  -82.0
##  1st Qu.:2020-11-02   1st Qu.:   330   1st Qu.:    9917   1st Qu.:    4.0
##  Median :2021-08-15   Median :  2272   Median :   24892   Median :   37.0
##  Mean   :2021-08-15   Mean   : 14088   Mean   :   99604   Mean   :  186.9
##  3rd Qu.:2022-05-28   3rd Qu.:  8159   3rd Qu.:   64979   3rd Qu.:  122.0
##  Max.   :2023-03-09   Max.   :3710586   Max.   :10039107   Max.   :35545.0
```

## Step 4: Visualization and Analysis of the Dataset

```
# Filter US dataset for only the rows where Province_State is Wisconsin.
wisc <- US %>%
  filter(Province_State == "Wisconsin", cases > 0) %>%
  group_by(date, Admin2)
# Group Wisconsin data by county and add mortality rate column.
wisc_counties <- wisc %>%
  group_by(Admin2, date) %>%
  mutate(mortality_rate = deaths / cases) %>%
  select(Admin2, date, cases, deaths, Population, mortality_rate)
```

```r
# Sum all Wisconsin county cases, deaths, and populations.
wisc_totals <- wisc %>%
  group_by(date) %>%
  summarize(cases = sum(cases), deaths = sum(deaths), Population = sum(Population)) %>%
  select(date, cases, deaths, Population) %>%
  ungroup()
# Create a dataframe that contains the most recent statistics for each Wisconsin county. .
current_counties <- wisc_counties %>%
  filter(date == "2022-04-22") %>%
  group_by(Admin2) %>%
  mutate(county_mortality_rate = deaths/cases) %>%
  select(date, Admin2, cases, deaths, Population, county_mortality_rate) %>%
  ungroup()
```

```r
max(wisc_totals$cases)
```

```
## [1] 2006582
```

```r
max(wisc_totals$deaths) / max(wisc_totals$cases)
```

```
## [1] 0.008160643
```

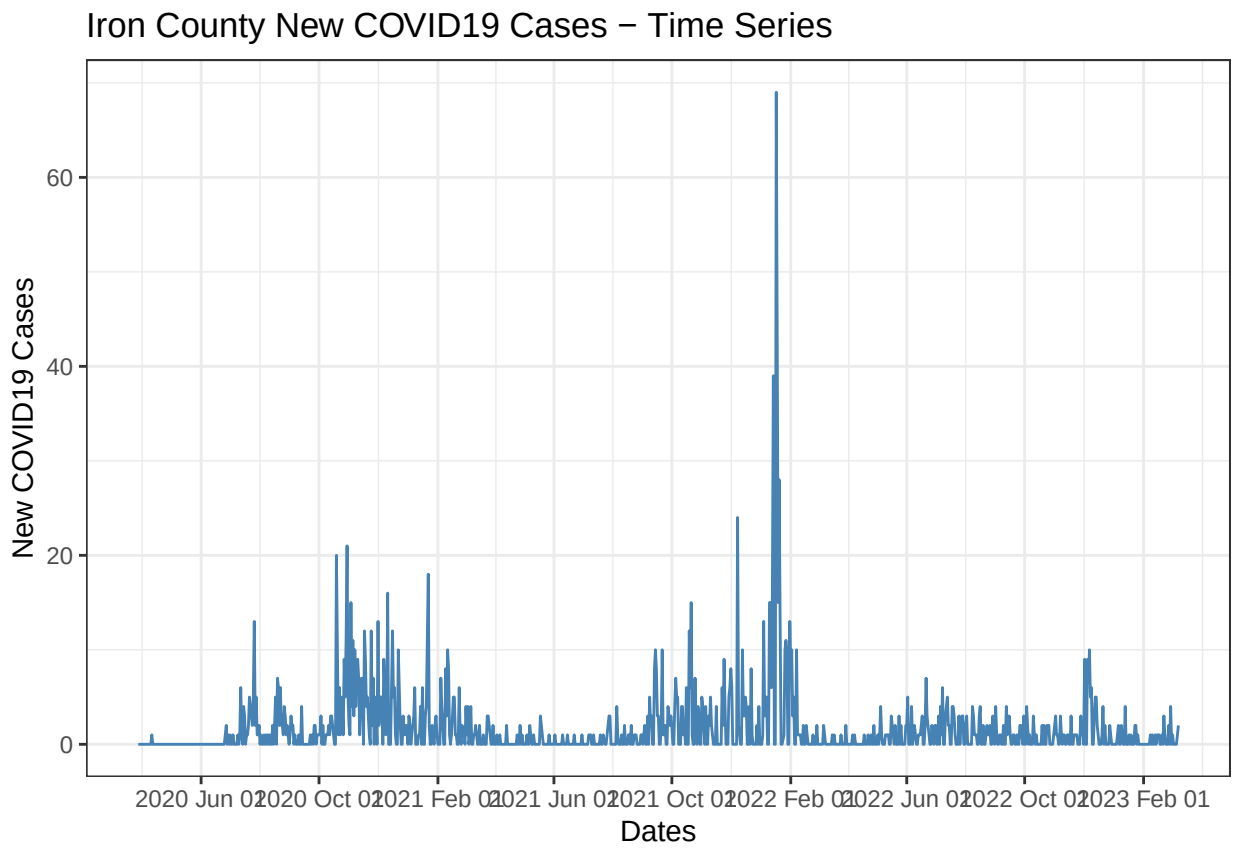
```r
current_counties %>% slice_max(county_mortality_rate)
```

