COVID19_data_analysis

2024-08-20

COVID19 data analysis

Statement of interest:

- In this analysis, the primary goal is to examine the impact of COVID-19 on a global and national scale for US, with a specific focus on mortality rates. Understanding these rates is crucial as it helps identify regions that have been most severely affected by the pandemic and may require more targeted public health interventions.**

In this analysis the aim is to achieve the following objectives:

1. **Visualize deaths per 1000 population in the US:**
   - Create a colored map to highlight the regions with the highest mortality rates.
   - Provide insights into which areas were most severely affected by the pandemic.
2. **Compare US mortality rates to global rates:**
   - Develop a visualization comparing US deaths per 1000 population to the global average.
   - Assess the effectiveness of US mitigation strategies relative to the global response.
3. **Build a predictive model:**
   - Analyze factors contributing to mortality rates.
   - Forecast potential future trends to prepare for future public health crises.

Data source:

- I will be using Global and US covid19 cases and deaths data shared John hopkins university. This data is available on github: "https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_time_series"

The data contains files regarding global number of cases and deaths as well as files having US number of cases and deaths. The data has information of province/state, country/region, latitude/longitude, Date

Installing necessary libraries

```
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)
library(dplyr)
library(readr)
library(tidyr)
library(knitr)
library(ggplot2)
```

Loading and saving the data

```
# Load the CSV files
url_in <-"https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covi

file_names <-
  c("time_series_covid19_confirmed_US.csv",
    "time_series_covid19_confirmed_global.csv",
    "time_series_covid19_deaths_US.csv",
    "time_series_covid19_deaths_global.csv")

urls <- str_c(url_in, file_names)

#read and save each filein respective variable
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.
```

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

```
## 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.
```

```
global_deaths <- read_csv(urls[4])
```

```
## 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.
```

##Tranform the data

Clean and Transform data

1. Pivot each data set by all columns except Province/State, Country/Region, Lat, and Long.
2. Create new column called “date” and “cases” for the two column’s that is being pivoted.
3. Merge the two global variables and get combined global set with cases and deaths. Repeat for US data as well.

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

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

#combine global data, rename a few columns
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)`
```

```
#pivot US cases 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_))

#pivot US deaths data
US_deaths <- US_deaths %>%
  pivot_longer(cols = -(UID:Population),
               names_to = "date",
               values_to = "deaths") %>%
  select(Admin2:deaths) %>%
  mutate(date = mdy(date)) %>%
  select(-c(Lat, Long_))

#combine US data
US <- US_cases %>%
  full_join(US_deaths)
```

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

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

Add FIPS lookup table for adding population to the global data

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

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.
```

```
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)
```

