John Hopkins Data

RLReichel

2024-04-20

Libraries

The libraries we will be using include:

- tinytex
- tidyverse
- lubridate

```
if(!file.exists("Data")){dir.create("Data")}
library(tinytex)
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   3.5.1     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 errors
```

```
library(lubridate)
```

These libraries will allow us to import, tidy, transform, visualize, and model the data.

Import COVID-19 Time Data

Here we will import COVID-19 data from the time series from John Hopkins' github page.

First we will get the URLs needed into a list.

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

suffix_list <- c("confirmed_US.csv","confirmed_global.csv","deaths_global.csv","deaths_US.csv")

urls <- str_c(url_in,suffix_list)
#This assembles a list of four URLs pointing to csv files from the John Hopkins COVID-19 datasets.

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

Next we will read in our data.

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

```
## 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.
```

```
global_cases <- 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.
```

```
global_deaths <- read_csv(urls[3])
```

```
## 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.
```

```
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.
```

```
#This reads in four csv files of COVID-19 data from the URLs assembled in the previous chunk.
```

Tidy US and Global Datasets

Next we are going to tidy our data by making each date have its own entry, removing Lat and Long since we don't need it for this analysis, and editing Province/State and Country/Region for easier use while maintaining data integrity.

First we will tidy the global data.

```
global_cases <- global_cases %>%
  pivot_longer(cols = -c("Province/State","Country/Region",Lat,Long),
               names_to = "date",
               values_to = "cases") %>%
  select(-c(Lat,Long))
#This pivots the date data into a longer series of rows, putting the dates into a new "date" column and
```

```
global_deaths <- global_deaths %>%
  pivot_longer(cols = -c("Province/State","Country/Region", Lat, Long),
               names_to = "date",
               values_to = "deaths") %>%
  select(-c(Lat,Long))
#This pivots the date data into a longer series of rows, putting the dates into a new "date" column and

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`, date)`
```

```
#This will fully join global cases and deaths into a global dataframe, renaming the country and provinc

global <- global %>% filter(cases > 0)
#This filters out any days where there were no cases, keeping only when cases were positive.
```

Second, we'll tidy the US data.

```
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_))
#This pivots the dates listed in one entry in the table into multiple entries by date, expanding the ro

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_))
#This pivots the dates listed in one entry in the table into multiple entries by date, expanding the ro

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

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

```
#This joins the deaths and cases in the US into a single dataframe containing both sets of information

global <- global %>%
  unite("Combined_Key",
        c(Province_State,Country_Region),
        sep = ", ",
        na.rm = TRUE,
        remove = FALSE)
#This produces a new column combining the values from the listed columns in the order listed with the s
```

```
uid <- read_csv(uid_lookup_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.
```

```
#This reads in the csv file for the UID look up from a URL provided in a previous code block "URLs for
```

```
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)

#This combines the UID data read in from the csv file with the province and region columns, getting rid
```

Visualizing the Data

Next we are going to visualize the data in a variety of ways to attempt to find patterns and meaning in the data.

Sort US Data by State

First we will sort our data by state in the US in order to visualize deaths and cases by state.

```
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()
```

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

```
#This produces a dataframe where the cases and per state are listed on each date available, with deaths

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()
```

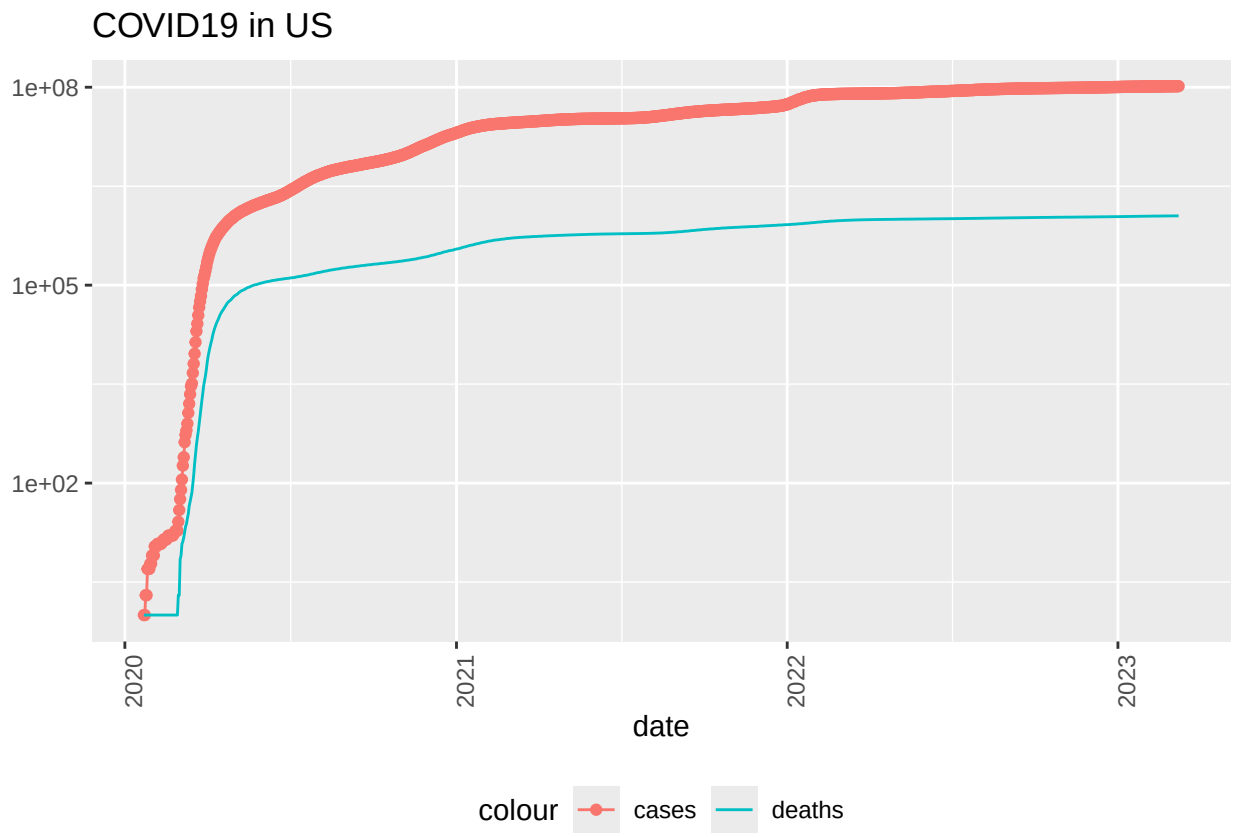
```
## `summarise()` has grouped output by 'Country_Region'. You can override using
## the `.groups` argument.
```

```
#This produces totals across the US per date, summing state totals.
```

Visualize US Data by State

Next we are going to visualize some of the data we just sorted by state for any preliminary patterns we may see.

```
US_totals %>%
  filter(cases > 0) %>%
  ggplot(aes(x = date, y = cases)) +
  geom_line(aes(color = "cases")) + #adds a line to the graph, random color
  geom_point(aes(color = "cases")) + #adds points to the graph, same random color
  geom_line(aes(y = deaths, color = "deaths")) + #adds a line to the graph, new random color
  scale_y_log10() + #scaled the y-variable on a log scale due to large numbers
  theme(legend.position = "bottom", #positions key on the bottom of the graph
        axis.text.x = element_text(angle = 90)) + #angles x-axis labels at 90 degrees to the graph
  labs(title = "COVID19 in US", y = NULL) #titles the graph and keeps the y-axis from having a title
```




```
#This chunk produces a graph of US total deaths and cases over time. The cases line has points of the s
```

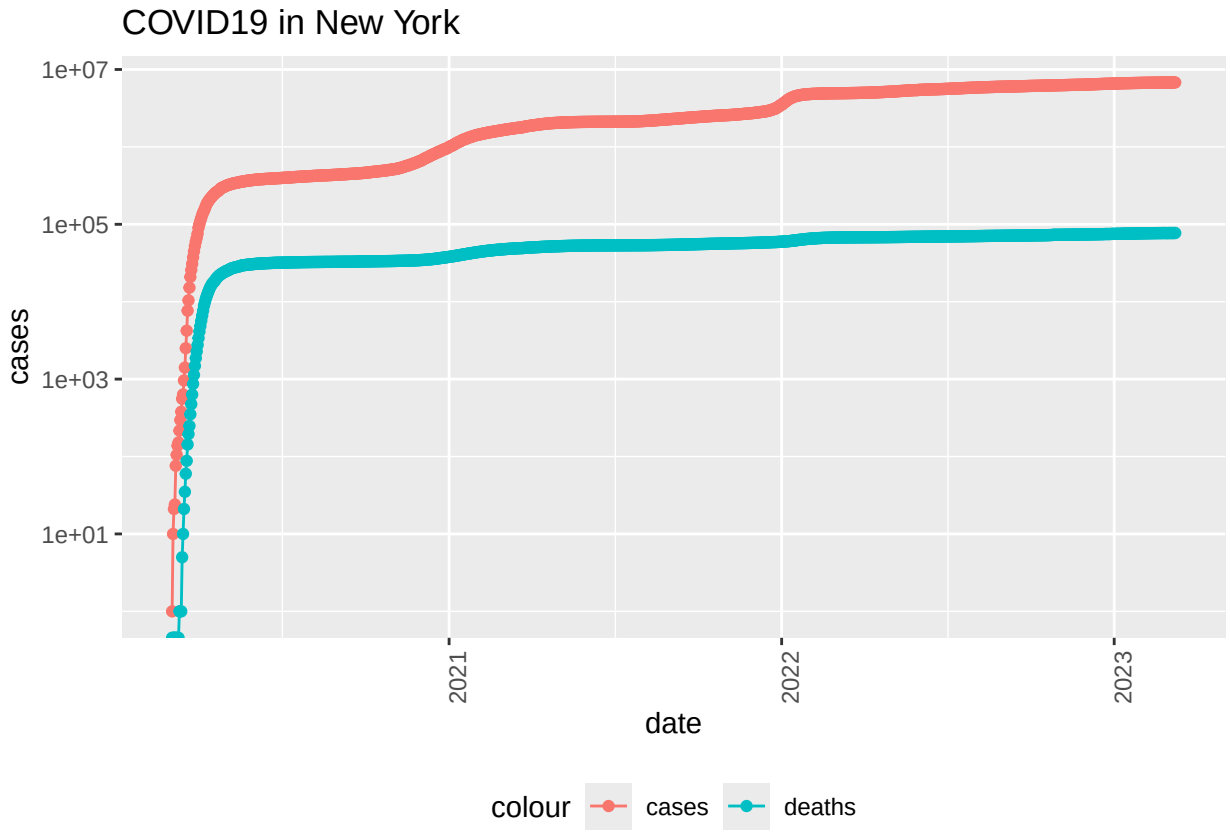
New York State Visualization

Next we will create a visualization of New York (state)'s COVID-19 cases and deaths over time.

