Covid-19 Cases & Deaths Analysis

D.Parikh

2025-03-01

Data Origin

The data we are using today is sourced from the COVID-19 time series datasets maintained by the Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE).

These datasets provide daily cumulative counts of confirmed cases and deaths globally and in the US.

Analysis

We are going to do the following to analyse the cases and deaths data:

-> Map: Global deaths data by Country

-> Map: US death rate by State

-> Forecast Model Comparison: ARIMA vs Exponential Smoothing

First, we will start with the libraries we will use today

```r
library(tidyr)
library(stringr)
library(tidyverse)
library(lubridate)
library(forecast)
library(maps)
library(ggrepel)
```

Data Loading

Then lets load this data from the source: github.

```r
url_in <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/refs/heads/master/csse_covid_19_da

file_name <- c("time_series_covid19_confirmed_global.csv","time_series_covid19_confirmed_US.csv","time_

urls <-str_c(url_in,file_name)

global_cases <- read_csv(urls[1])
US_cases <- read_csv(urls[2])
global_deaths <- read_csv(urls[3])
US_deaths <- read_csv(urls[4])
```

Now we will look at the data specs to identify what data is provided and what kind of analysis can we present.

It can be done by using “spec(global_cases)”.

I won’t be doing it in this markdown file as the dataset consists of 100s of columns, and it will unnecessarily increase the no. of pages.

Data Cleaning

The next step in the process is Data Cleaning. As the format above suggests, the dates are all in columns, but for our analysis, we need the dates in rows and in proper format.

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

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

global_deaths <- global_deaths %>%
    mutate(date = mdy(date))

global_cases <- global_cases %>%
    mutate(date = mdy(date))

US_cases <- US_cases %>%
    pivot_longer(cols = -c(UID, iso2, iso3, code3, FIPS, Admin2, 'Province_State', 'Country_Region',Lat
    select(-c(UID, iso2, iso3, code3, FIPS, Admin2, Lat,Long_, Combined_Key))

US_deaths <- US_deaths %>%
    pivot_longer(cols = -c(UID, iso2, iso3, code3, FIPS, Admin2, 'Province_State', 'Country_Region',Lat
    select(-c(UID, iso2, iso3, code3, FIPS, Admin2, Lat,Long_, Combined_Key, Population))

US_deaths <- US_deaths %>%
    mutate(date = mdy(date))

