Part A - covid19 Data Analysis with R

Phuoc Vinh Dat Nguyen

2024-07-06

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats 1.0.0     v stringr 1.5.1
## v purrr   1.0.2     v tibble  3.2.1
## v readr   2.1.5     v tidyr   1.3.1

## -- 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
```

```r
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(rpart)
library(rpart.plot)
```

```r
theme_set(theme_classic())
```

Part A: Topic 1 COVID-19

Task 1: Data wrangling and integration

```r
# read  and save 3 datasets
covid_data <- read.csv("Covid-data.csv")
lockdown_data <- read.csv("CountryLockdowndates.csv")
vaccine_data <- read.csv("WorldwideVaccineData.csv")
```

```r
# Display the structure of the datasets
str(covid_data)
```

```
## 'data.frame':    1575 obs. of  8 variables:
##  $ location      : chr  "Australia" "Australia" "Australia" "Australia" ...
##  $ date          : chr  "2019-12-31" "2020-01-01" "2020-01-02" "2020-01-03" ...
##  $ total_cases   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ new_cases     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ total_deaths  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ new_deaths    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ gdp_per_capita: num  44649 44649 44649 44649 44649 ...
##  $ population    : int  25499881 25499881 25499881 25499881 25499881 25499881 25499881 25499881 2549
```

```r
str(lockdown_data)
```

```
## 'data.frame':    307 obs. of  5 variables:
##  $ Country.Region: chr  "Afghanistan" "Albania" "Algeria" "Andorra" ...
##  $ Province      : chr  "" "" "" "" ...
##  $ Date          : chr  "24/03/2020" "08/03/2020" "24/03/2020" "16/03/2020" ...
##  $ Type          : chr  "Full" "Full" "Full" "Full" ...
##  $ Reference     : chr  "https://www.thestatesman.com/world/afghan-govt-imposes-lockdown-coronavirus-
```

```r
str(vaccine_data)
```

```
## 'data.frame':    187 obs. of  5 variables:
##  $ Country                           : chr  "Afghanistan" "Albania" "Algeria" "Angola" ...
##  $ Doses.administered.per.100.people: int  17 102 35 64 237 73 162 229 207 137 ...
##  $ Total.doses.administered          : num  6.45e+06 2.91e+06 1.52e+07 2.04e+07 1.06e+08 ...
##  $ X..of.population.vaccinated       : num  15 46 19 41 92 38 84 88 77 53 ...
##  $ X..of.population.fully.vaccinated: num  13 44 16 22 84 33 78 86 75 48 ...
```

We will then create a function that help convert the date (in character) to Date format

```r
convert_dates <- function(dates) {
  converted_dates <- vector("character", length(dates))
  # using regex pattern
  pattern <- "^(\\d{4})-(\\d{2})-(\\d{2})$"

  # Loop through each date to extract Y D M
  for (i in seq_along(dates)) {
    # Check if date matches the pattern
    if (grepl(pattern, dates[i])) {
      YYYY <- substr(dates[i], 1, 4)
      MM <- substr(dates[i], 6, 7)
      DD <- substr(dates[i], 9, 10)
```

```r
      # Check if MM > 12, then switch with DD
      if (as.numeric(MM) > 12) {
        temp <- MM
        MM <- DD
        DD <- temp
      }

      # Format the date into YYYY-MM-DD
      converted_dates[i] <- paste(YYYY, MM, DD, sep = "-")
    } else {
      converted_dates[i] <- NA
    }
  }

  return(converted_dates)
}
# Applying to the function to dataset
covid_data$date <- convert_dates(covid_data$date)
```

```r
# Convert Date column to Date object using lubricate library
lockdown_data$Date <- dmy(lockdown_data$Date)
# Format Date column to YYYY-MM-DD
lockdown_data$Date <- format(lockdown_data$Date, "%Y-%m-%d")
```

Taking a look at covid_data

```r
summary(covid_data)
```

```
##    location             date            total_cases        new_cases
##  Length:1575        Length:1575        Min.   :      0   Min.   :-29726
##  Class :character   Class :character   1st Qu.:     22   1st Qu.:      1
##  Mode  :character   Mode  :character   Median :  58226   Median :    205
##                                        Mean   : 180452   Mean   :   2971
##                                        3rd Qu.: 173133   3rd Qu.:   1880
##                                        Max.   :3363056   Max.   : 66625
##
##   total_deaths      new_deaths      gdp_per_capita    population
##  Min.   :      0   Min.   :-1918.0   Min.   :15309   Min.   :2.550e+07
##  1st Qu.:      0   1st Qu.:    0.0   1st Qu.:26677   1st Qu.:6.046e+07
##  Median :   2837   Median :    5.0   Median :38606   Median :6.789e+07
##  Mean   : 14060   Mean   :  183.8   Mean   :35140   Mean   :2.652e+08
##  3rd Qu.: 25100   3rd Qu.:  149.0   3rd Qu.:42201   3rd Qu.:2.075e+08
##  Max.   :135605   Max.   : 4928.0   Max.   :54225   Max.   :1.439e+09
##  NA's   :6        NA's   :7
```

Notice we have 13 NA values in covid_data, we will remove them for further analysis