```
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))
```

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



New Transformations

In order to check if the graphs produced by the previous visualizations accurately represent the idea that new COVID-19 cases have plateaued, we will transform the US data by state and as a total.

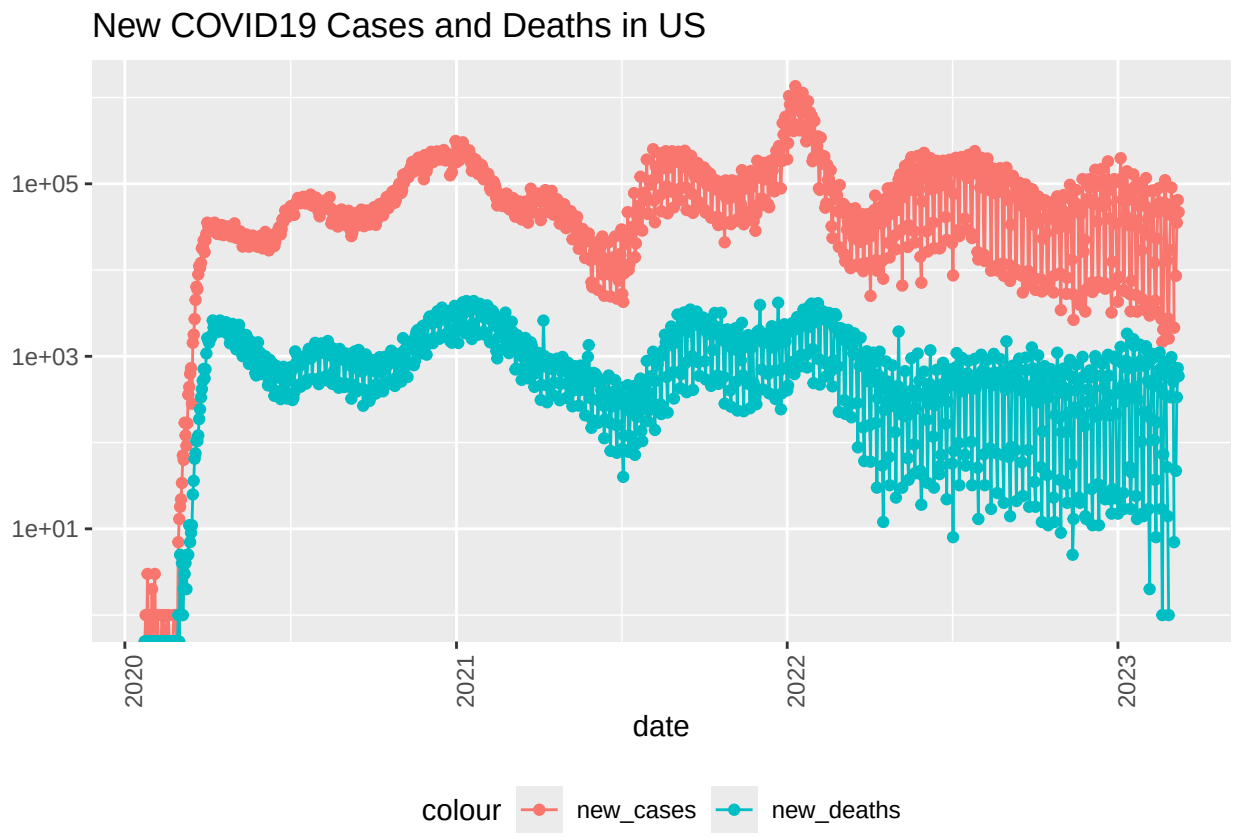
```
US_by_state <- US_by_state %>%
  mutate(new_cases = cases - lag(cases), new_deaths = deaths - lag(deaths))

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

#tail(US_totals %>% select(new_cases, new_deaths, everything()))
```

Visualize New Transformations

Having transformed the data to include new cases and new deaths, we will now visualize new cases and deaths over time in a graph for the full US totals.

```
US_totals %>%
  ggplot(aes(x = date, y = new_cases)) +
  geom_line(aes(color = "new_cases")) + #adds a line to the graph, random color
  geom_point(aes(color = "new_cases")) + #adds points to the graph, same random color
  geom_line(aes(y = new_deaths, color = "new_deaths")) + #adds a line to the graph, new random color
  geom_point(aes(y = new_deaths, color = "new_deaths")) + #adds points to the graph, same random color
  scale_y_log10() + #scaled the y-variable on a log scale due to large numbers
  theme(legend.position = "bottom", #positions key on the bottom of the graph
        axis.text.x = element_text(angle = 90)) + #angles x-axis labels at 90 degrees to the graph
  labs(title = "New COVID19 Cases and Deaths in US", y = NULL) #titles the graph and keeps the y-axis f
```

```
## 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 4 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



Visualize a Few States

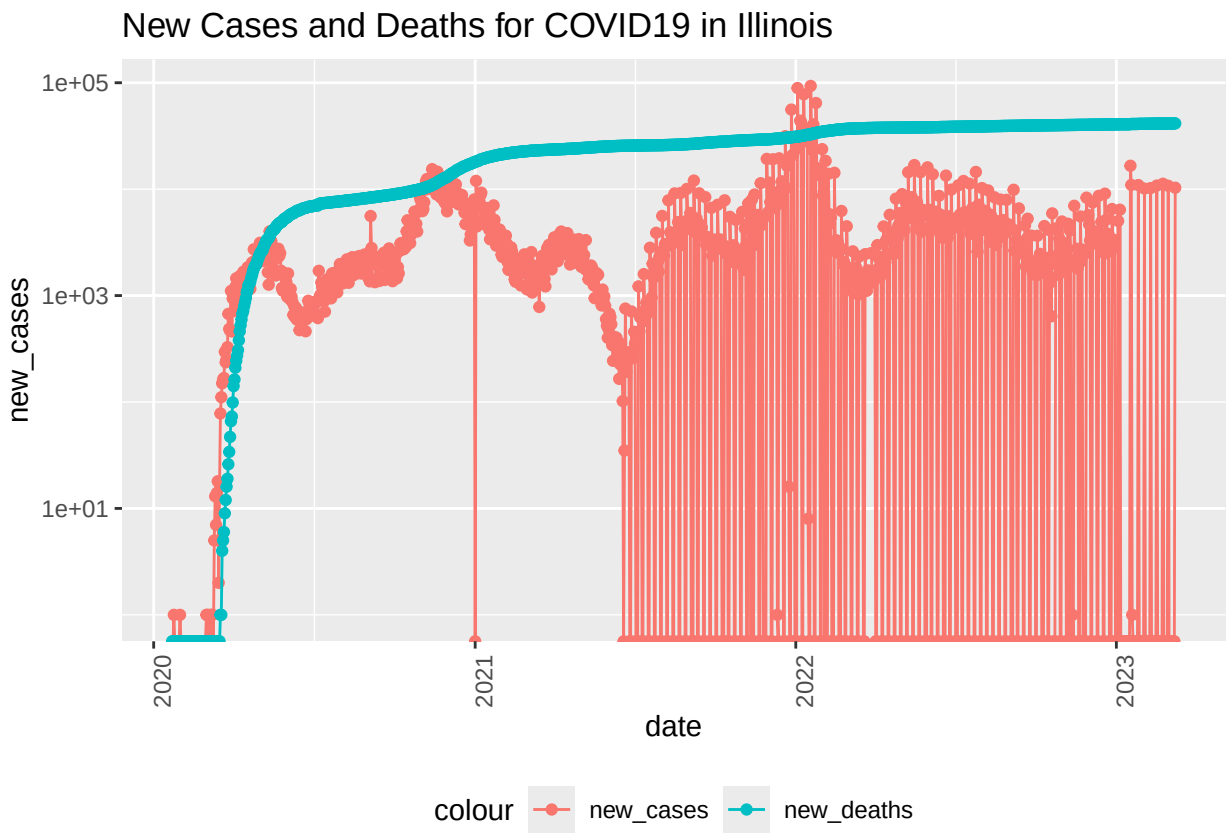
Next we will visualize a few states' data regarding new cases and new deaths.

```
state <- "Illinois"

US_by_state %>%
  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 = deaths, color = "new_deaths")) +
  geom_point(aes(y = deaths, color = "new_deaths")) +
  scale_y_log10() +
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 90)) +
  labs(title = str_c("New Cases and Deaths for 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.
## log-10 transformation introduced infinite values.
## log-10 transformation introduced infinite values.

## Warning: Removed 1 row containing missing values or values outside the scale range
```

```
## (`geom_line()`).
```

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
# creates a graph of new deaths and cases of COVID-19 in the listed state

state <- "Colorado"

US_by_state %>%
  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 = deaths, color = "new_deaths")) +
  geom_point(aes(y = deaths, color = "new_deaths")) +
  scale_y_log10() +
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 90)) +
  labs(title = str_c("New Cases and Deaths for COVID19 in ",state, y = NULL))
```

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

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

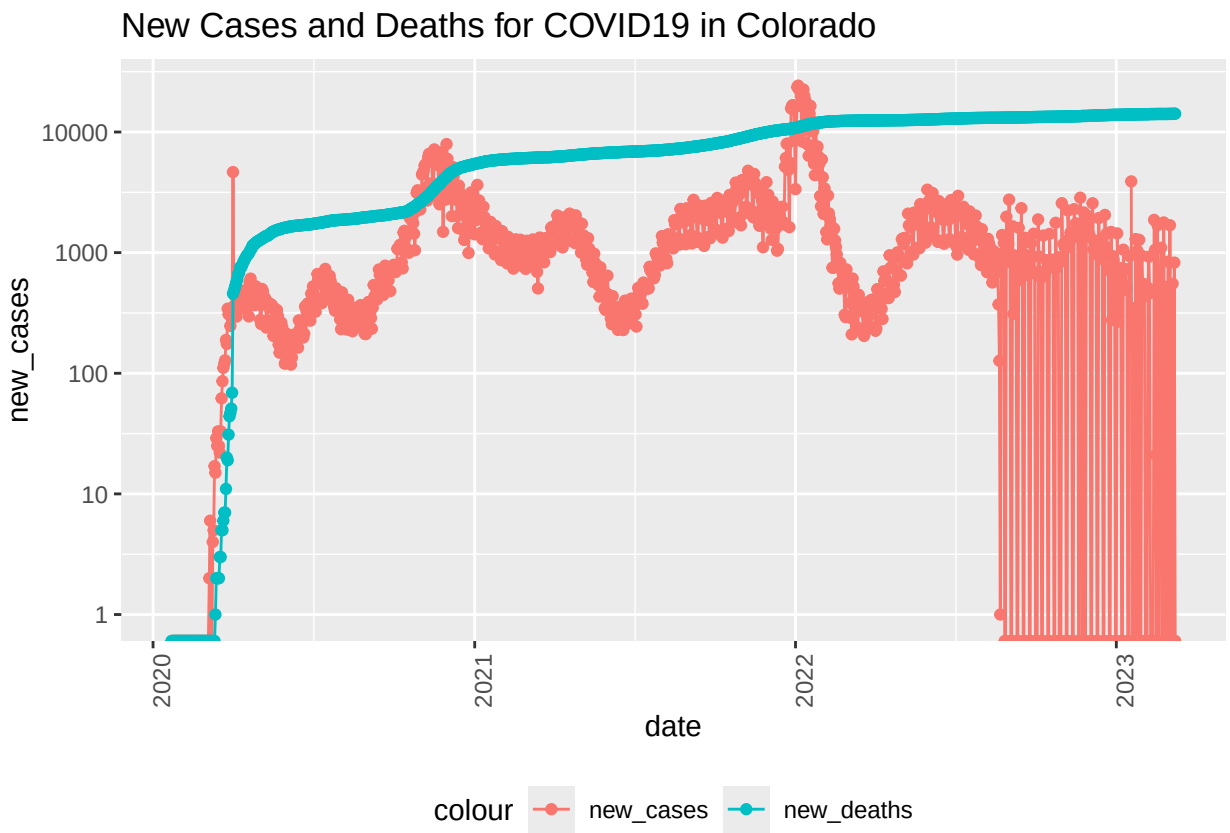
```
## Warning in transformation$transform(x): NaNs produced
```

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