US_cases <- US_cases %>%
    mutate(date = mdy(date))
```

Map: Global deaths data by Country

For this analysis, we will need a little more cleaning:

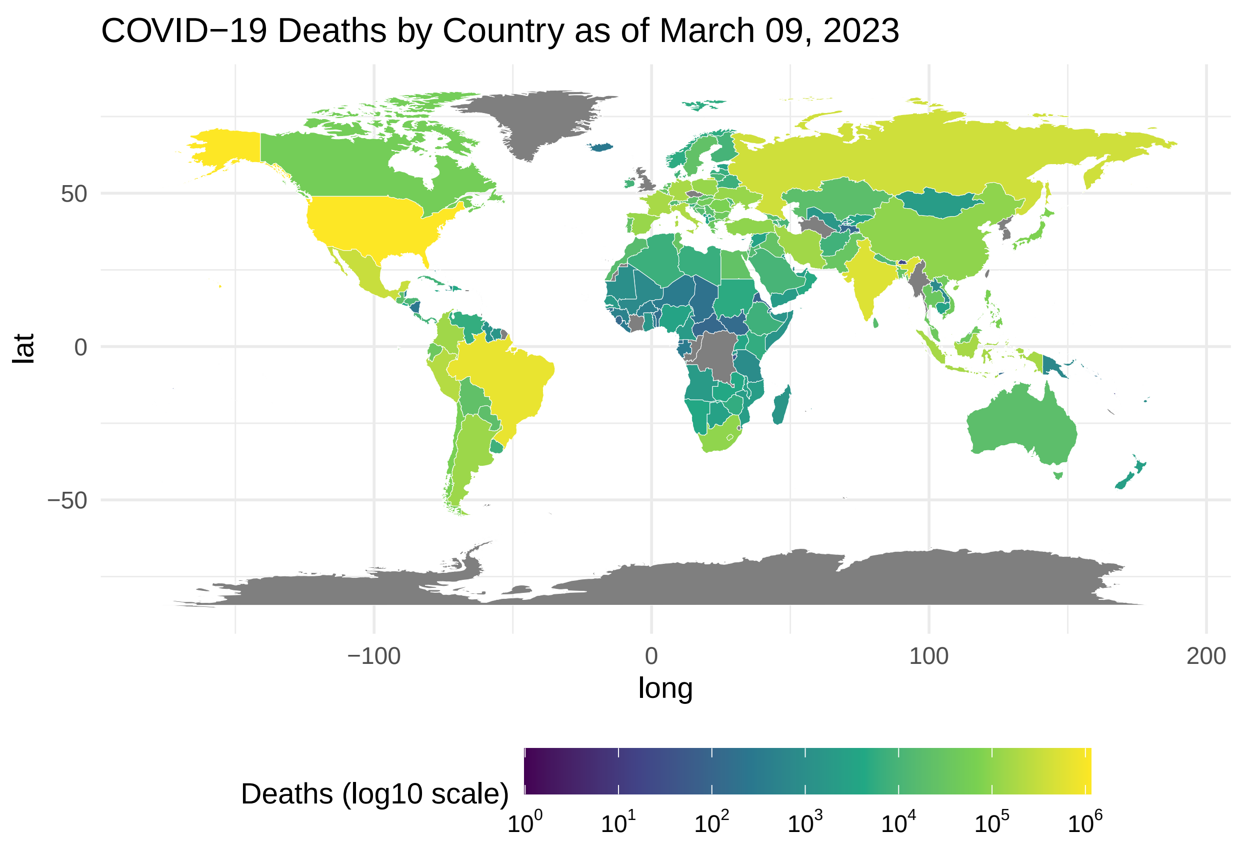
-> Global death data does not have US data, thus we will add it to the dataset.

-> The columns between the global and US dataset are named differently.

Now, we will use the combined dataset and link it with the world map.

```
US_deaths <- US_deaths %>%
  rename(`Province/State` = Province_State) %>%
  rename(`Country/Region` = Country_Region) %>%
  mutate(`Country/Region` = "USA")
```

```r
deaths_combined <- bind_rows(global_deaths, US_deaths)

deaths_combined <- deaths_combined %>%
  filter(date == max(deaths_combined$date))

deaths_by_country <- deaths_combined %>%
    group_by(`Country/Region`) %>%
    summarize(deaths = sum(deaths))%>%
    rename(Country = `Country/Region`)

world_map <- map_data("world")

world_map <- left_join(world_map,deaths_by_country, by = c("region" = "Country"))