```r
# Remove rows with NA values
covid_data <- covid_data[!apply(is.na(covid_data), 1, any), ]

str(covid_data)
```

```
## 'data.frame':    1564 obs. of  8 variables:
##  $ location      : chr  "Australia" "Australia" "Australia" "Australia" ...
##  $ date          : chr  "2019-12-31" "2020-01-01" "2020-01-02" "2020-01-03" ...
##  $ total_cases   : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ new_cases     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ total_deaths  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ new_deaths    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ gdp_per_capita: num  44649 44649 44649 44649 44649 ...
##  $ population    : int  25499881 25499881 25499881 25499881 25499881 25499881 25499881 25499881 2549
```

remove the "-" value in the new death and new cases. As we can see from the summary(), there are a few minimum values at $new_case and $new_dealths that having negative value.
It is probably the entry data errors. Thus we will iterate over these values and replace negative values with absolute value by remove "-"

```r
# Convert negative values to positive in new_cases and new_deaths columns
covid_data$new_cases <- abs(covid_data$new_cases)
covid_data$new_deaths <- abs(covid_data$new_deaths)
```

Notice there are a few inconsistencies in $location column

```r
unique_locations <- unique(covid_data$location)
print(unique_locations)
```

```
##  [1] "Australia"      "Australia  "    "China"          " China"
##  [5] "France"         "Iran"           "iran"           "Italy"
##  [9] "Itly"           "Spain"          "United Kingdom" "UnitedKingdom"
## [13] "United States"  "United Stats"
```

We then need to make sure these names are consistent for future analysis

```r
#create a data frame that maps inconsistent location names to standardized names
location_mapping <- data.frame(
  original = c("Australia", "Australia  ", "China", " China", "France", "Iran", "iran", "Italy", "Itly"
  standardized = c("Australia", "Australia", "China", "China", "France", "Iran", "Iran", "Italy", "Ital
)


#Create a function to standardize the location names based on the mapping
standardize_location <- function(df, column_name, mapping_df) {
  df[[column_name]] <- mapping_df$standardized[match(df[[column_name]], mapping_df$original)]
  df[[column_name]][is.na(df[[column_name]])] <- df[[column_name]][is.na(df[[column_name]])]
  return(df)
}
```

```r
covid_data <- standardize_location(covid_data, "location", location_mapping)
```

Taking a look at lockdown_data

```r
summary(lockdown_data)
```

```
##  Country.Region       Province             Date               Type
##  Length:307         Length:307         Length:307         Length:307
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##   Reference
##  Length:307
##  Class :character
##  Mode  :character
```

```r
# Count empty strings in Province and Date columns
empty_province <- sum(lockdown_data$Province == "")
```

```r
cat("Empty strings in Province column:", empty_province, "\n")
```

```
## Empty strings in Province column: 178
```

A large percentage of values in province column is missing, we should work on country level of analysis

Also, we will select the closest Date of lockdown as country lockdown_date

```r
lockdown_aggregated <- lockdown_data %>%
  group_by(Country.Region) %>%
  summarize(Date = first(na.omit(Date))) %>%
  ungroup() %>%
  rename(Country = Country.Region)
```

```r
summary(vaccine_data)
```

```
##    Country          Doses.administered.per.100.people Total.doses.administered
##  Length:187         Min.   :  0                       Min.   :1.714e+04
##  Class :character   1st Qu.: 62                       1st Qu.:1.810e+06
##  Mode  :character   Median :130                       Median :8.179e+06
##                     Mean   :131                       Mean   :6.493e+07
##                     3rd Qu.:199                       3rd Qu.:2.865e+07
##                     Max.   :343                       Max.   :3.408e+09
##  X..of.population.vaccinated X..of.population.fully.vaccinated
##  Min.   : 0.10               Min.   : 0.10
##  1st Qu.:36.50               1st Qu.:29.00
##  Median :62.00               Median :55.00
##  Mean   :56.91               Mean   :51.94
##  3rd Qu.:80.00               3rd Qu.:75.00
##  Max.   :99.00               Max.   :99.00
```

We will only need to change the name of the column as assessment required

```r
vaccine_data <- vaccine_data %>%
  rename(
    Country = Country,
    `Doses Administered` = Doses.administered.per.100.people,
    `Total Doses Administered` = Total.doses.administered,
    `% of Population Vaccinated` = X..of.population.vaccinated,
    `% of Population Fully Vaccinated` = X..of.population.fully.vaccinated
  )

# Check the cleaned data
str(vaccine_data)
```

```
## 'data.frame':    187 obs. of  5 variables:
##  $ Country                         : chr  "Afghanistan" "Albania" "Algeria" "Angola" ...
##  $ Doses Administered              : int  17 102 35 64 237 73 162 229 207 137 ...
##  $ Total Doses Administered        : num  6.45e+06 2.91e+06 1.52e+07 2.04e+07 1.06e+08 ...
##  $ % of Population Vaccinated      : num  15 46 19 41 92 38 84 88 77 53 ...
##  $ % of Population Fully Vaccinated: num  13 44 16 22 84 33 78 86 75 48 ...
```