```
## 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()`).
```




```
# creates a graph of new deaths and cases of COVID-19 in the listed state

state <- "Virgin Islands"

US_by_state %>%
  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 = deaths, color = "new_deaths")) +
  geom_point(aes(y = deaths, color = "new_deaths")) +
  scale_y_log10() +
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 90)) +
  labs(title = str_c("New Cases and Deaths for COVID19 in ",state, y = NULL))
```

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

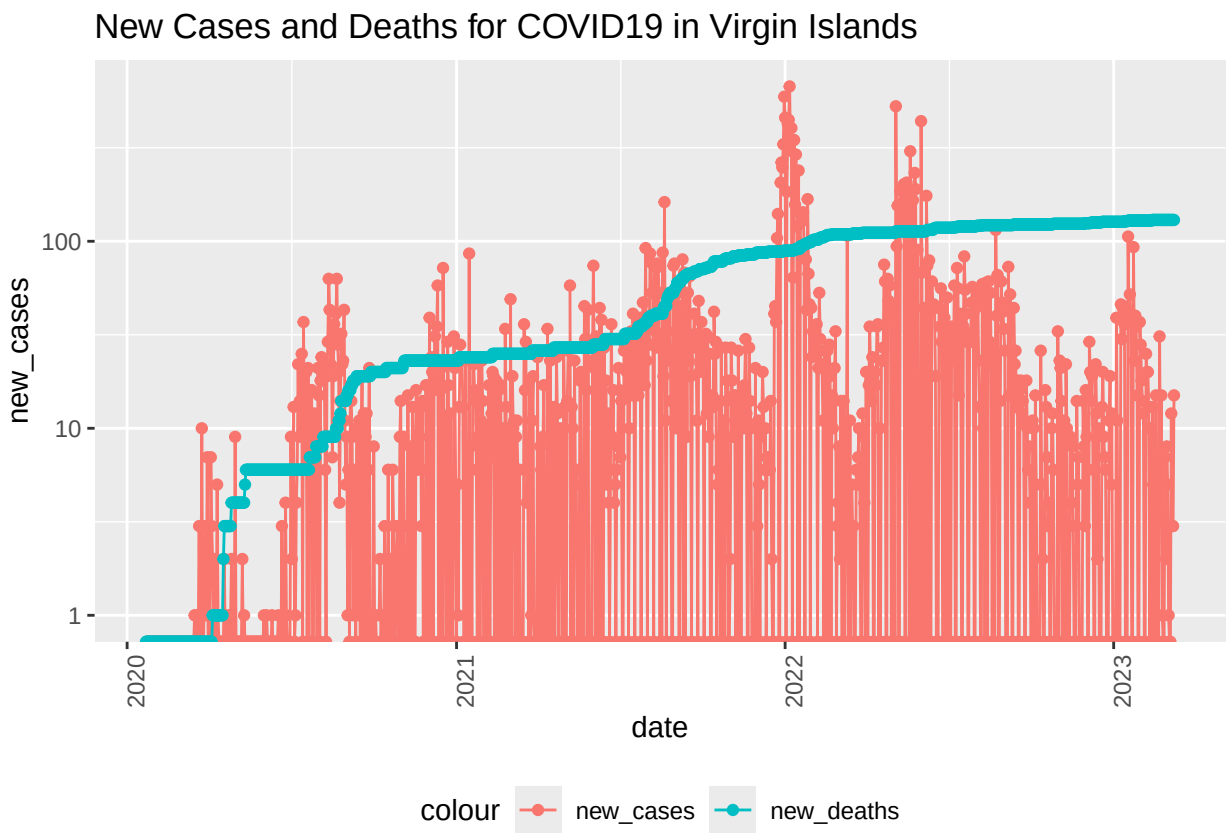
```
## Warning in scale_y_log10(): log-10 transformation introduced infinite values.
```

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

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

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

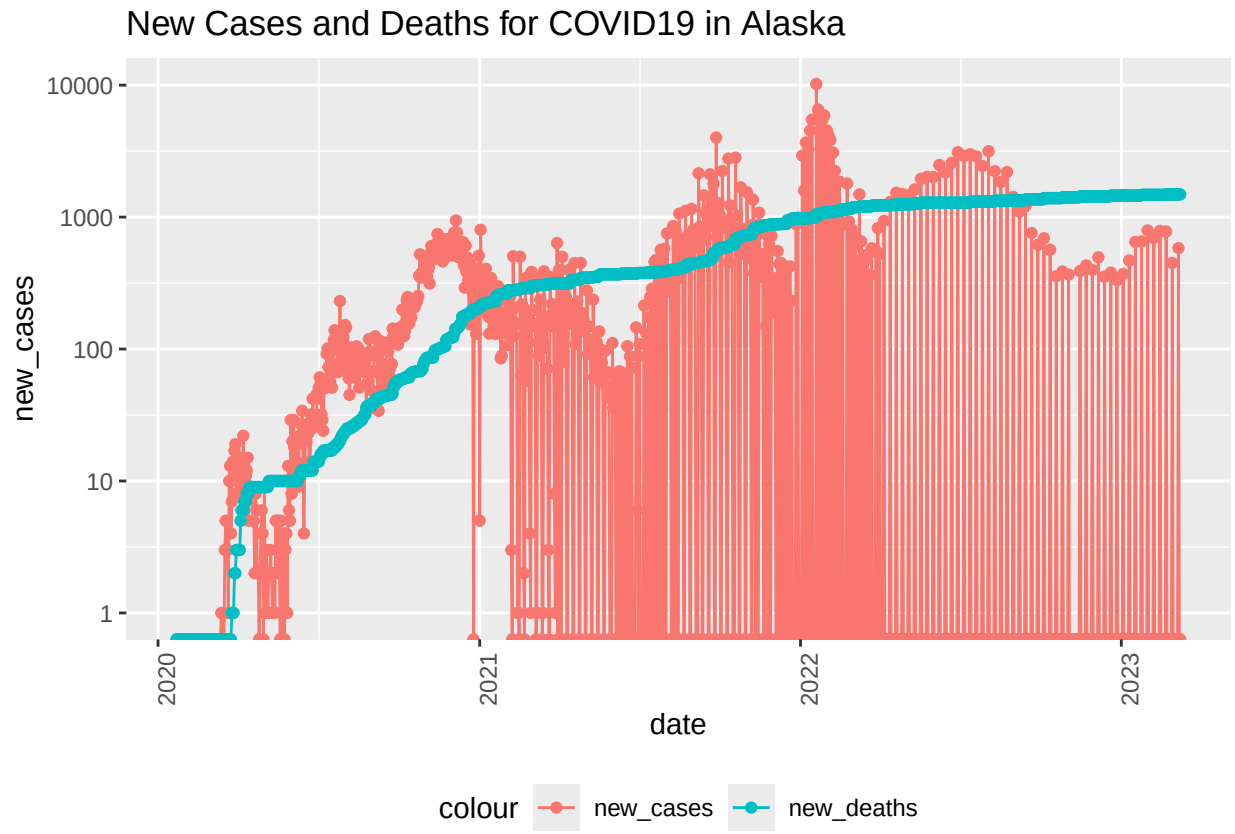
```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
# creates a graph of new deaths and cases of COVID-19 in the listed state

state <- "Alaska"

US_by_state %>%
  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 = deaths, color = "new_deaths")) +
  geom_point(aes(y = deaths, color = "new_deaths")) +
  scale_y_log10() +
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 90)) +
  labs(title = str_c("New Cases and Deaths for COVID19 in ",state, y = NULL))
```

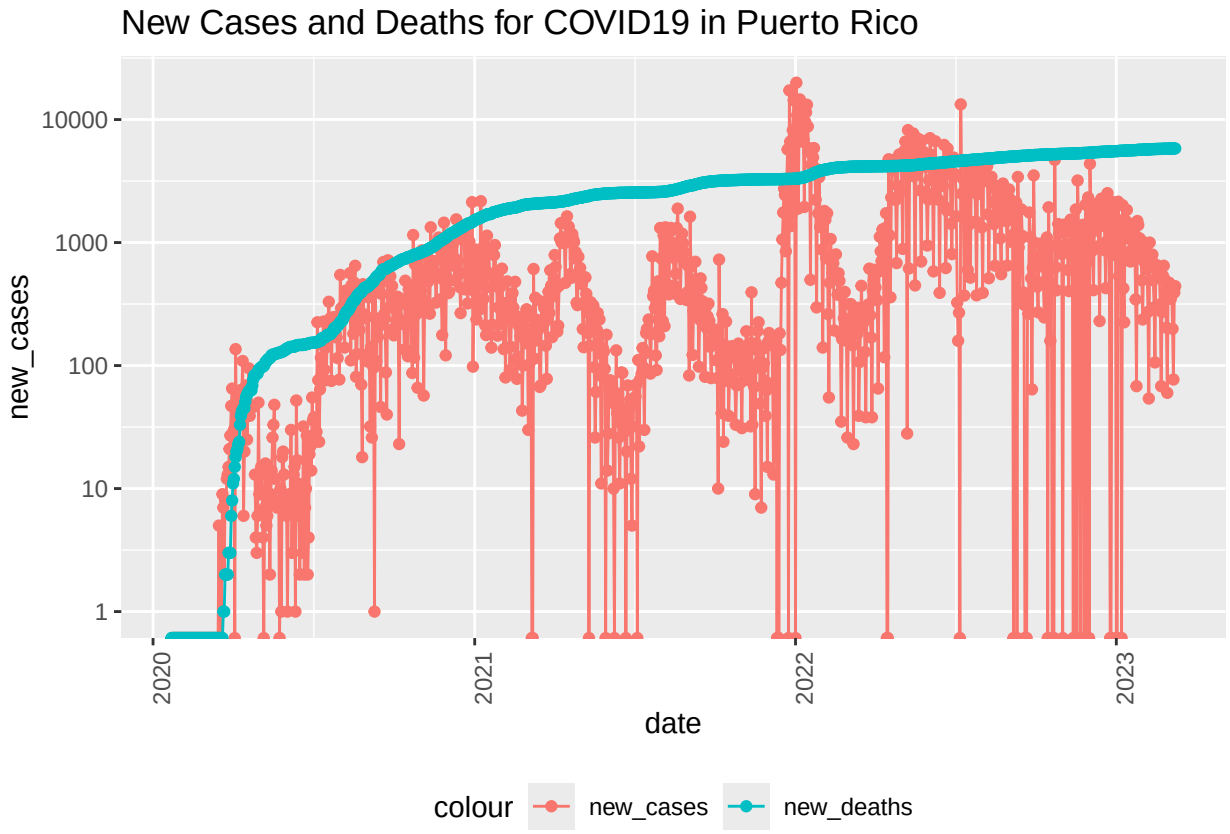
```
## Warning in transformation$transform(x): NaNs produced
```

```
## 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.
## log-10 transformation introduced infinite values.
## 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()`).
```




```r
# creates a graph of new deaths and cases of COVID-19 in the listed state

state <- "Puerto Rico"

US_by_state %>%
  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 = deaths, color = "new_deaths")) +
  geom_point(aes(y = deaths, color = "new_deaths")) +
  scale_y_log10() +
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 90)) +
  labs(title = str_c("New Cases and Deaths for 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.
## log-10 transformation introduced infinite values.
## log-10 transformation introduced infinite values.