##Exploratory analysis

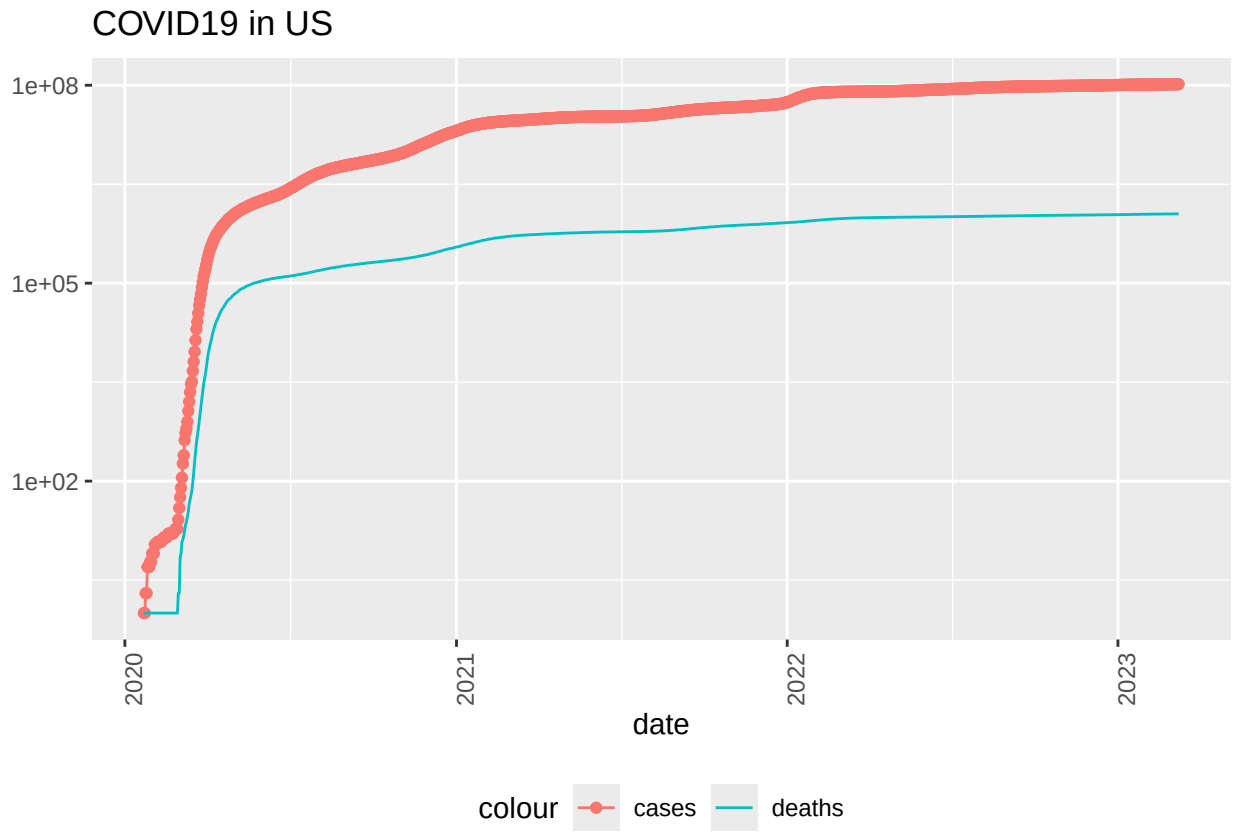
```
#visualizing data by state
US_by_state <- US %>%
  group_by(Province_State, Country_Region, date) %>%
  summarize(cases = sum(cases), deaths = sum(deaths),
            Population = sum(Population)) %>%
  mutate(deaths_per_mill = deaths *1000000 / Population) %>%
  select(Province_State, Country_Region, date, cases, deaths, deaths_per_mill, Population) %>%
  ungroup()
```

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

```
# visualizing by date
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()
```

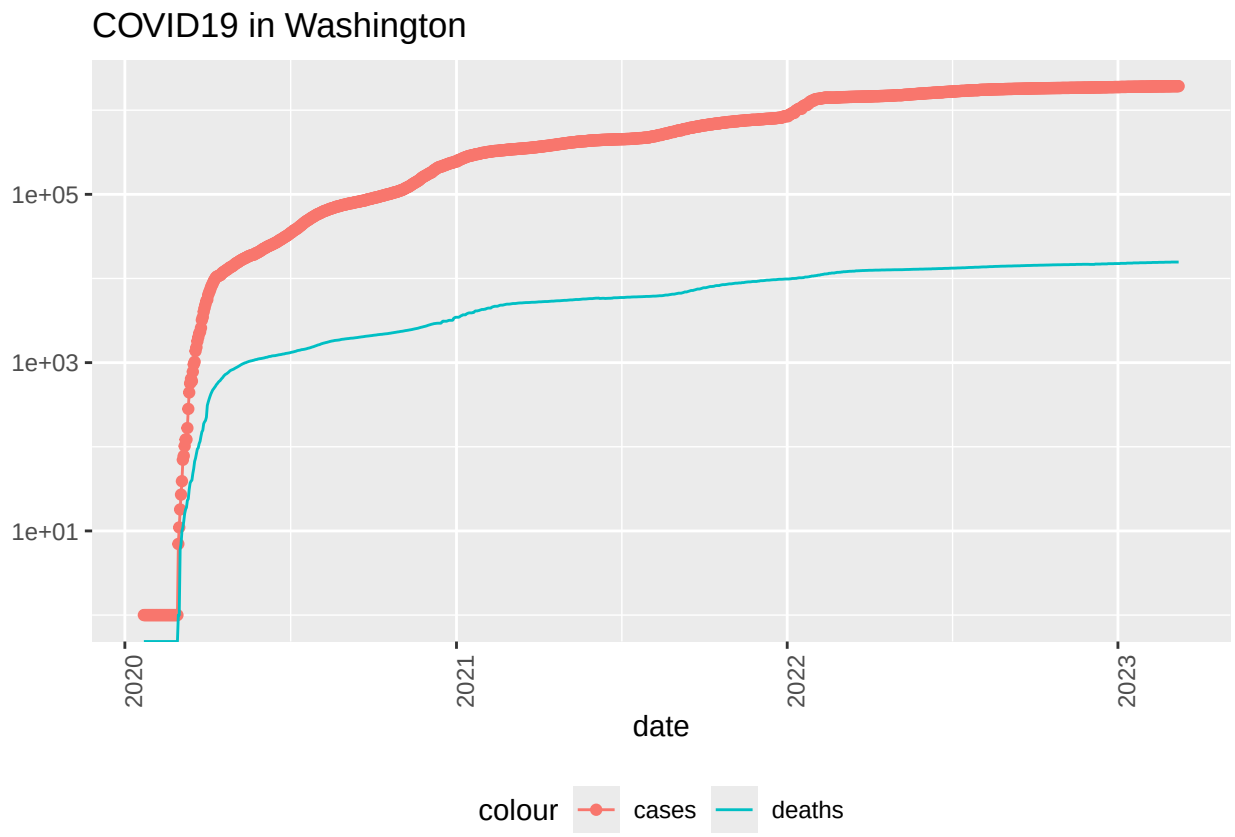
```
## `summarise()` has grouped output by 'Country_Region'. You can override using
## the `.groups` argument.
```

```
#visualize filtering where there are cases
US_totals %>%
  filter(cases > 0) %>%
  ggplot(aes(x = date, y = cases)) +
  geom_line(aes(color = "cases")) +
  geom_point(aes(color = "cases")) +
  geom_line(aes(y = deaths, color = "deaths")) +
  scale_y_log10() +
  theme(legend.position="bottom",
        axis.text.x = element_text(angle = 90)) +
  labs(title = "COVID19 in US", y= NULL)
```




```
state <- "Washington"
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")) +
  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.
```