```r
summary(vaccine_data)
```

```
##    Country          Doses Administered Total Doses Administered
##  Length:187         Min.   :  0        Min.   :1.714e+04
##  Class :character   1st Qu.: 62        1st Qu.:1.810e+06
##  Mode  :character   Median :130        Median :8.179e+06
```

```
##                        Mean   :131        Mean   :6.493e+07
##                        3rd Qu.:199        3rd Qu.:2.865e+07
##                        Max.   :343        Max.   :3.408e+09
##  % of Population Vaccinated % of Population Fully Vaccinated
##  Min.   : 0.10              Min.   : 0.10
##  1st Qu.:36.50              1st Qu.:29.00
##  Median :62.00              Median :55.00
##  Mean   :56.91              Mean   :51.94
##  3rd Qu.:80.00              3rd Qu.:75.00
##  Max.   :99.00              Max.   :99.00
```

As requirement, we want the joined dataset includes these columns:
location, date, total_cases, new_cases, total_deaths, new_deaths, gdp_per_capita, population, lockdown_date, total doses administered, % of population fully vaccinated.

As these data will be used for visualization and predictive analysis on country level data, we will need the left join method to integrate 3 data sets based on location. Left join is selected because we need all records from covid_data but only those records from dataset lockdown_aggregated and vaccine_data whose key (location) is contained in L

Before that, we will need to check if lockdown_aggregated and vaccine_data having the same country name with these unique country from covid_data

```r
unique_locations <- unique(covid_data$location)
print(unique_locations)
```

```
## [1] "Australia"      "China"          "France"         "Iran"
## [5] "Italy"          "Spain"          "United Kingdom" "United States"
```

```r
# fix 'US' to 'United States' in lockdown_aggregated
lockdown_aggregated$Country[lockdown_aggregated$Country == "US"] <- "United States"

# fix 'U.K.' to 'United Kingdom' in vaccine_data
vaccine_data$Country[vaccine_data$Country == "U.K."] <- "United Kingdom"
```

We will then rename the common ID which is location before join the datasets

```r
colnames_vaccine_data <- colnames(vaccine_data)
print(colnames_vaccine_data)
```

```
## [1] "Country"                          "Doses Administered"
## [3] "Total Doses Administered"         "% of Population Vaccinated"
## [5] "% of Population Fully Vaccinated"
```

```r
library(dplyr)

# Ensure consistent column names
covid_data <- covid_data %>%
  rename(Country = location)

lockdown_data <- lockdown_data %>%
  rename(Country = Country.Region)


# Perform left joins
joined_data <- covid_data %>%
  left_join(lockdown_aggregated, by = "Country") %>%
  left_join(vaccine_data, by = "Country") %>%
```

```r
  select(
    Country, date, total_cases, new_cases, total_deaths, new_deaths,
    gdp_per_capita, population, Date,
    'Total Doses Administered','% of Population Fully Vaccinated'
  )

# Rename Country back to location for consistency with the desired column names
joined_data <- joined_data %>%
  rename(location = Country,
         lockdown_date = Date)

# taking a look at a joined_dataset
str(joined_data)
```

```
## 'data.frame':    1564 obs. of  11 variables:
##  $ location                        : chr  "Australia" "Australia" "Australia" "Australia" ...
##  $ date                            : chr  "2019-12-31" "2020-01-01" "2020-01-02" "2020-01-03" ...
##  $ total_cases                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ new_cases                       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ total_deaths                    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ new_deaths                      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ gdp_per_capita                  : num  44649 44649 44649 44649 44649 ...
##  $ population                      : int  25499881 25499881 25499881 25499881 25499881 25499881 25499
##  $ lockdown_date                   : chr  "2020-03-24" "2020-03-24" "2020-03-24" "2020-03-24" ...
##  $ Total Doses Administered        : num  5.8e+07 5.8e+07 5.8e+07 5.8e+07 5.8e+07 ...
##  $ % of Population Fully Vaccinated: num  86 86 86 86 86 86 86 86 86 86 ...
```