## 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()`).
```




```
# creates a graph of new deaths and cases of COVID-19 in the listed state

state <- "Hawaii"

US_by_state %>%
  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 = deaths, color = "new_deaths")) +
  geom_point(aes(y = deaths, color = "new_deaths")) +
  scale_y_log10() +
```

```
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 90)) +
  labs(title = str_c("New Cases and Deaths for COVID19 in ",state, y = NULL))
```

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

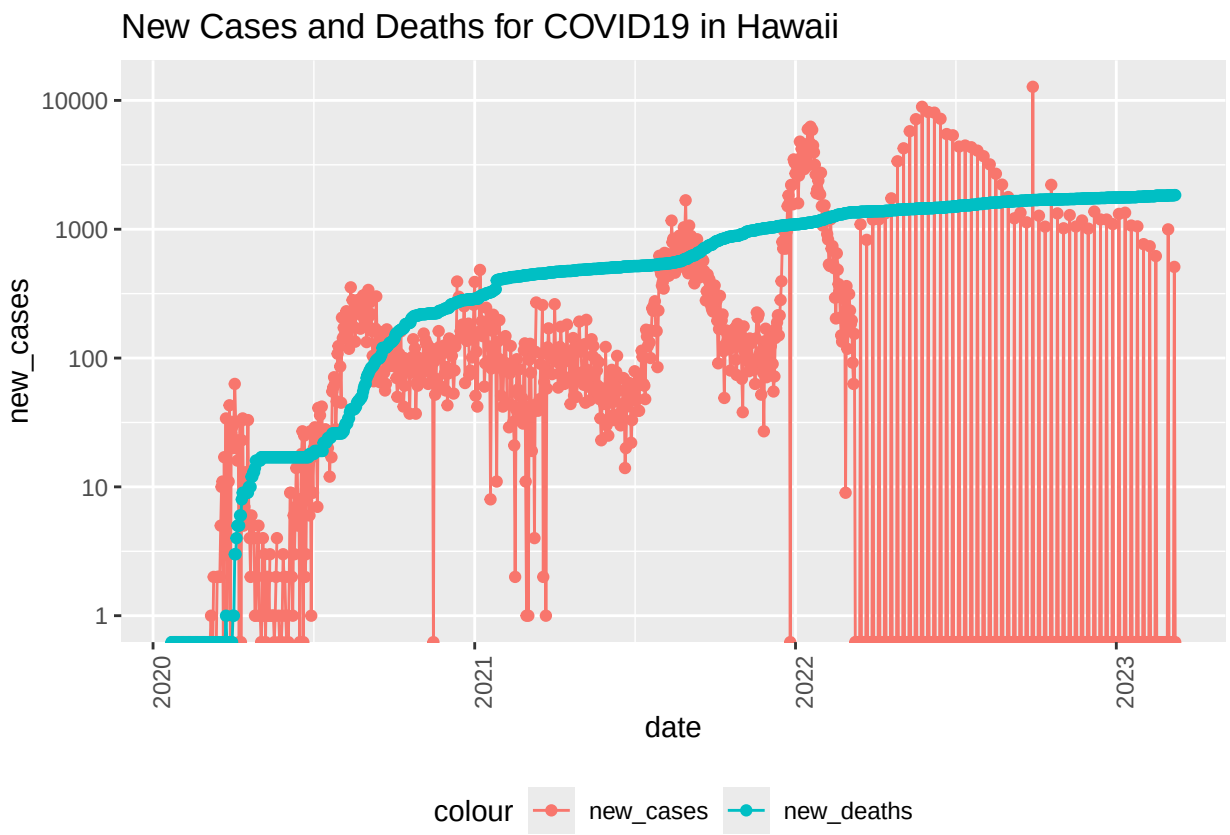
```
## Warning in scale_y_log10(): log-10 transformation introduced infinite values.
```

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

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

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

```
## Warning: Removed 10 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
# creates a graph of new deaths and cases of COVID-19 in the listed state
```

Transform US by State Data for State Totals

Here, we will create a new data set to hold the maximum totals of the cases and deaths per state in the US.

```
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) %>%
  filter(cases > 0, population > 0)
# This creates a new dataframe to hold the max deaths, cases, and population of each state in the US.

#US_state_totals %>%
#  slice_min(deaths_per_thou, n = 10)
#Gives smallest ten states in deaths_per_thou
```

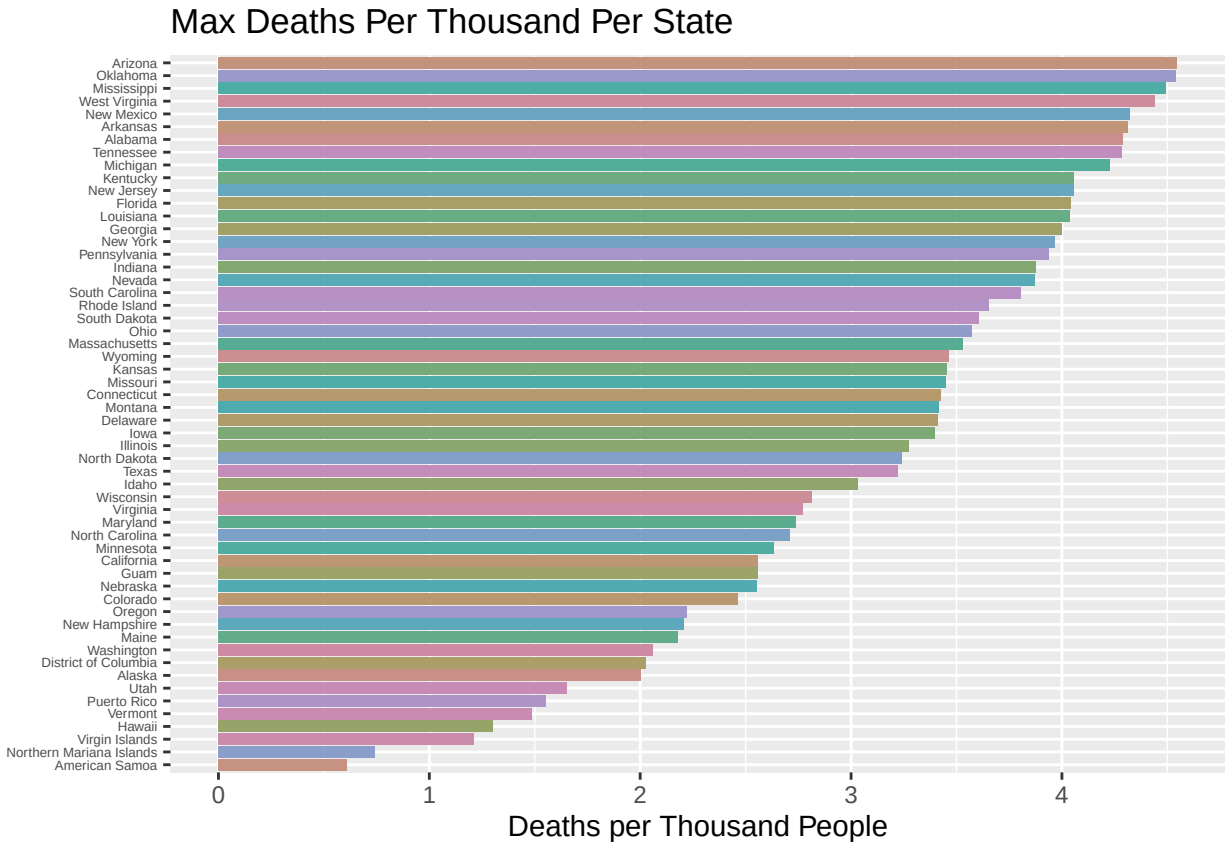
Visualize Some State Comparisons

Now we will compare some states within the same visualizations.

All States Max Deaths Per Thousand

First we'll look at all the states in a bar plot.

```
ggplot(US_state_totals, aes(x= as.factor(reorder(Province_State, deaths_per_thou)), fill = as.factor(Pr
  labs(title = "Max Deaths Per Thousand Per State")
```

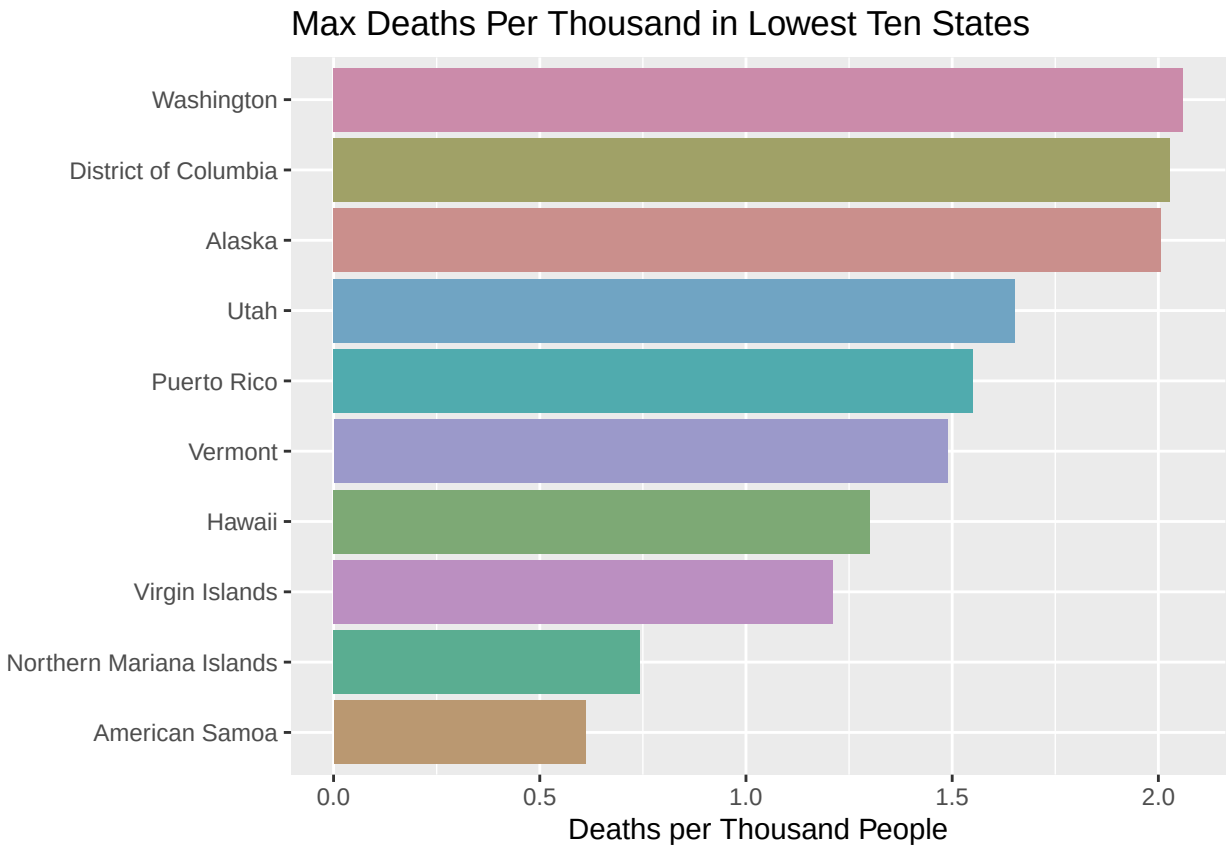



```
#Creates a bar plot of all US States in descending order
```

Lowest Ten States

Since it's difficult to really grasp all the states at once, here, we will show the lowest ten states in deaths per thousand within the same graph to see if there are large jumps between them.