ggplot(world_map, aes(long, lat, group = group)) +
    geom_polygon(aes(fill = deaths), color = "white", size = 0.1) +
    scale_fill_viridis_c(
        trans = "log10",
        breaks = c(1, 10, 100, 1000, 10000, 100000, 1000000),
        labels = scales::trans_format("log10", scales::math_format(10^.x)),
        limits = c(1, max(world_map$deaths, na.rm = TRUE))
    ) +
    theme_minimal() +
    labs(title = paste("COVID-19 Deaths by Country as of", format(max(global_deaths$date), "%B %d, %Y"))
         fill = "Deaths (log10 scale)") +
    theme(legend.position = "bottom",
          legend.key.width = unit(1.5, "cm"))
```



Map: US death rate by State

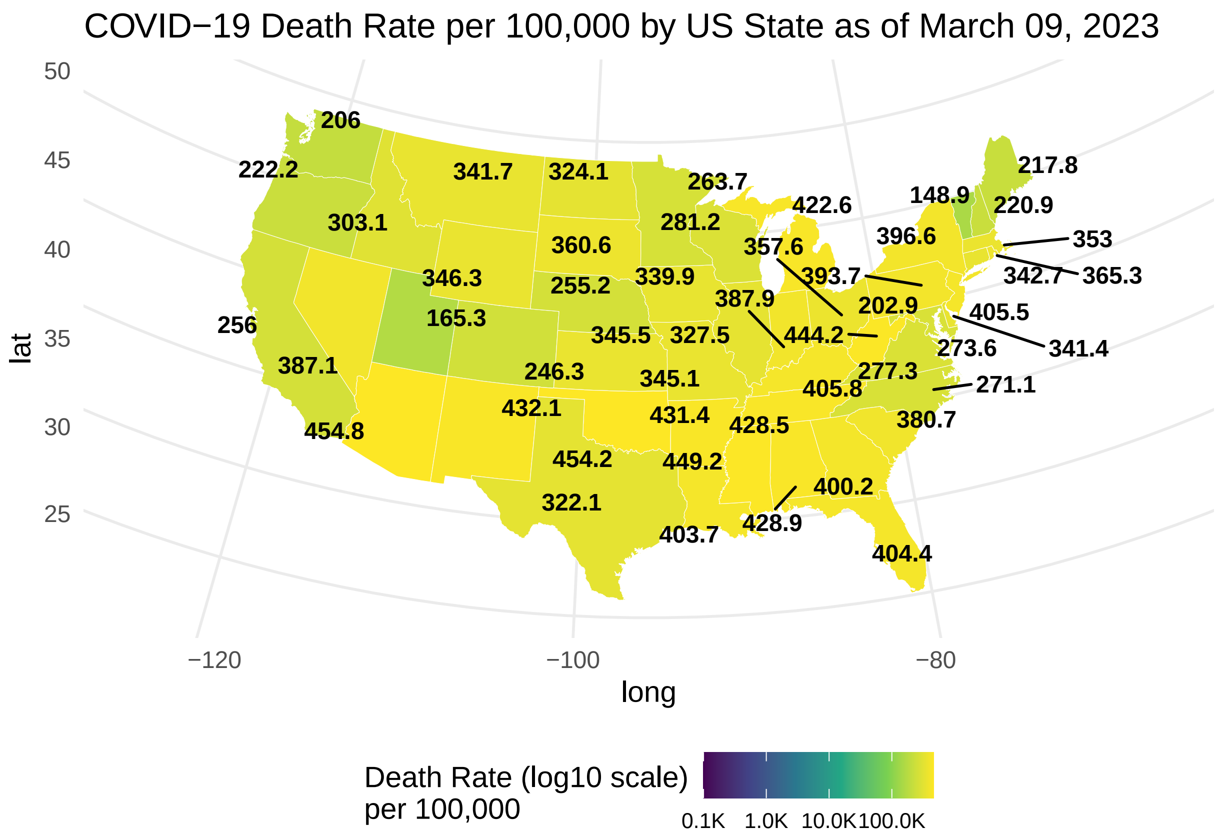
Looking at the map, US has considerable amount of deaths due to COVID-19. Let's explore it a further and see how each state did.

This time to see how each state did in terms of preventing deaths due to the virus, we will look at death rate. Death rate is just No. of Deaths/Population.

In this case, we are going to take it Death Rate per 100,000 people.

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

US_deaths <- US_deaths %>%
    pivot_longer(cols = -c(UID, iso2, iso3, code3, FIPS, Admin2, 'Province_State', 'Country_Region',Lat
    select(-c(UID, iso2, iso3, code3, FIPS, Admin2, Lat,Long_, Combined_Key))

US_deaths <- US_deaths %>%
    mutate(date = mdy(date))


US_deaths <- US_deaths %>%
    group_by(Province_State, date) %>%
    summarise(Deaths = sum(deaths), Population = sum(Population))


latest_deaths = US_deaths %>%
```

```r
  filter(date == max(date))

latest_deaths = latest_deaths %>%
  mutate(population = as.numeric(Population))

death_rates <- latest_deaths %>%
  mutate(death_rate = (Deaths / population) * 100000)

us_states <- map_data("state")

death_rates <- death_rates %>%
  mutate(state = tolower(`Province_State`))

us_states <- us_states %>%
  mutate(state = tolower(region))

us_states_map <- left_join(us_states, death_rates, by = "state")

state_centroids <- us_states_map %>%
  group_by(state) %>%
  summarize(
    centroid_long = mean(long, na.rm = TRUE),
    centroid_lat = mean(lat, na.rm = TRUE),
    death_rate = mean(death_rate, na.rm = TRUE) # Also keep death rate
  )

ggplot() + # remove the us_states_map
  geom_polygon(data = us_states_map, aes(x = long, y = lat, group = group, fill = death_rate), color =
  scale_fill_viridis_c(
    trans = "log10",
    breaks = c(0.1, 1, 10, 100, 1000, 10000),
    labels =  scales::number_format(accuracy = 0.1,
                               scale = 1,
                               suffix = "K"),
    na.value = "grey",
    limits = c(0.1, max(us_states_map$death_rate, na.rm = TRUE))
  ) +
  coord_map("albers", lat0 = 39, lat1 = 45) +
  labs(title = paste("COVID-19 Death Rate per 100,000 by US State as of", format(max(US_deaths$date), "
       fill = "Death Rate (log10 scale)\nper 100,000") +
  theme_minimal() +
  theme(legend.position = "bottom",
        legend.text = element_text(size = 8)) +

  # Add state labels
   geom_text_repel(
    data = state_centroids,
    aes(x = centroid_long, y = centroid_lat, label = round(death_rate, 1)), # Data labels, rounded
    color = "black",
    size = 3,
    fontface = "bold" # Added bold
  )
```