```r
summary(joined_data)
```

```
##    location             date            total_cases        new_cases
##  Length:1564        Length:1564        Min.   :      0   Min.   :    0.0
##  Class :character   Class :character   1st Qu.:     18   1st Qu.:    1.0
##  Mode  :character   Mode  :character   Median :  57866   Median :  209.5
##                                        Mean   : 180674   Mean   : 3005.2
##                                        3rd Qu.: 174112   3rd Qu.: 1876.5
##                                        Max.   :3363056   Max.   :66625.0
##
##   total_deaths      new_deaths     gdp_per_capita    population
##  Min.   :     0   Min.   :   0.0   Min.   :15309   Min.   :2.550e+07
##  1st Qu.:     0   1st Qu.:   0.0   1st Qu.:34272   1st Qu.:4.675e+07
##  Median :  2774   Median :   5.0   Median :38606   Median :6.527e+07
##  Mean   : 14074   Mean   : 186.6   Mean   :35173   Mean   :2.646e+08
##  3rd Qu.: 25217   3rd Qu.: 149.0   3rd Qu.:44649   3rd Qu.:1.457e+08
##  Max.   :135605   Max.   :4928.0   Max.   :54225   Max.   :1.439e+09
##
##  lockdown_date      Total Doses Administered % of Population Fully Vaccinated
##  Length:1564        Min.   : 57988175        Min.   :67.00
##  Class :character   1st Qu.: 95245599        1st Qu.:70.00
##  Mode  :character   Median :146731437        Median :79.00
##                     Mean   :190908057        Mean   :77.58
##                     3rd Qu.:149957751        3rd Qu.:86.00
##                     Max.   :597655035        Max.   :86.00
##                     NA's   :195              NA's   :195
```

Change the date formate for $date and $lockdown_date

```r
joined_data$date <- as.Date(joined_data$date)
joined_data$lockdown_date <- as.Date(joined_data$lockdown_date)
```

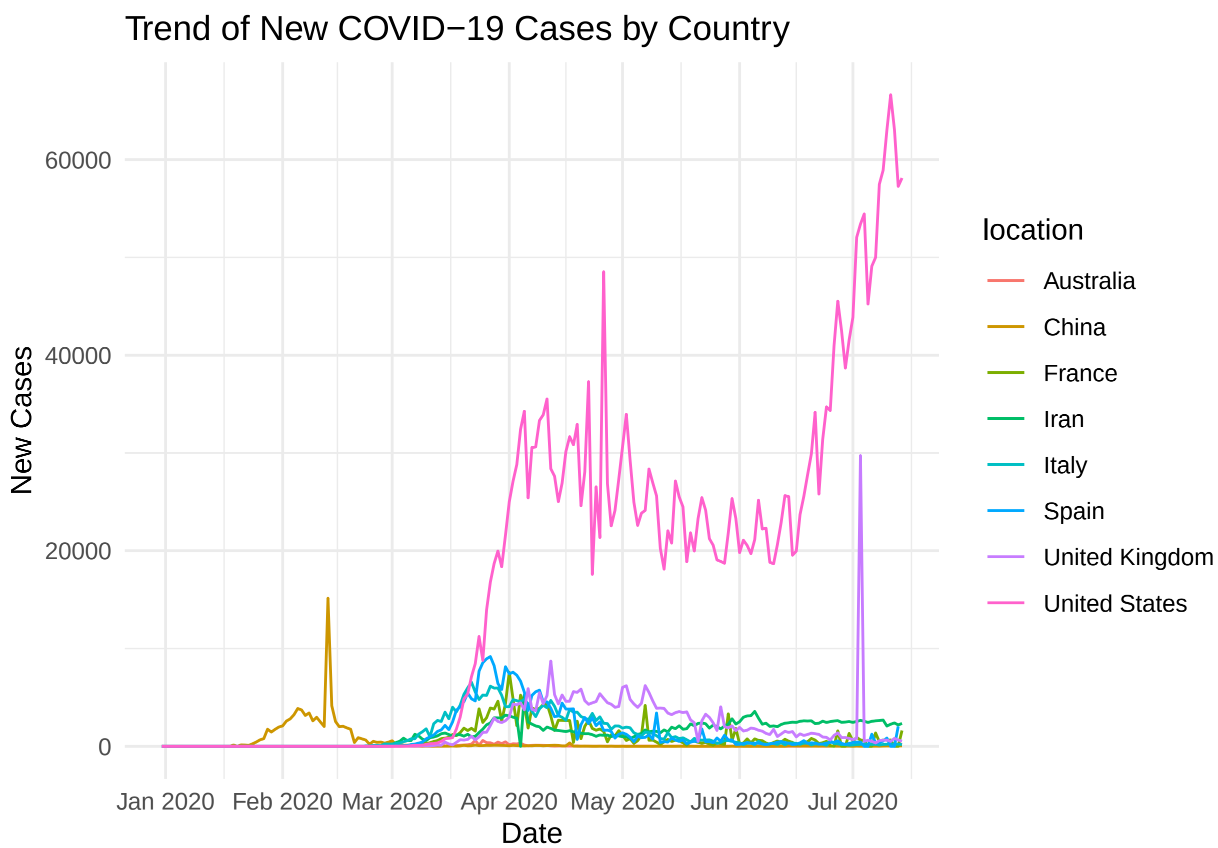
```r
# Check for duplicates in joined_data
duplicates <- joined_data[duplicated(joined_data), ]

# Print the duplicate rows if any
if (nrow(duplicates) > 0) {
  print("Duplicate rows found:")
  print(duplicates)
} else {
  print("No duplicate rows found.")
}
```

```
## [1] "No duplicate rows found."
```

Task 2: Data visualization and analysis

Let's first plot new COVID-19 cases across all countries.