```
US_low_deaths <- US_state_totals %>%
  slice_min(deaths_per_thou, n = 10)

#Stores the smallest ten states in deaths_per_thou into US_low_deaths


ggplot(US_low_deaths, aes(x= as.factor(reorder(Province_State, deaths_per_thou)), fill = as.factor(Provi
  labs(title = "Max Deaths Per Thousand in Lowest Ten States")
```




```
#creates a horizontal bar chart showing the ten lowest states in deaths per thousand in descending orde
# + axis.text.x = element_text(size = 8)
```

Compare Countries Across the Globe

First we need to transform and tidy the global data to compare max deaths and cases per country/province.

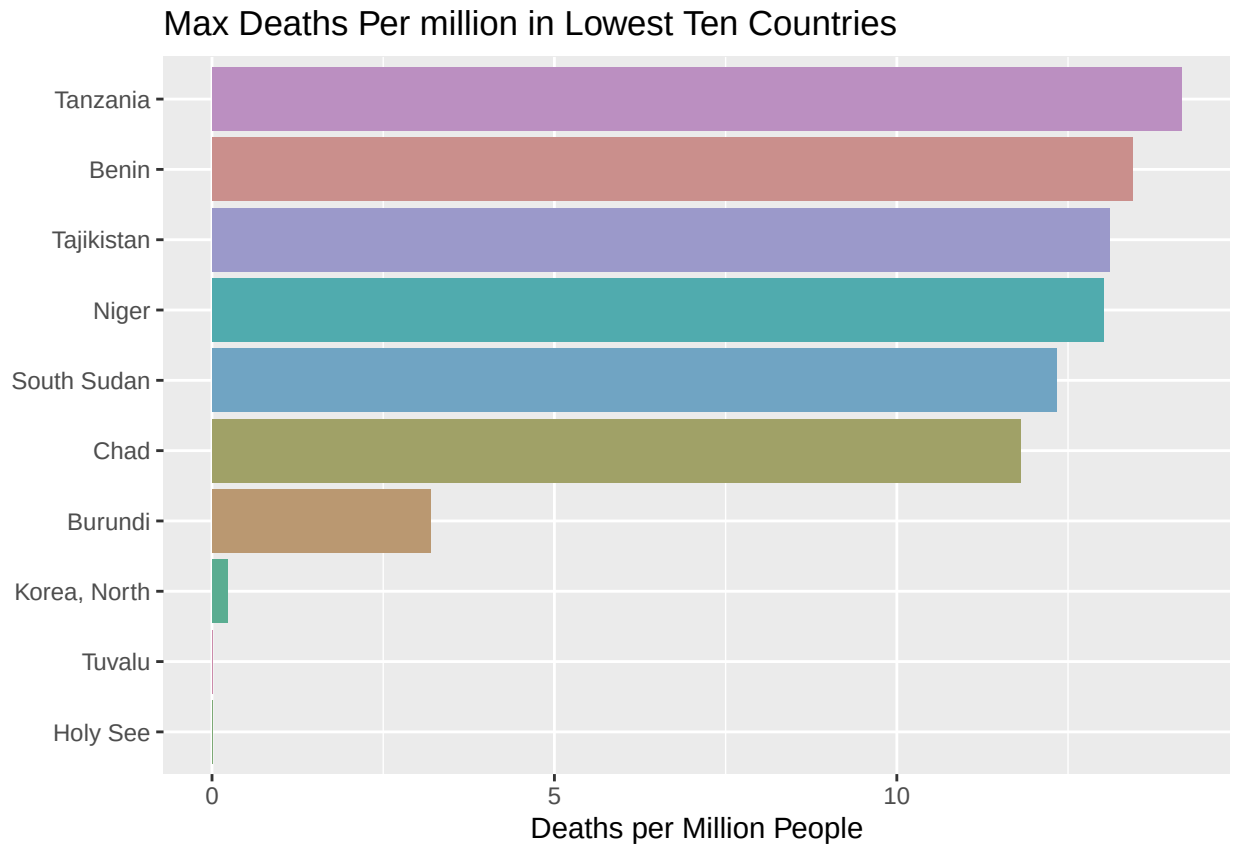
```
global_by_region <- global %>%
  group_by(Country_Region, date) %>%
  summarize(cases = sum(cases), deaths = sum(deaths), Population = sum(Population)) %>%
  mutate(deaths_per_mill = deaths * 1000000 / Population) %>%
  select(Country_Region, date, cases, deaths, deaths_per_mill, Population) %>%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

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

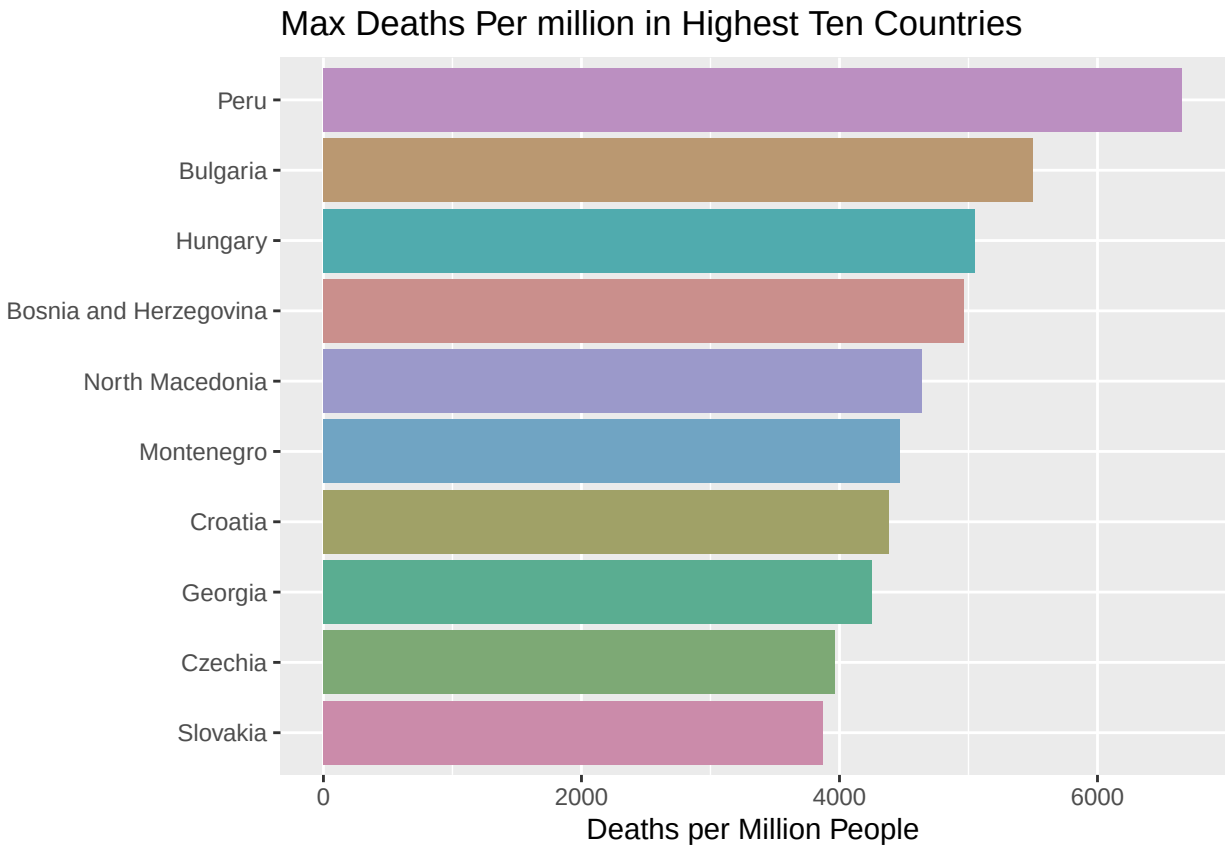
```
#This creates a dataframe that shows the country cases and deaths per date globally.

global_totals <- global_by_region %>%
  group_by(Country_Region) %>%
  summarize(deaths = max(deaths), cases = max(cases),
            population = max(Population),
            cases_per_mill = 1000000 * cases / population,
            deaths_per_mill = 1000000 * deaths / population) %>%
  filter(cases > 0, population > 0)
#This creates a dataframe with the max deaths and cases of each country or region globally.
```

Next we will visualize the top 10 and bottom 10 in deaths.

```
global_low_deaths <- global_totals %>%
  slice_min(deaths_per_mill, n = 10) #creates subset of lowest ten countries

global_high_deaths <- global_totals %>%
  slice_max(deaths_per_mill, n = 10) #creates subset of highest ten countries

ggplot(global_low_deaths, aes(x= as.factor(reorder(Country_Region, deaths_per_mill)), fill = as.factor(
  labs(title = "Max Deaths Per million in Lowest Ten Countries")
```




```
#Produces a graph showing the ten lowest max deaths per country or region globally form available data

ggplot(global_high_deaths, aes(x= as.factor(reorder(Country_Region, deaths_per_mill)), fill = as.factor
  labs(title = "Max Deaths Per million in Highest Ten Countries")
```