```
## # A tibble: 1 x 6
##   date       Admin2 cases deaths Population county_mortality_rate
##   <date>     <chr>  <dbl>  <dbl>      <dbl>                 <dbl>
## 1 2022-04-22 Iron    1470     47       5687                0.0320
```

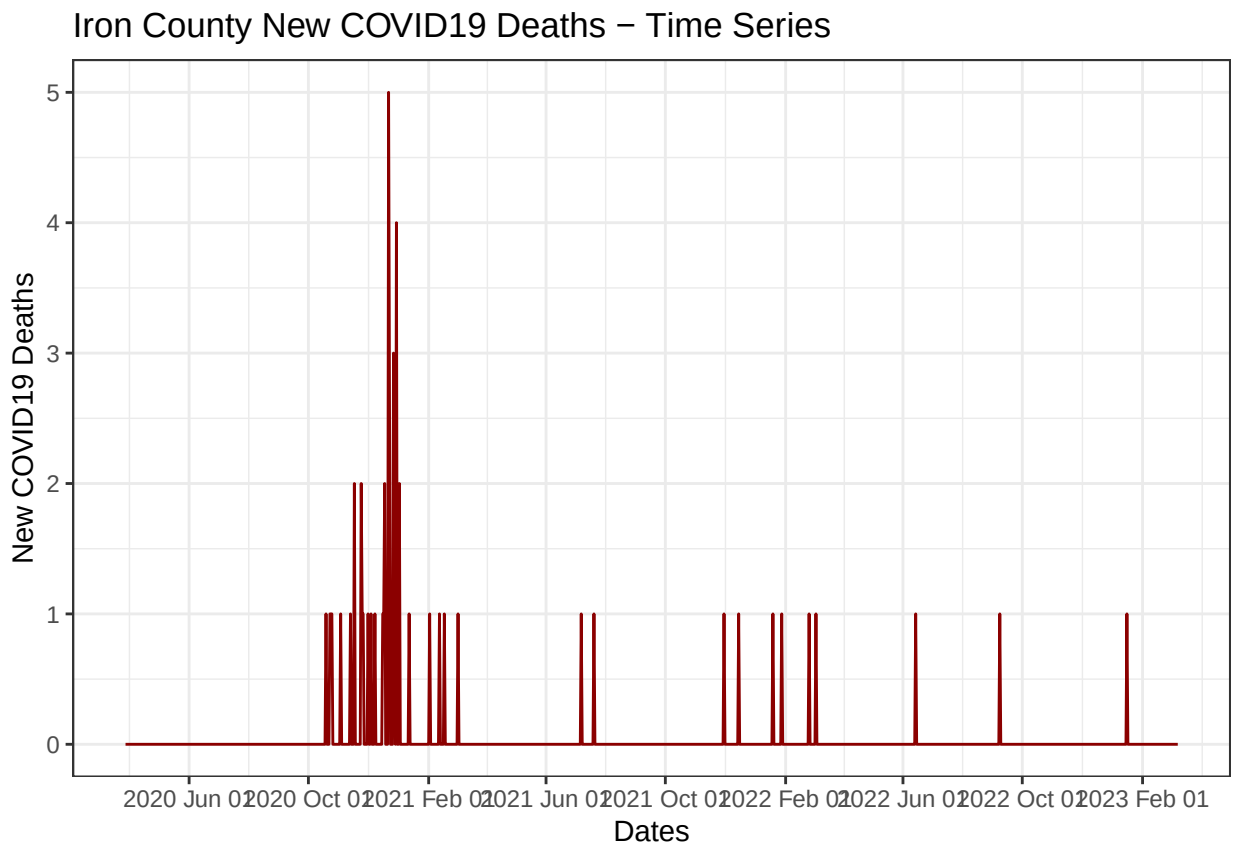
```r
current_counties %>% slice_min(county_mortality_rate)
```