```r
library(ggplot2)

#Plotting new cases trend for each country
ggplot(data = joined_data, aes(x = date, y = new_cases, group = location, color = location)) +
  geom_line() +
  scale_x_date(date_labels = "%b %Y", date_breaks = "1 month") +
  labs(title = "Trend of New COVID-19 Cases by Country",
       x = "Date", y = "New Cases") +
  theme_minimal()
```

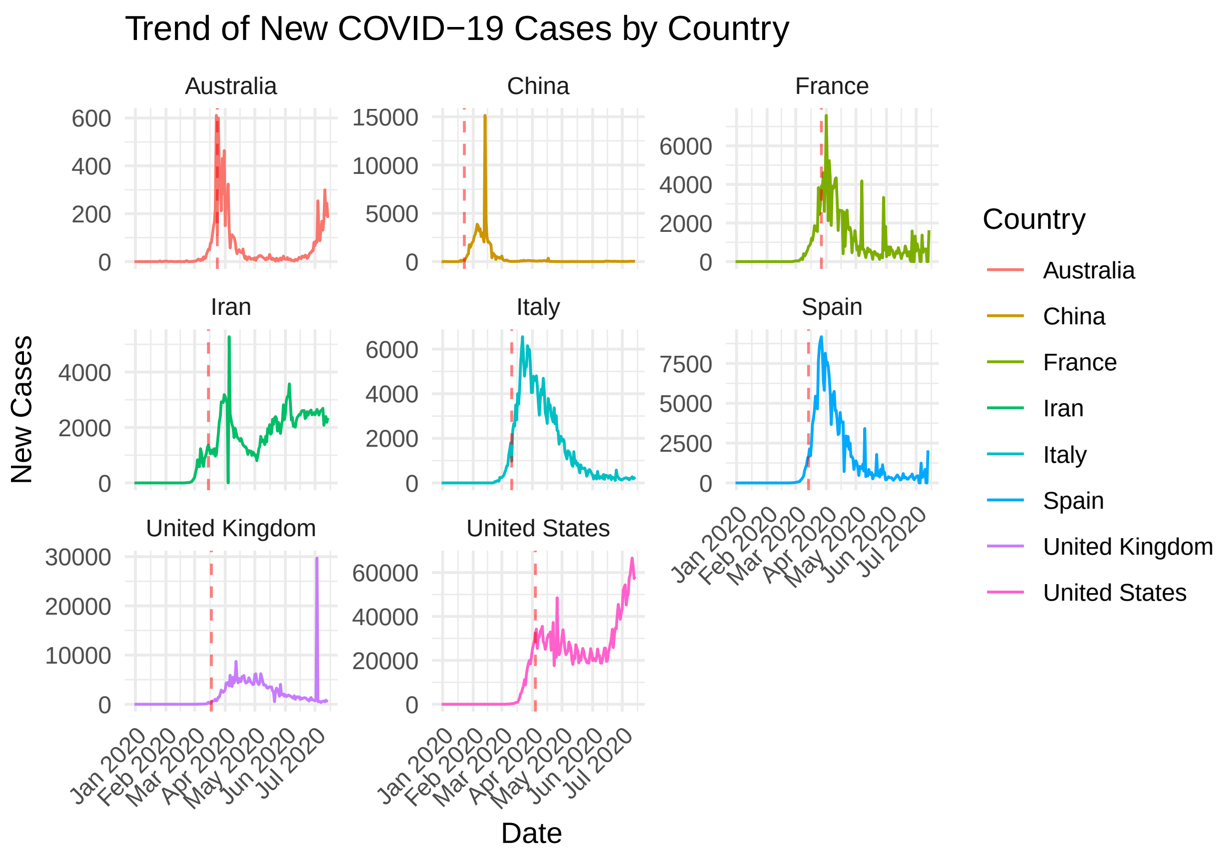


The plot reveals varying trends in new cases from different countries. Some countries saw the peaks following by declines and others have fluctuating patterns.

Many countries saw an initial surge in covid cases as the globe started to be aware of the virus speading. The subsequent fluctuations in new cases could be affected by testing capacity and the emergence of new variants can not be detected by current testing equipment.

Let's plot each country separately with the affect of lockdown_date to find out any other information.

```
#Plotting new cases trend for each country with lockdown dates
ggplot(data = joined_data, aes(x = date, y = new_cases, group = location, color = location)) +
  geom_line() +
  scale_x_date(date_labels = "%b %Y", date_breaks = "1 month") +
  labs(title = "Trend of New COVID-19 Cases by Country",
       x = "Date", y = "New Cases") +
  theme_minimal() +
  facet_wrap(~ location, scales = "free_y", ncol = 3) +
  geom_vline(data = joined_data, aes(xintercept = as.numeric(as.Date(lockdown_date))),
             linetype = "dashed", color = "red", alpha = 0.5) +  # Add vertical lines for lockdown date
  guides(color = guide_legend(title = "Country")) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Let's delve into the spike events on the plots and compare the trends among countries

Australia:

- Peak at March to April 2020: Australia experienced a peak in new cases just before the first lock down were announced
- As the government implemented a strict lock down, social distance and travel restriction, the number of cases drop significantly
- The tail of plot experience another spike during June 2021 as the new waves of covid cases spreading as Australia enter the second lock down period