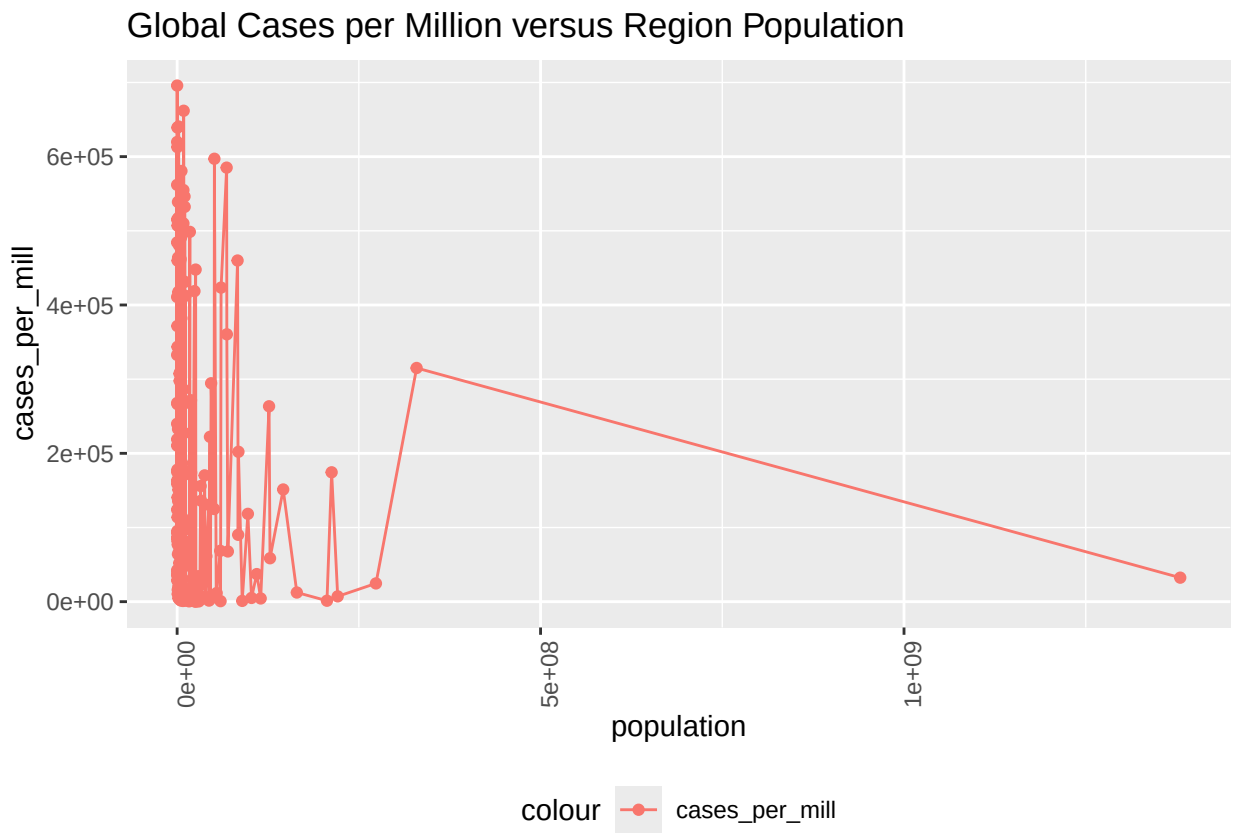



```
#Produces a graph showing the ten highest max deaths per country or region globally form available data
```

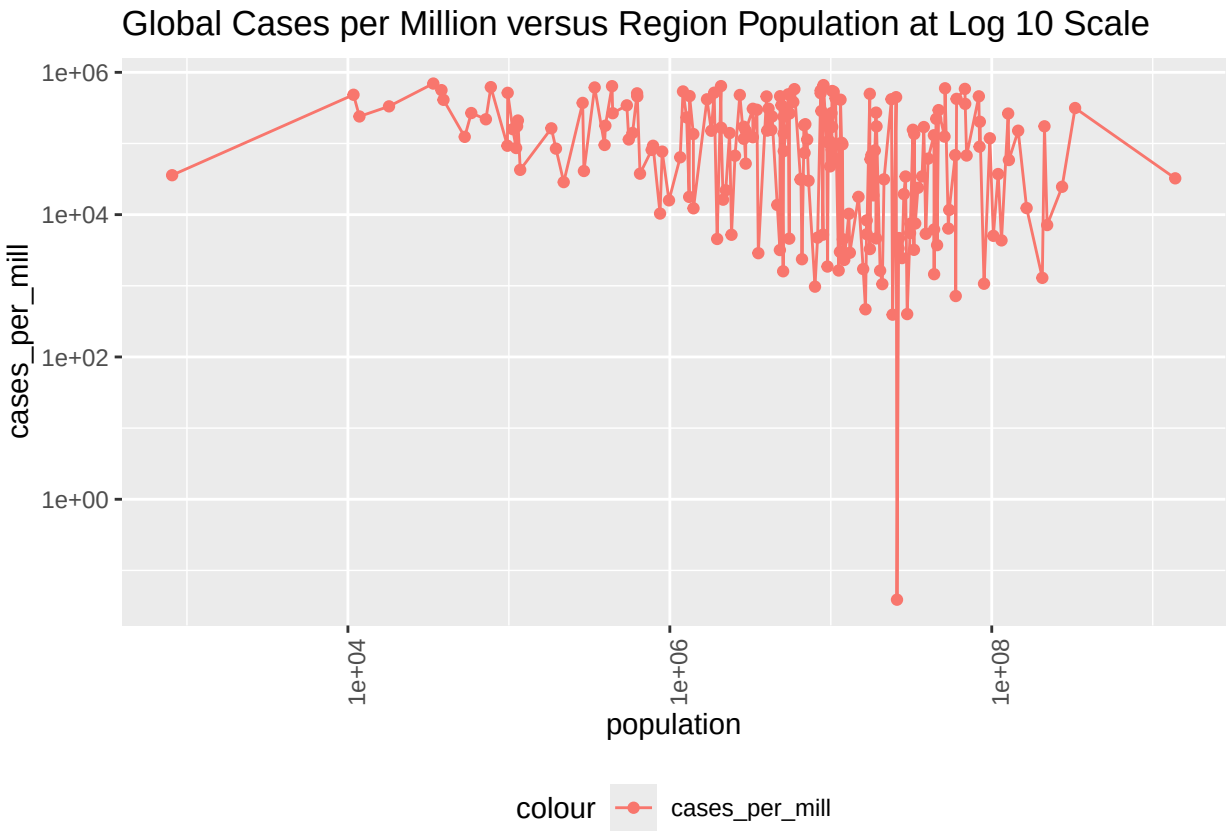
Compare Global Populations versus Cases per Million

Higher populations are more likely to have higher counts, which is supposedly mitigated by looking at transformations like cases per million. However, we will still take a closer look at the relationship between population and cases per million on a global scale.

```
global_totals %>%
ggplot(aes(x = population, y = cases_per_mill)) +
  geom_line(aes(color = "cases_per_mill")) +
  geom_point(aes(color = "cases_per_mill")) +
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 90)) +
  labs(title = "Global Cases per Million versus Region Population")
```




```r
#Unscaled graph created of global cases per million versus region population

global_totals %>%
ggplot(aes(x = population, y = cases_per_mill)) +
  geom_line(aes(color = "cases_per_mill")) +
  geom_point(aes(color = "cases_per_mill")) +
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 90)) +
  scale_x_log10() +
  scale_y_log10() +
  labs(title = "Global Cases per Million versus Region Population at Log 10 Scale") #same graph with lo
```



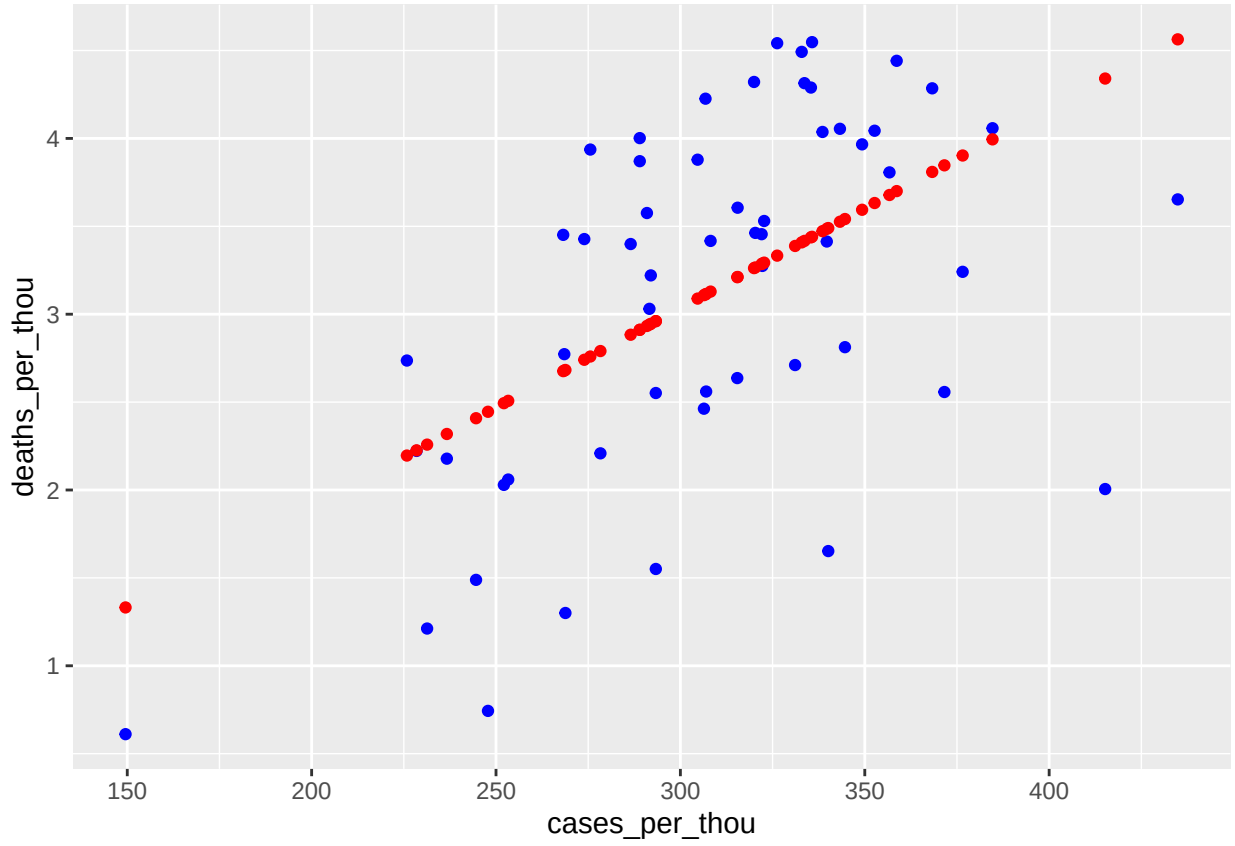
Modeling US and Global Data

This can involve adding new variables, but for now we will start simply.

Linear Model of US Deaths vs. Cases.

We are going to look at a linear model where the deaths per thousand people will be the result of a function of the cases per thousand people.