Analysis and visualizations

```
#Add deaths/1000 to the data
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)
```

Analyze top 5

```
# Calculate the average death rate per 1000 population
average_death_rate <- mean(US_state_totals$deaths_per_thou, na.rm = TRUE)

# Sort by death rate and get the top 5 states
top_5_worst_states <- US_state_totals %>%
  arrange(desc(deaths_per_thou)) %>%
  top_n(5, wt = deaths_per_thou)
```

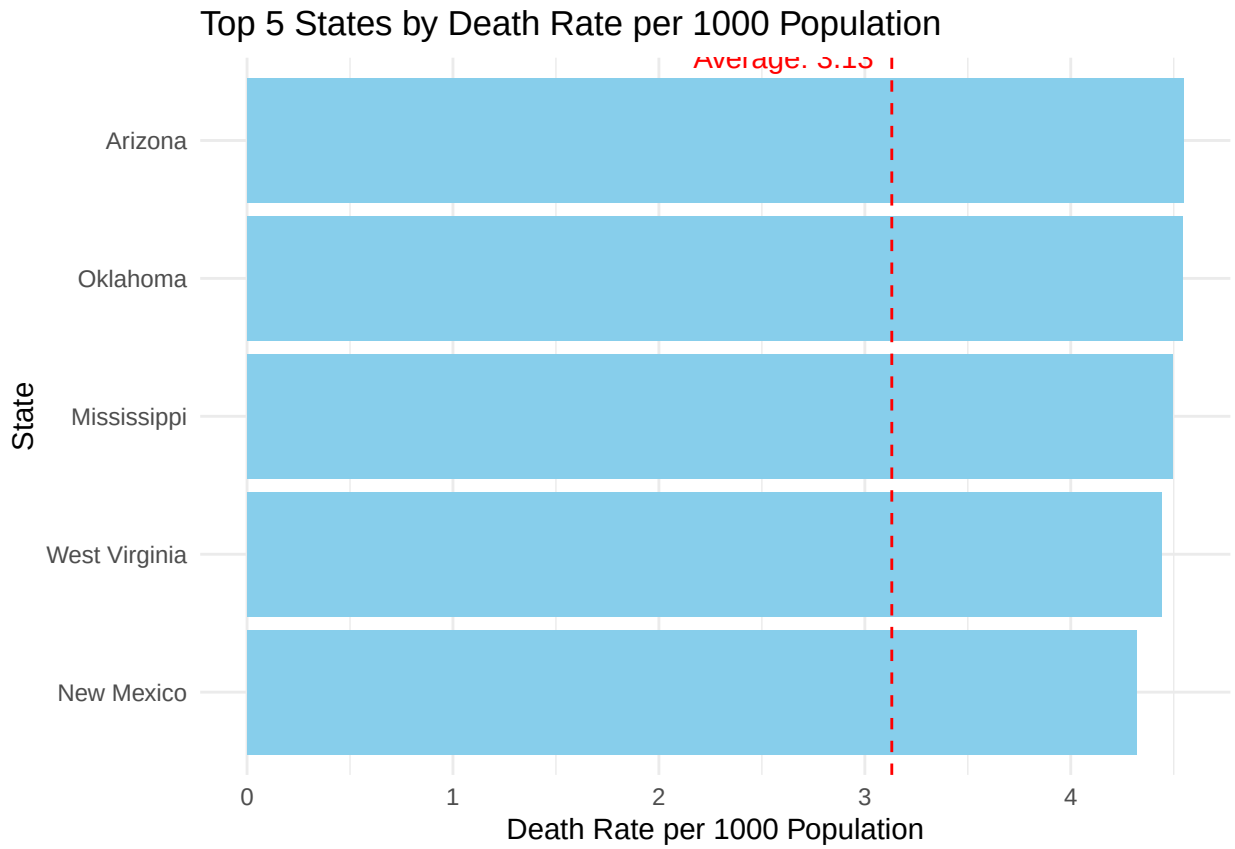
```r
# Summarize and calculate the death rate per 1000 population
US_state_totals <- US_by_state %>%
  group_by(Province_State) %>%
  summarize(deaths = max(deaths, na.rm = TRUE),
            cases = max(cases, na.rm = TRUE),
            population = max(Population, na.rm = TRUE),
            cases_per_thou = 1000 * cases / population,
            deaths_per_thou = 1000 * deaths / population) %>%
  filter(cases > 0, population > 0)

# Calculate the average death rate
average_death_rate <- mean(US_state_totals$deaths_per_thou, na.rm = TRUE)

# Get the top 5 states with the highest death rates
top_5_worst_states <- US_state_totals %>%
  arrange(desc(deaths_per_thou)) %>%
  top_n(5, wt = deaths_per_thou)
```