China:

- Peak at around February 2020, China was the origin of COVID-19 where saw a sharp rise in cases a few weeks after lock down.
- The sharp spike could be the result from bulk reporting the new number of cases
- Thank to the very strict lock down in Wuhan and other areas, the country brought the number of cases down rapidly

France:

- Peak around March-April 2020: France experienced the first wave of COVID-19 in Europe
- The number of cases reduced significantly due to lock down restriction. However, the relaxations allowed smaller peaks still in the following months

Iran:

- Peak is in March 2020, Iran experienced an outbreak and have placed the restriction immediately. However, the ease of restriction and public gatherings during the period of Ramadan and Nowruz celebration led to the sharp spike in late April 2020
- The sharp spike could be the result from bulk reporting the new number of cases

Italy and Spain

- Peak in March 2020, Italy and Spain are two of the European countries with high number of cases. With the high number of cases, these countries implement a strict lock down and healthcare intervention helped bringing down the number of cases gradually

UK:

- The UK experienced a peak at late April while the lock down was placed in March. The restriction significantly reduced the number of cases and the early response shows that spike is not as sharp as other countries.
- The spike in July 2020 is unusual and the internet could not provide any important event that happened in UK during that time. Therefore, it could be a false data were filled in and we need to remove this value for further analysis purpose

```r
# Remove row 1357 from joined_data which contains a abnormal value
joined_data <- joined_data[-1357, ]
```

US:

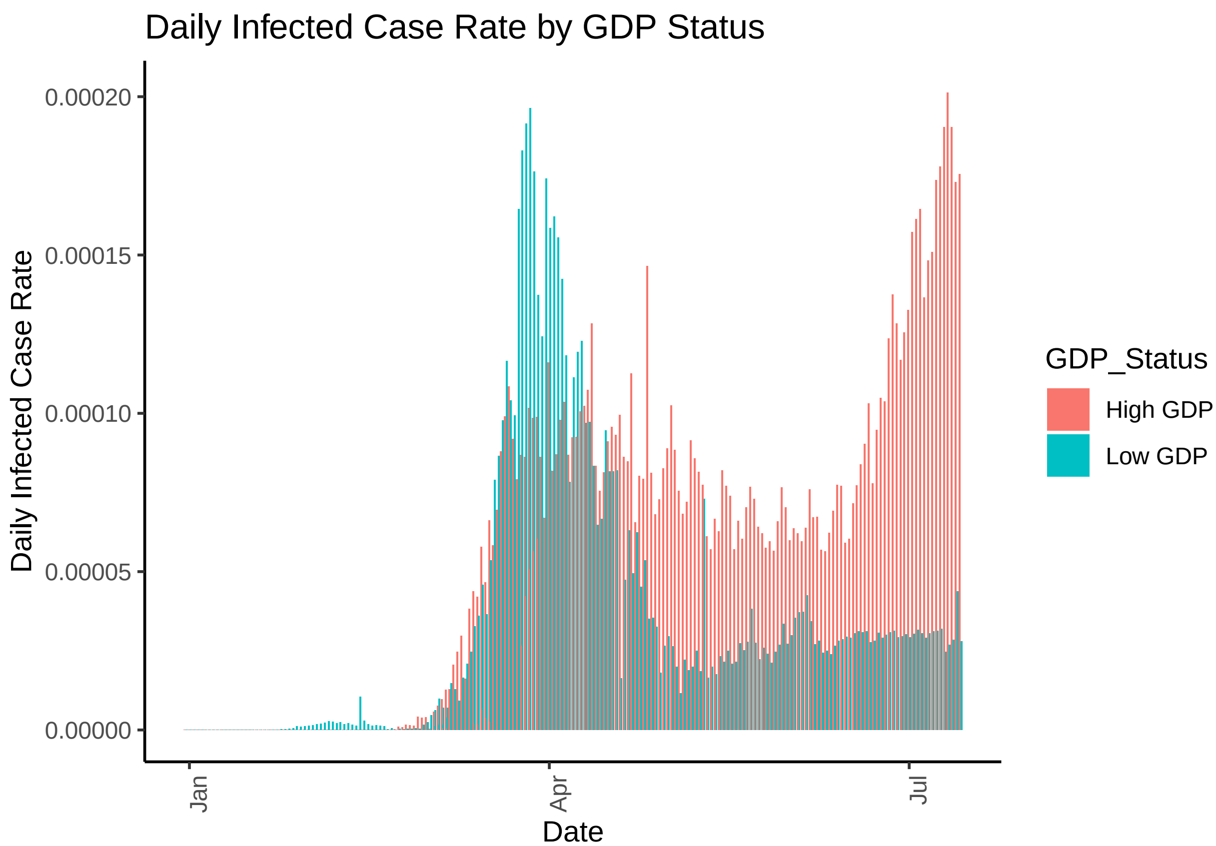
- Peak in April 2020 after a lock down, US is a country with largest number of cases across all period, it was driven by outbreaks in big cities with high density like New York and New Orleans.
- US experienced second, larger peak in June 2020 due to the early lift of restrictions, some States decided to reopen and different response to the outbreak in each State led to this surge.
- The number of cases continued to to Regional Outbreaks in late 2020. The trends in the US were influenced by many factors such as state-level policies, healthcare system, public behaviors and social beliefs

Next, we want to observe the relationship between the death rate or new case numbers with the GDP of each country.