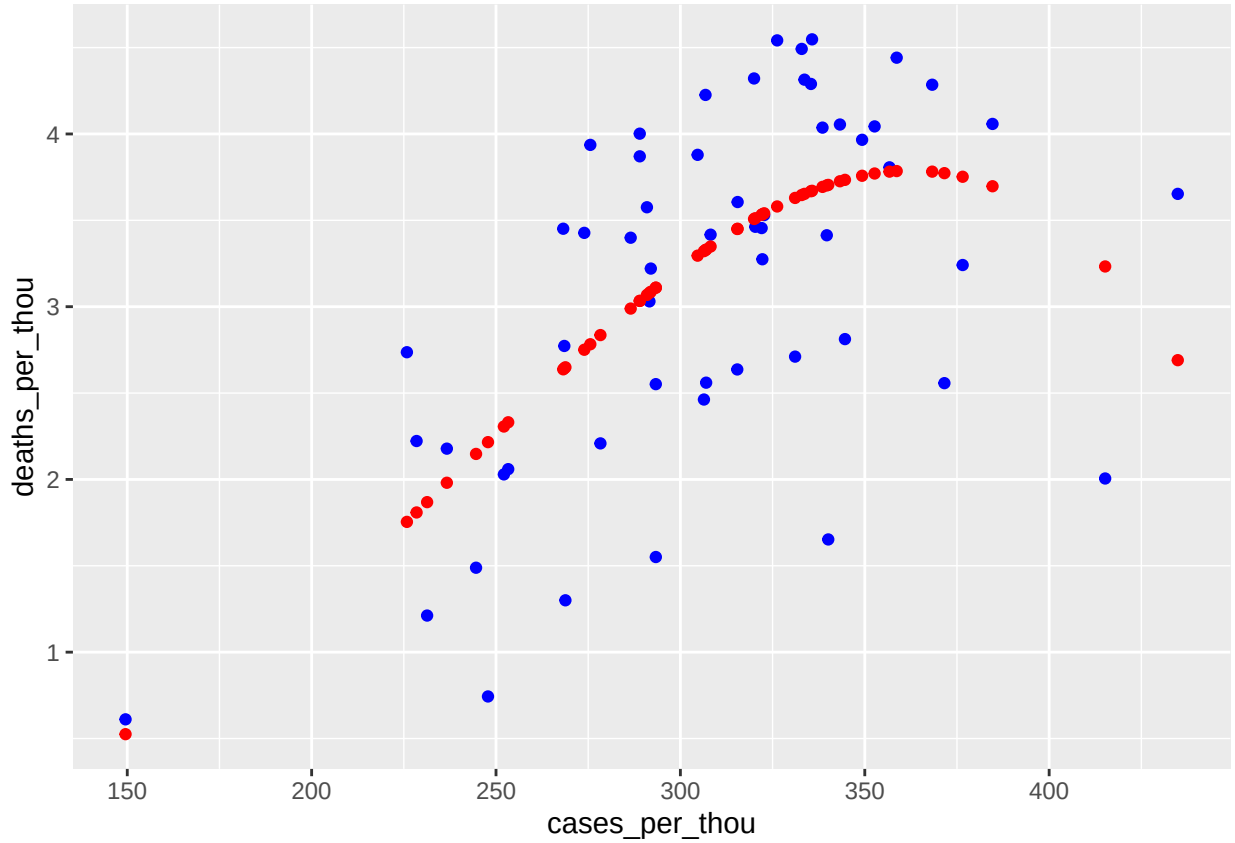
```
mod <- lm(deaths_per_thou ~ cases_per_thou, data = US_state_totals)
#This creates the linear function; summary(mod) can produce the coefficients, R value, etc.

#x_grid <- seq(1,450)
#found min and max cases per thousand to be 150 to 435, set grid to be 1 to 450 to encompass all of thi

#new_df <- tibble(cases_per_thou = x_grid)
#puts cases per thousand into a tibble named new_df on the created grid

US_tot_w_pred <- US_state_totals %>% mutate(pred = predict(mod))
#adds a new column "pred" showing the prediction based off of the model "mod" for cases per thousand pe

US_tot_w_pred %>% ggplot() +
  geom_point(aes(x = cases_per_thou, y = deaths_per_thou), color = "blue") + geom_point(aes(x = cases_p
```



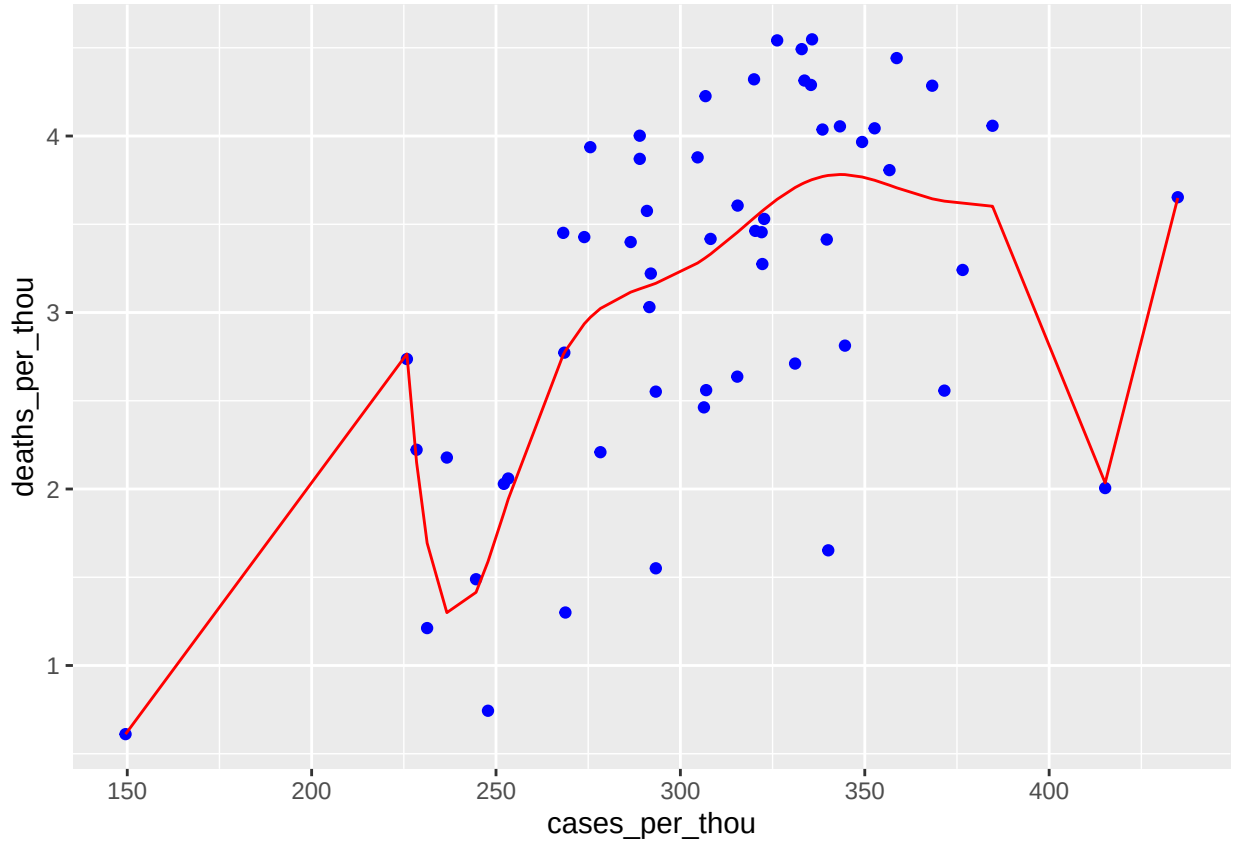
Modeling Deaths vs Cases per Thousand with a Larger Polynomial Function

Next, we will see if a cubic or other polynomial function is a better fit.

```
cubic_mod <- lm(deaths_per_thou ~ poly(cases_per_thou, degree = 3), data = US_state_totals)
#This has a higher R squared value than the linear model (0.2933), but it still falls short of 0.50 at

US_tot_w_pred <- US_state_totals %>% mutate(pred = predict(cubic_mod))
#adds a new column "pred" showing the prediction based off of the model "mod" for cases per thousand pe

US_tot_w_pred %>% ggplot() +
  geom_point(aes(x = cases_per_thou, y = deaths_per_thou), color = "blue") + geom_point(aes(x = cases_p
```




```
poly_mod <- lm(deaths_per_thou ~ poly(cases_per_thou, degree = 9), data = US_state_totals)
#creates a polynomial function with a degree of 9 exploring deaths_per_thou over cases_per_thou

# R-squared for poly degrees: 12 is 0.4148, 11 is 0.4261, 10 is 0.4387, 9 is 0.4472,8 is 0.4421

US_tot_w_pred <- US_state_totals %>% mutate(pred = predict(poly_mod))
#adds a new column "pred" showing the prediction based off of the model "mod" for cases per thousand pe

US_tot_w_pred %>% ggplot() +
  geom_point(aes(x = cases_per_thou, y = deaths_per_thou), color = "blue") + geom_line(aes(x = cases_pe
```