Forecast Model Comparison: ARIMA vs Exponential Smoothing

In our final analysis, we are going to test a couple forecast models and see which one suits the best to forecast covid-19 data.

For this exercise, we will provide data from 22nd Jan 2020 to 21st Jan 2023 as historical and forecast data from 22nd Jan 2023 to 9th March 2023.

We will then use this forecasts and compare it with actual data to see which model works better.

Models used: ARIMA & Exponential Smoothing

Note: A weekly seasonality factor is considered for ARIMA.

```r
global_cases_grouped <- global_cases %>%
  group_by(date) %>%
  summarize(total_cases = sum(cases))

train_data <- global_cases_grouped %>%
  filter(date <= as.Date("2023-01-21"))

test_data <- global_cases_grouped %>%
  filter(date > as.Date("2023-01-21") & date <=as.Date("2023-03-09"))

time_series_model<-ts(train_data$total_cases, frequency = 7)

arima_run <- auto.arima(time_series_model)
```

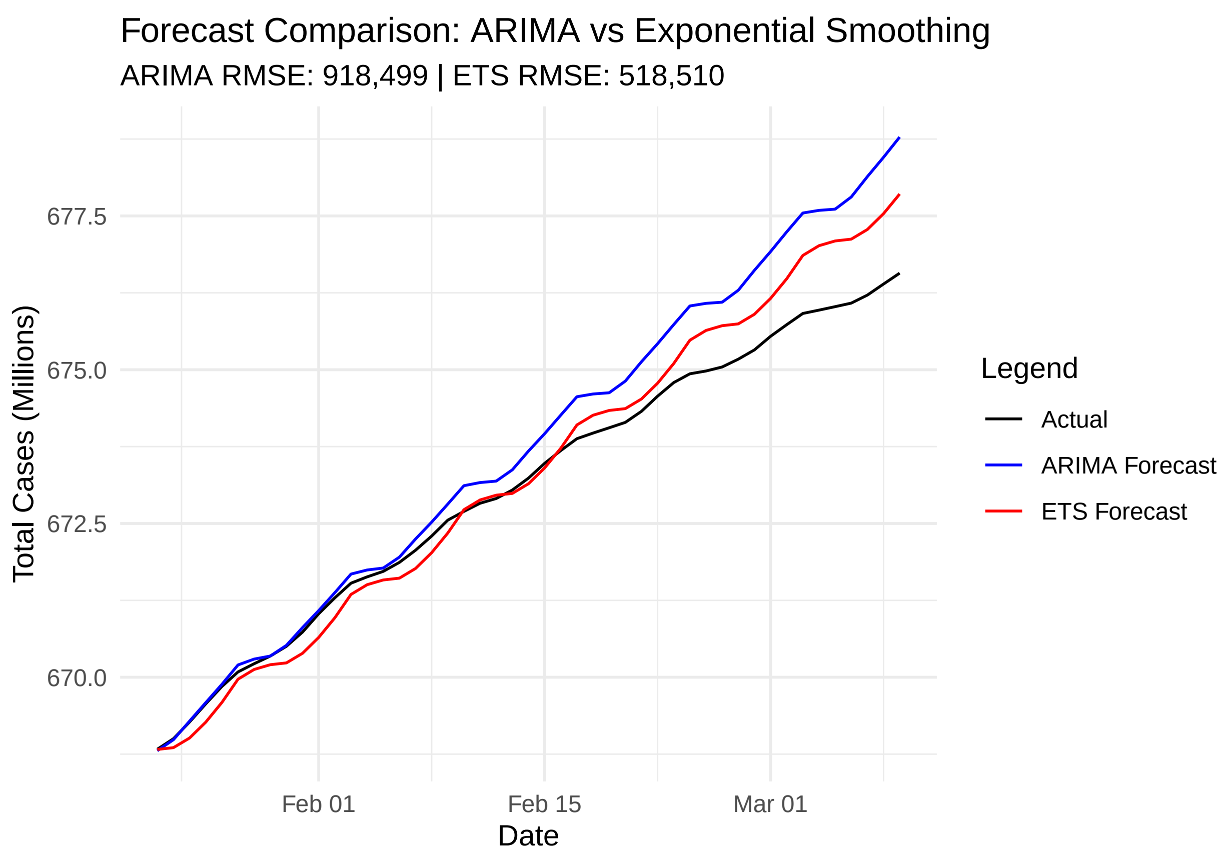
```r
ets_run <- ets(time_series_model)

cases_forecast_a <- forecast(arima_run, h=nrow(test_data))

cases_forecast_e <- forecast(ets_run, h = nrow(test_data))

comparison <- data.frame(
  date = test_data$date,
  actual = test_data$total_cases,
  arima_forecast = as.numeric(cases_forecast_a$mean),
  ets_forecast = as.numeric(cases_forecast_e$mean)
)

error_calc <- function(actual, forecast) {
  rmse <- round(sqrt(mean((actual - forecast)^2)), 0)
  return(rmse)
}

arima_rmse <-error_calc(comparison$actual, comparison$arima_forecast)
ets_rmse <- error_calc(comparison$actual, comparison$ets_forecast)


print(paste("ARIMA Model RMSE:", arima_rmse))
```

```
## [1] "ARIMA Model RMSE: 918499"
```

```r
print(paste("Exponential Smoothing Model RMSE:", ets_rmse))
```

```
## [1] "Exponential Smoothing Model RMSE: 518510"
```

```r
ggplot(comparison, aes(x = date)) +
  geom_line(aes(y = actual / 1e6, color = "Actual")) +
  geom_line(aes(y = arima_forecast / 1e6, color = "ARIMA Forecast")) +