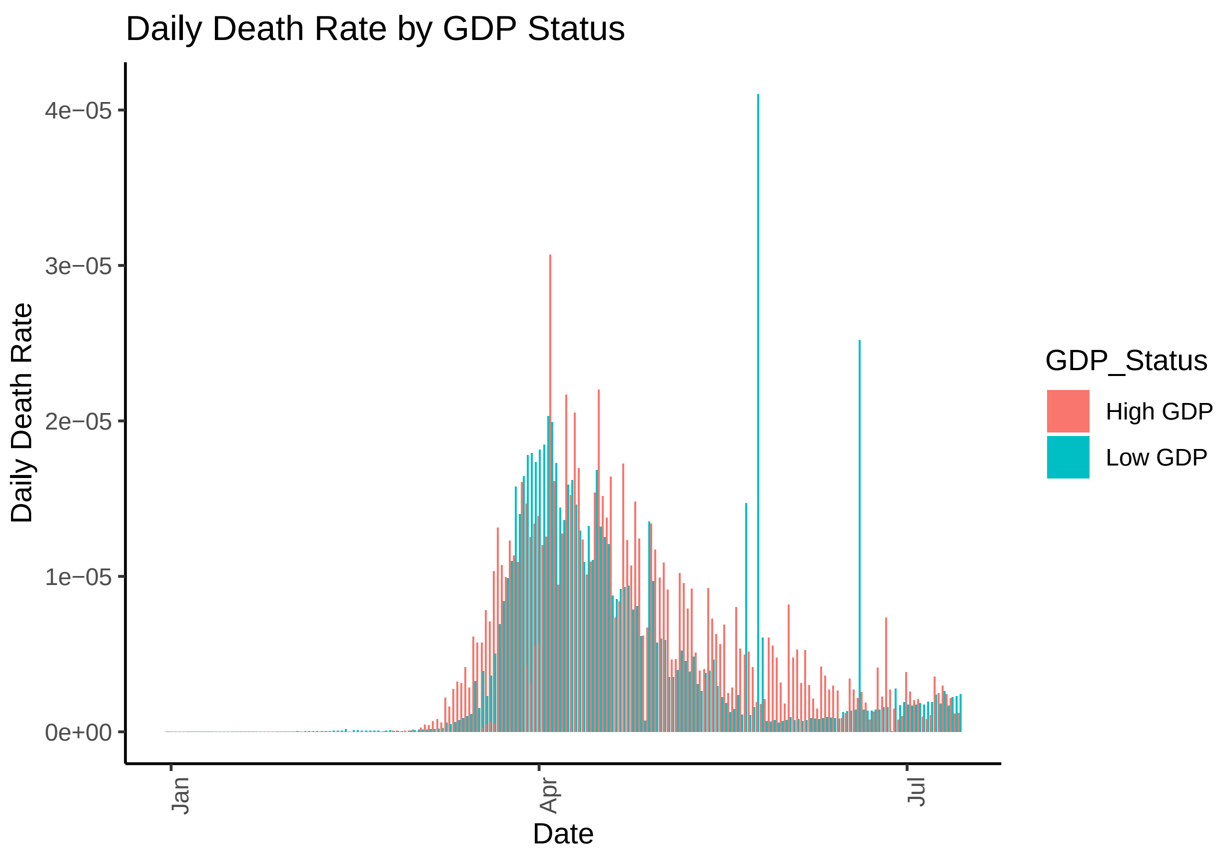
```r
# calculation of the mean GDP to split status
mean_gdp <- mean(joined_data$gdp_per_capita, na.rm = TRUE)

#Add a column 'GDP_Status'
joined_data <- joined_data %>%
  mutate(GDP_Status = ifelse(gdp_per_capita >= mean_gdp, "High GDP", "Low GDP"))

#Calculate the daily infected case rate and daily death rate
joined_data <- joined_data %>%
  mutate(daily_infected_case_rate = new_cases / population,
         daily_death_rate = new_deaths / population)
```

```r
# Bar plot for daily infected case rate
infected_case_plot <- ggplot(joined_data, aes(x = date, y = daily_infected_case_rate, fill = GDP_Status
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Daily Infected Case Rate by GDP Status",
       x = "Date",
       y = "Daily Infected Case Rate") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

# Bar plot for daily death rate
```

```
death_rate_plot <- ggplot(joined_data, aes(x = date, y = daily_death_rate, fill = GDP_Status)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Daily Death Rate by GDP Status",
       x = "Date",
       y = "Daily Death Rate") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))


print(infected_case_plot)
```




```
print(death_rate_plot)
```