```
## # A tibble: 1 x 6
##   date       Admin2  cases deaths Population county_mortality_rate
##   <date>     <chr>   <dbl>  <dbl>      <dbl>                 <dbl>
## 1 2022-04-22 Buffalo  3531     12      13031               0.00340
```

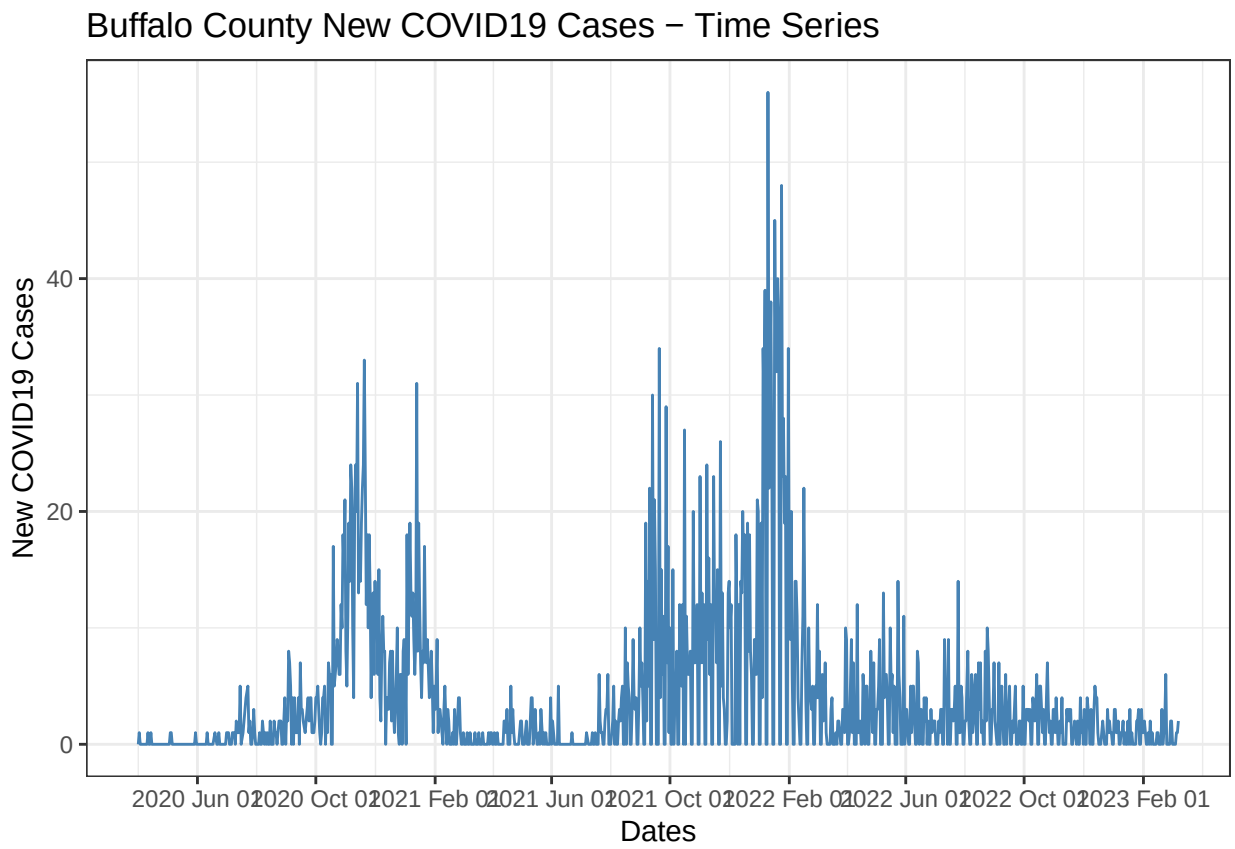
```r
#Create a new dataframe for Iron County and add columns for daily new cases and deaths.
iron_county <- wisc_counties %>%
  filter(Admin2 == "Iron") %>%
  group_by(Admin2) %>%
  mutate(new_cases = cases - lag(cases), new_deaths = deaths - lag(deaths)) %>%