  geom_line(aes(y = ets_forecast / 1e6, color = "ETS Forecast")) +
  labs(title = "Forecast Comparison: ARIMA vs Exponential Smoothing",
       subtitle = paste("ARIMA RMSE:", format(arima_rmse, big.mark = ","), "| ETS RMSE:", format(ets_rms
       y = "Total Cases (Millions)",
       x = "Date",
       color = "Legend") +
  theme_minimal() +
  scale_color_manual(values = c("Actual" = "black", "ARIMA Forecast" = "blue", "ETS Forecast" = "red"))
  scale_y_continuous(labels = scales::number_format(scale = 1, accuracy = 0.1))
```



```r
better_model <- if(arima_rmse < ets_rmse) "ARIMA" else "Exponential Smoothing"
print(paste("Considering Root Mean Squared Error for both models, the better model is:", better_model))
```

```
## [1] "Considering Root Mean Squared Error for both models, the better model is: Exponential Smoothing
```

Bias

Data reporting: the accuracy of data depends on reporting practices of John Hopkins CSSE.

Geographical & Population: the population and climate of different regions would significantly impact the transmission of the virus.

External factors: factors such as COVID-19 vaccine and it's distribution as well as government mandates such as lock downs would heavily impact the amount of cases ad deaths, skewing the actuals from forecast.

Data Outliers: the data may have some outliers due to the disproportionality of a region's size or incorrect entry of data.

Map : the map data from the map library, might not completely interject with the data from John Hopkins. TO solve this, I have tried to match the data values between both files so the join is accurate.

Conclusion

The visualizations offer insights into the geographical distribution of COVID-19 deaths, both globally and within the US. The time series analysis provides a basic framework for forecasting case trends, allowing for comparison between two common forecasting methods. The better performing model is determined by whichever has the lower RMSE.