#Plot TOP 5 states with deaths/1000 against the average deaths/1000 in US

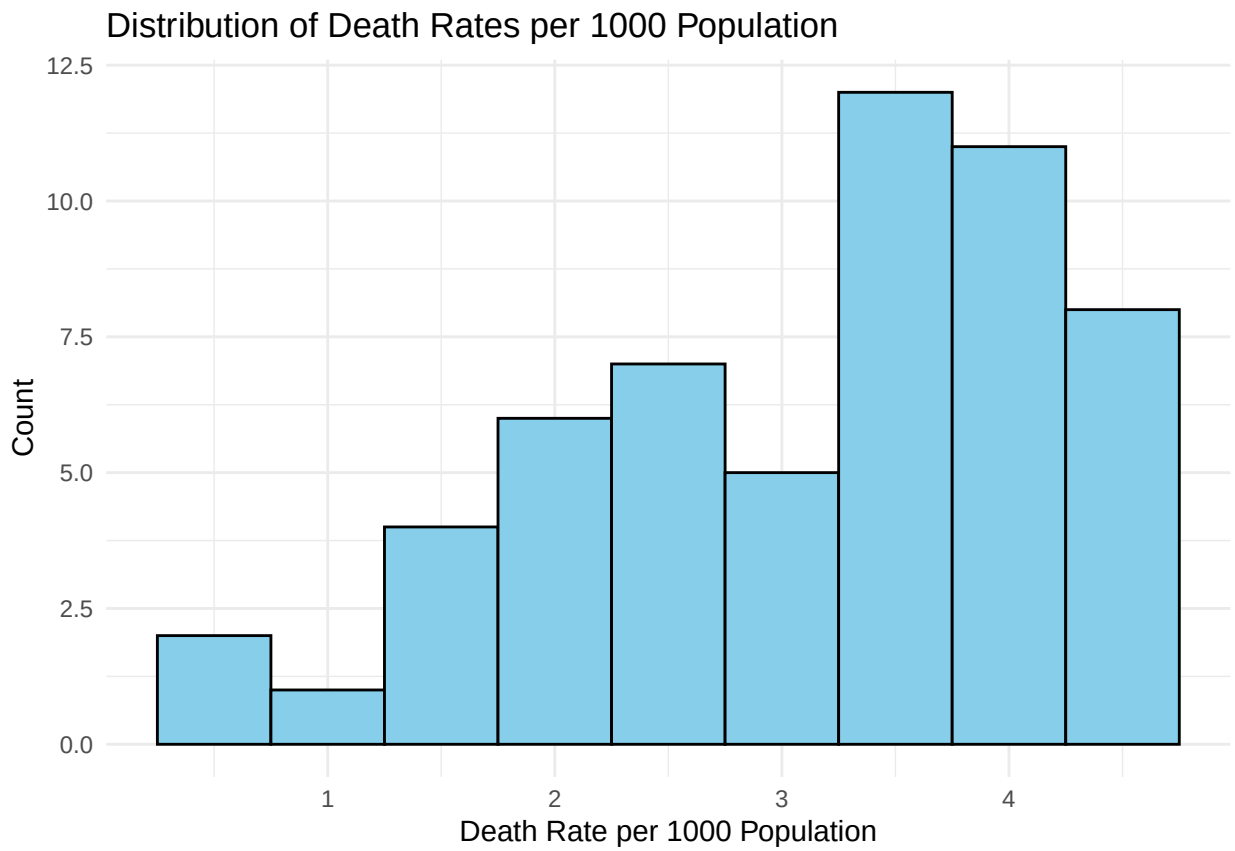
```r
# Plot the top states with the average death rate line
ggplot(top_5_worst_states, aes(x = reorder(Province_State, deaths_per_thou), y = deaths_per_thou)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  geom_hline(yintercept = average_death_rate, linetype = "dashed", color = "red") +
  coord_flip() +
  labs(title = "Top 5 States by Death Rate per 1000 Population",
       x = "State",
       y = "Death Rate per 1000 Population") +
  annotate("text", x = Inf, y = average_death_rate, label = paste("Average:", round(average_death_rate,
           hjust = 1.1, color = "red") +
  theme_minimal()
```




```
summary(US_state_totals$deaths_per_thou)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.6111  2.5294  3.4153  3.1310  3.9751  4.5478
```

```
# Plot the distribution
library(ggplot2)
ggplot(US_state_totals, aes(x = deaths_per_thou)) +
  geom_histogram(binwidth = 0.5, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Death Rates per 1000 Population",
       x = "Death Rate per 1000 Population",
       y = "Count") +
  theme_minimal()
```