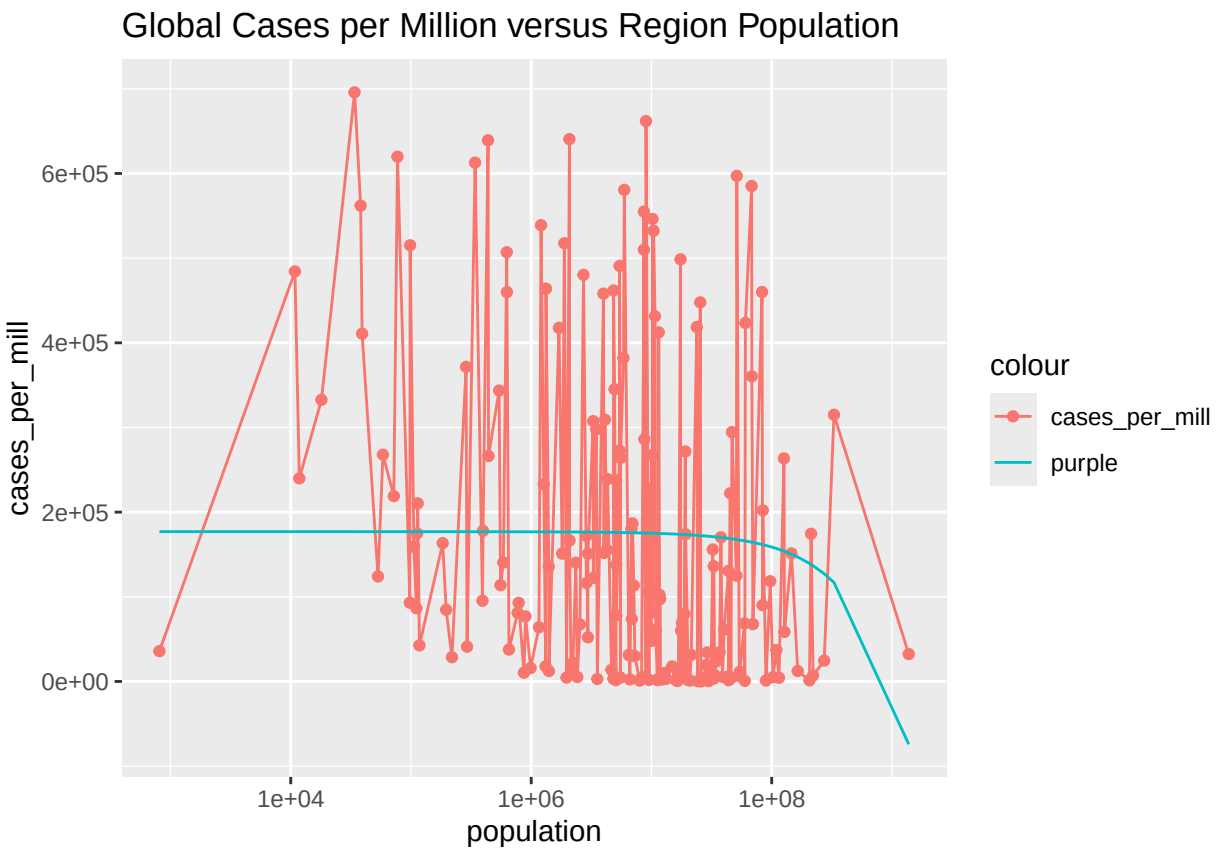


Modelling Cases over Population by Region, Globally

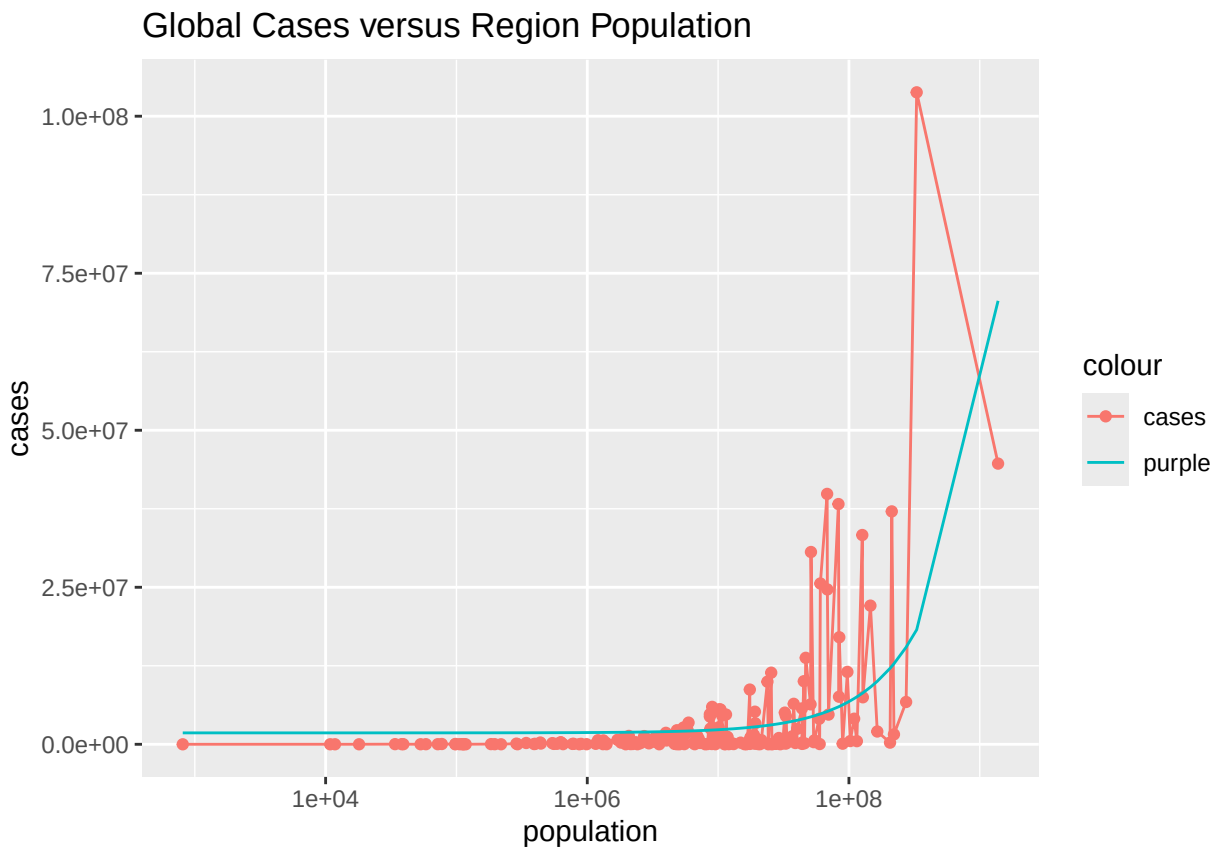
First we will look globally at cases per million over population, to see if the ratio of cases climbs with the population or if the same basic percentage of people are infected and reported regardless of population.

Next, we will double check if cases climb as population does, overall, on a global scale.

```
lin_mod <- lm(cases_per_mill ~ population, data = global_totals)
#Adjusted R-squared is all the way down at 0.005729 currently

global_tot_w_pred <- global_totals %>% mutate(pred = predict(lin_mod))


global_tot_w_pred %>%
  filter(cases_per_mill > 0) %>%
ggplot(aes(x = population, y = cases_per_mill)) +
  geom_line(aes(color = "cases_per_mill")) +
  geom_point(aes(color = "cases_per_mill")) +
  geom_line(aes(y = pred, color = "purple")) +
  scale_x_log10() +
  labs(title = "Global Cases per Million versus Region Population")
```




```r
#Linear regression with graph of global cases per million versus region population


lin_mod2 <- lm(cases ~ population, data = global_totals)
#Adjusted R-squared is at 0.2703 at time of writing

global_tot_w_pred <- global_totals %>% mutate(pred = predict(lin_mod2))


global_tot_w_pred %>%
ggplot(aes(x = population, y = cases)) +
  geom_line(aes(color = "cases")) +
  geom_point(aes(color = "cases")) +
  geom_line(aes(y = pred, color = "purple")) +
  scale_x_log10() +
  labs(title = "Global Cases versus Region Population")
```




```
#Linear regression with graph of max global cases total per region versus region population
```

The linear regression implies an exponential function might be a better fit.

```
exp_mod <- lm(cases ~ log(population), data = global_totals)

global_tot_w_pred <- global_totals %>% mutate(pred = predict(exp_mod))
#Adds pred column with predictions based on exponential regression

global_tot_w_pred %>%
ggplot(aes(x = population, y = cases)) +
  geom_line(aes(color = "cases")) +
  geom_point(aes(color = "cases")) +
  geom_line(aes(y = pred, color = "purple")) +
  scale_x_log10() +
  labs(title = "Global Cases versus Region Population")
```


```
#Creates a graph of global cases over region population with an exponential regression
```

It seems from this analysis that the initial linear regression is a better fit.

Analysis and Bias Discussion

Looking at our modeled graphs, the polynomial linear regression with a degree of 9 comes closest to an acceptable R-squared value (at the time of this publication: 0.4472 adjusted; multiple R-squared is acceptable at 0.5376). This raises some questions: * Why do more cases trend towards resulting in disproportionately more deaths compared to lower case counts up to a point where that trend seems to start reversing? * Additionally, what accounts for our outliers with very low deaths compared to case count, * and some that have higher deaths compared to the prediction at higher case counts?

We could speculate that higher case counts result in overworking our hospitals and doctors, resulting in more deaths for our outliers.

However, we could also speculate that income of an area may play a stronger role in determining deaths per case count, since socioeconomic status in the US often results in differences in health care, environmental health, and the public's physical health overall. This would account for greater cases and greater deaths proportionately.

Lower income families have less choice about going to work, too, meaning a higher chance for spread and for higher case counts. This might be harder to track by state than by city, but we do have city data available in our larger datasets that could be worth looking into.

For both avenues of analysis, we first would need to acquire financial information and healthcare employment information regarding the states and cities in question.

We also might see bias in how cases are reported in communities based on whether the individual goes to a

hospital or doctor or performs a test at home. This can be influenced by distrust of medicine, distrust of the government, a lack of time, option, or education, or other fears. They might not even know they have COVID if they are asymptomatic or have mild symptoms.

It is also possible that increased cases result in people catching COVID more than once, meaning they already have some immunity to it and are unlikely to die, resulting in the downward turn in deaths at the highest cases per thousand. There could be benefits from doctors and the community having more experience with COVID-19 and being able to identify it and isolate, or simply a push for vaccination once cases are high enough.

There is the possibility of deaths being reported as being caused by secondary infections when COVID-19 was the primary factor or vice versa, where a secondary infection or respiratory infection was the cause of death, but COVID-19 was blamed.

For myself, I tried to mitigate any bias towards what countries might have higher or lower COVID counts by simply letting the data show me better and picking countries to investigate either by random or by top/bottom ten.

What we have learned from our regressions is that global cases per million max totals do not have a strong relationship with population, but do have an upward trend that seems to be exponential. While this relies on countries to report their cases number closely, the law of large numbers does play in the favor of the validity of this data. There are some smaller sample sizes which may throw off the analysis, however it does seem that population is not the main factor in number of cases per million people.

When we compared total max cases to total population, we got a seemingly exponential relationship with a linear regression, but an exponential regression did not fit the data nearly as well. This does show an upward trend in cases as population increases, but there are outliers causing an exponential regression to be a bad fit, where higher population did not result in higher cases. This could be due to countries misreporting case data, missing data, or simply that these countries simply did not have that many cases. That could be in result of prior immunities to similar viruses, differences in social, financial, or healthcare situations, or differences in how they handled the virus as a whole. It's even possible that topography could play a role in COVID-19 cases based on whether there was easy entry into the country from others and whether people were able to easily intermingle based on the geography of the country.

When it comes right down to it, the data we have of cases, deaths, and dates versus the populations involved is not enough to come to a solid conclusion on causality without financial, social, topographical, historical, and political information we do not have. However, we can say that, *in our data, population is not a clear and direct indicator of case totals but does seem to show that an increase in population has some effect on increase in cases* and there are other factors that require investigation.

Session Information

Below is a generated summary of the R session information.

```
sessionInfo()
```

```
## R version 4.3.3 (2024-02-29 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 11 x64 (build 22631)
##
## 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/Chicago
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] lubridate_1.9.3 forcats_1.0.0   stringr_1.5.1   dplyr_1.1.4
##  [5] purrr_1.0.2     readr_2.1.5     tidyr_1.3.1     tibble_3.2.1
##  [9] ggplot2_3.5.1   tidyverse_2.0.0 tinytex_0.50
##
## loaded via a namespace (and not attached):
##  [1] bit_4.0.5         gtable_0.3.5      highr_0.10        crayon_1.5.2
##  [5] compiler_4.3.3    tidyselect_1.2.1  parallel_4.3.3    scales_1.3.0
##  [9] yaml_2.3.8        fastmap_1.1.1     R6_2.5.1          labeling_0.4.3
## [13] generics_0.1.3    curl_5.2.1        knitr_1.46        munsell_0.5.1
## [17] pillar_1.9.0      tzdb_0.4.0        rlang_1.1.3       utf8_1.2.4
## [21] stringi_1.8.3     xfun_0.43         bit64_4.0.5       timechange_0.3.0
## [25] cli_3.6.1         withr_3.0.0       magrittr_2.0.3    digest_0.6.35
## [29] grid_4.3.3        vroom_1.6.5       rstudioapi_0.16.0 hms_1.1.3
## [33] lifecycle_1.0.4   vctrs_0.6.5       evaluate_0.23     glue_1.7.0
## [37] farver_2.1.1      fansi_1.0.6       colorspace_2.1-0  rmarkdown_2.26
## [41] tools_4.3.3       pkgconfig_2.0.3   htmltools_0.5.8
```