  select(date, Admin2, cases, deaths, Population, new_cases, new_deaths)
iron_county <- iron_county %>%
  filter(new_cases >= 0, new_deaths >=0)
ggplot(iron_county, aes(x=date)) +
  geom_line(aes(y = new_cases), color="steelblue") +
  scale_x_date(date_labels = "%Y %b %d", date_breaks = "4 month") +
  theme_bw() +
  labs(x = "Dates",
       y = "New COVID19 Cases",
       title = "Iron County New COVID19 Cases - Time Series")
```



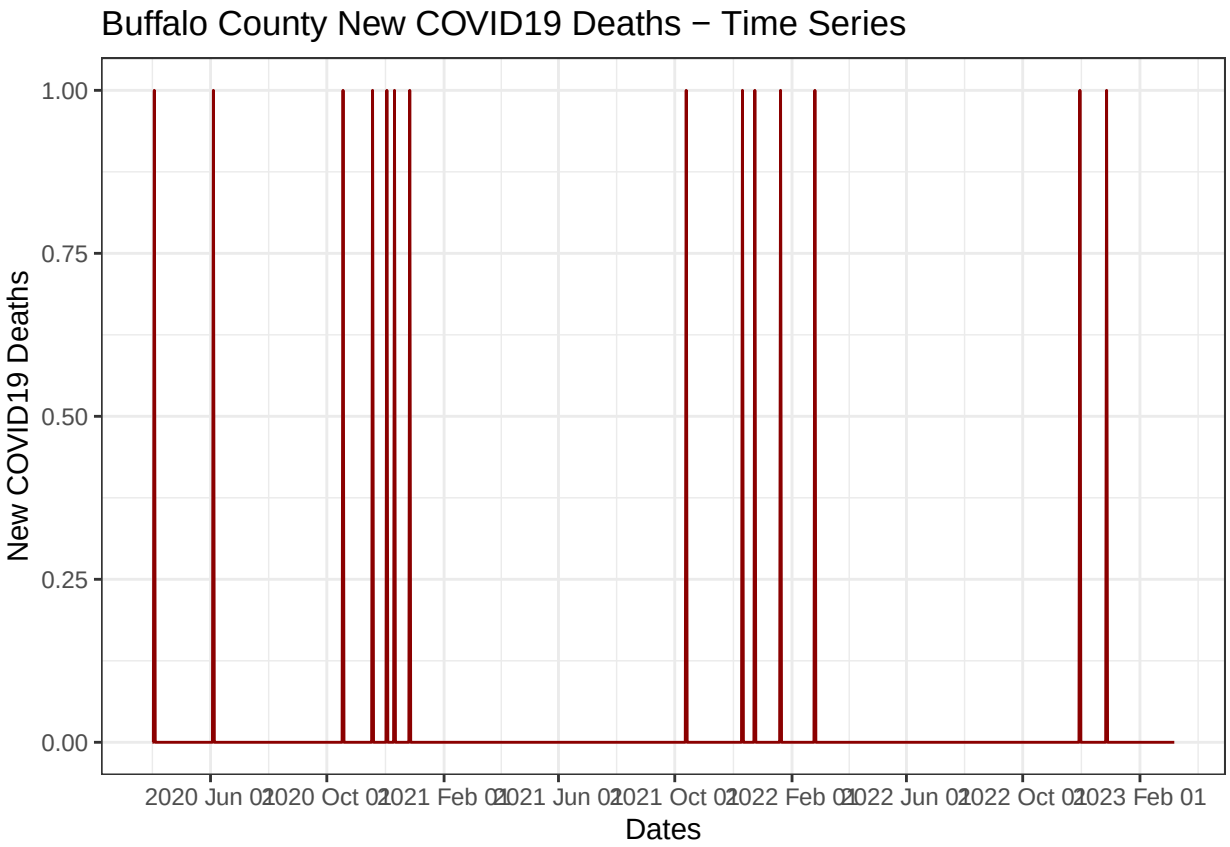