##Modeling - Predict the high risk states in US by creating a high-risk variable (1 for high-risk, 0 otherwise)

```
# Define a threshold value to 3rd quarter from the anayzed data of distribution of deaths per thousand.
threshold <-  3.9751

US_state_totals <- US_state_totals %>%
  mutate(high_risk = ifelse(deaths_per_thou > threshold, 1, 0))

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
```

```
# Fit logistic regression model
logistic_model <- glm(high_risk ~ deaths_per_thou + cases_per_thou + population , data = US_state_totals
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

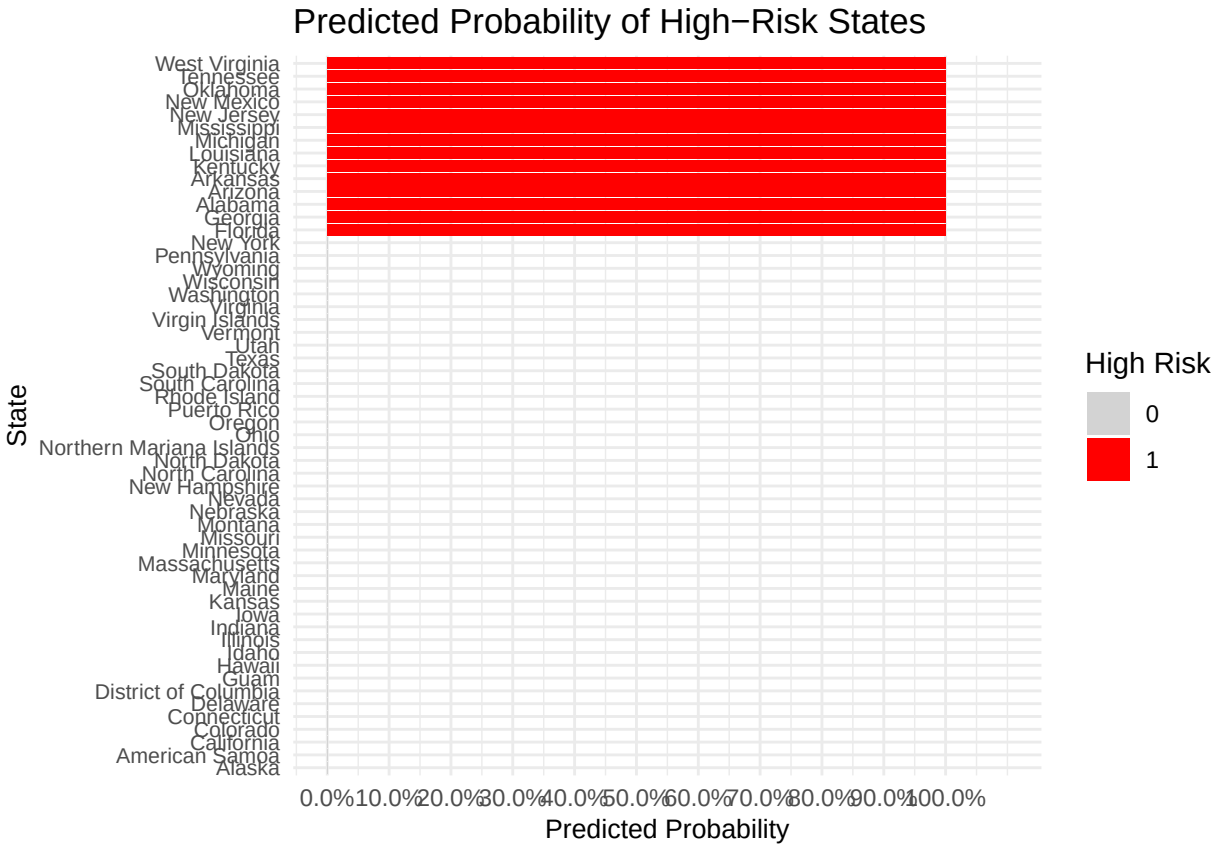
```
# Summary of the model
summary(logistic_model)
```

```
##
## Call:
## glm(formula = high_risk ~ deaths_per_thou + cases_per_thou +
##     population, family = binomial, data = US_state_totals)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -2.056e+03  3.901e+05  -0.005    0.996
## deaths_per_thou  5.241e+02  9.519e+04   0.006    0.996
## cases_per_thou   3.884e-03  3.871e+02   0.000    1.000
## population      -2.157e-06  3.008e-03  -0.001    0.999
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6.2982e+01  on 55  degrees of freedom
## Residual deviance: 6.7660e-08  on 52  degrees of freedom
## AIC: 8
##
## Number of Fisher Scoring iterations: 25
```

```
US_state_totals$predicted_prob <- predict(logistic_model, type = "response")

# Calculate the maximum value for y-axis limits and breaks
max_prob <- max(US_state_totals$predicted_prob, na.rm = TRUE)
breaks_seq <- seq(0, max_prob, by = 0.1)

ggplot(US_state_totals, aes(x = reorder(Province_State, predicted_prob), y = predicted_prob, fill = as.
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Predicted Probability of High-Risk States",
       x = "State",
       y = "Predicted Probability") +
  scale_fill_manual(values = c("lightgray", "red"), name = "High Risk") +
  scale_y_continuous(limits = c(0, max_prob * 1.1),  # Adjust y-axis limits
                     breaks = breaks_seq,  # Set custom breaks
                     labels = scales::percent_format(accuracy = 0.1)) +  # Format y-axis labels as perc
  theme_minimal() +
  theme(axis.text.y = element_text(size = 8),  # Adjust text size for better readability
        axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10))
```



Biases:

- Missing data on hospital utilization: Model does not account for how healthcare infrastructure will affect the outcomes. Availability of ICU and hospital capacity are some data points that can skew the estimates for variables (like death rates) and the prediction might be inaccurate.
- There is a over fitting in the model, so thought it may perform on the given data but can perform poorly on new, unseen data and may affect the predictive power.