```
ggplot(iron_county, aes(x=date)) +
  geom_line(aes(y = new_deaths), color = "dark red") +
    scale_x_date(date_labels = "%Y %b %d", date_breaks = "4 month") +
  theme_bw() +
  labs(x = "Dates",
       y = "New COVID19 Deaths",
       title = "Iron County New COVID19 Deaths - Time Series")
```




```
# Create a new dataframe for Iron County and add columns for daily new cases and deaths.
buffalo_county <- wisc_counties %>%
  filter(Admin2 == "Buffalo") %>%
  group_by(Admin2) %>%
  mutate(new_cases = cases - lag(cases), new_deaths = deaths - lag(deaths)) %>%
  select(date, Admin2, cases, deaths, Population, new_cases, new_deaths)
buffalo_county <- buffalo_county %>%
  filter(new_cases >= 0, new_deaths >=0)
ggplot(buffalo_county, aes(x=date)) +
  geom_line(aes(y = new_cases), color="steelblue") +
  scale_x_date(date_labels = "%Y %b %d", date_breaks = "4 month") +
  theme_bw() +
  labs(x = "Dates",
       y = "New COVID19 Cases",
       title = "Buffalo County New COVID19 Cases - Time Series")
```




```
ggplot(buffalo_county, aes(x=date)) +
  geom_line(aes(y = new_deaths), color = "dark red") +
    scale_x_date(date_labels = "%Y %b %d", date_breaks = "4 month") +
  theme_bw() +
  labs(x = "Dates",
       y = "New COVID19 Deaths",
       title = "Buffalo County New COVID19 Deaths - Time Series")
```




## Step 5: Bias and Conclusion of the Dataset

### Conclusion

My analysis revealed that Iron County has the highest COVID19 mortality rate in Wisconsin, while Buffalo County has the lowest COVID19 mortality rate in Wisconsin.

### Bias

COVID19 has turned into a highly politicized topic, and expressing a strong opinion on this debate could create a source of bias. I prevented this by remaining impartial and avoiding any presumptions. My primary focus was on the data itself rather than the political environment surrounding the pandemic. While data collection can also introduce bias, the dataset I utilized had detailed documentation about its acquisition and the entities involved. Hence, I feel more confident in using this dataset since it appears to be more reliable. Although there may be some uncertainty about how COVID19 cases were reported, this is a typical issue with any data related to infectious diseases. We should expect such ambiguities and work with the available data as efficiently as possible.