Daily Infected Case Rate by GDP Status: Both groups suffered from high number of new cases from the initial surge starting around end of March and April. While the lower GPD group had a higher number of daily infected case rate at the beginning, the higher GDP group experienced a greater peak after the significant increase in June. The new cases rate is mostly lower in Lower GDP group with a sharp decrease after the initial surge

Daily Death Rate by GDP Status: Both groups suffered from high number of new death at the beginning of the pandemic. We can see both group had managed to decease the number of death rate since the initial surge. It is noticeably that the lower GDP group has a sharper decline compare to the higher GPD group

Interpretation of daily Infected Case Rate: There are potential factors that led to the number of newly infected cases is higher in high GDP countries

- Testing capacity: compare to lower GDP group, the higher GDP group is having the better healthcare facilities and testing capacity which could quickly identify the new cases emerged
- Restriction policy: as per the above analysis in each countries, the higher GPD group generally had place a relaxation earlier than lower GPD group which created an opportunity for new cases spreading for example US and France. Some lower GDP countries such as China also placed a very strict isolation rule which led to the sharp reduction in new cases
- Lock down Date: low GPD country placed restriction quite early on which help reduce the spreading of the new cases effectively

Interpretation of Death Rate: The higher GDP group experienced rapid increase in death rate at the initial surge and lower to decrease this rate due to potential reasons

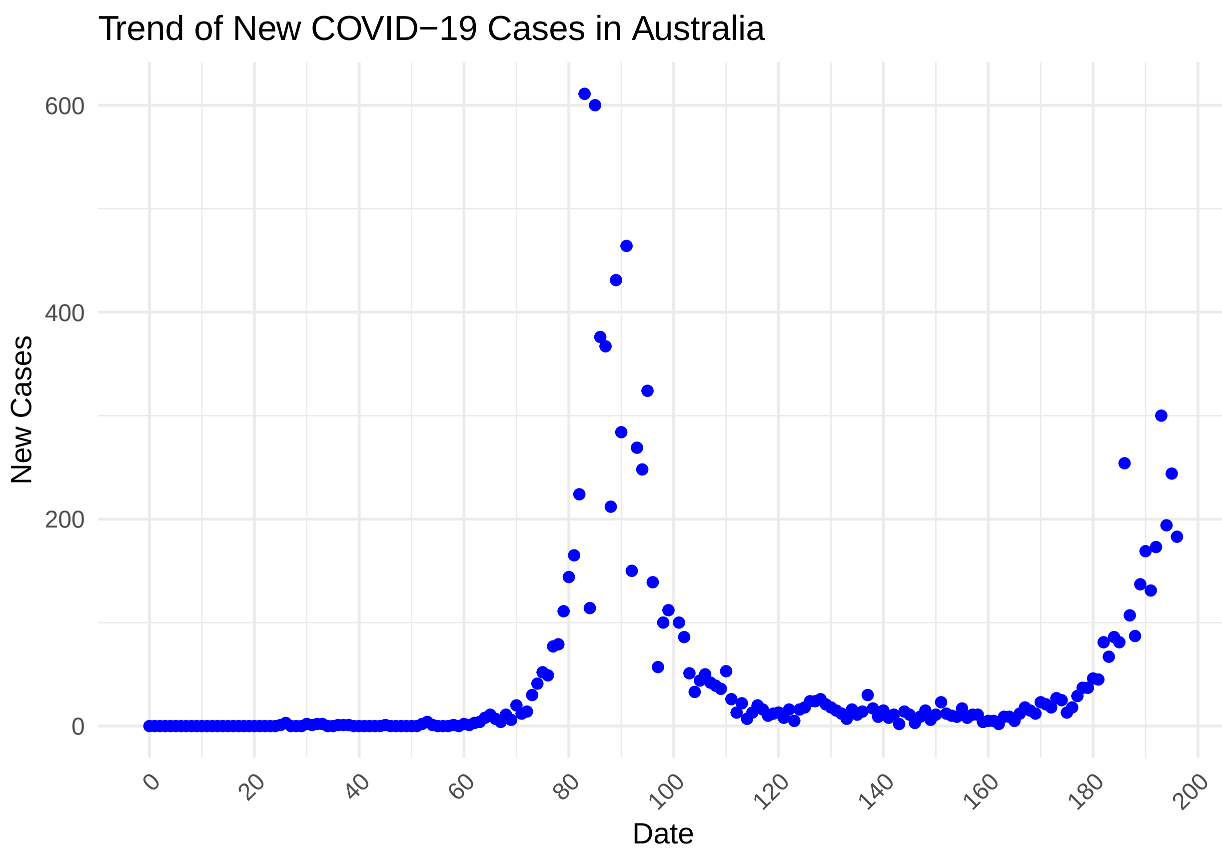
- Number of Infected cases: the number of new cases in high GDP group is higher compare to low GPD group which led to more death rate

- Demographics: High GDP countries is more likely to have significant more older population who are vulnerable to the infection of COVID-19 which led to more death rate
- There are a few sharp spikes in the death rate in low GPD countries can be inconsistency in reporting as these spikes could reflect a delay or bulk reporting rather than the actual death rate per day

Task 3: Predictive data analysis

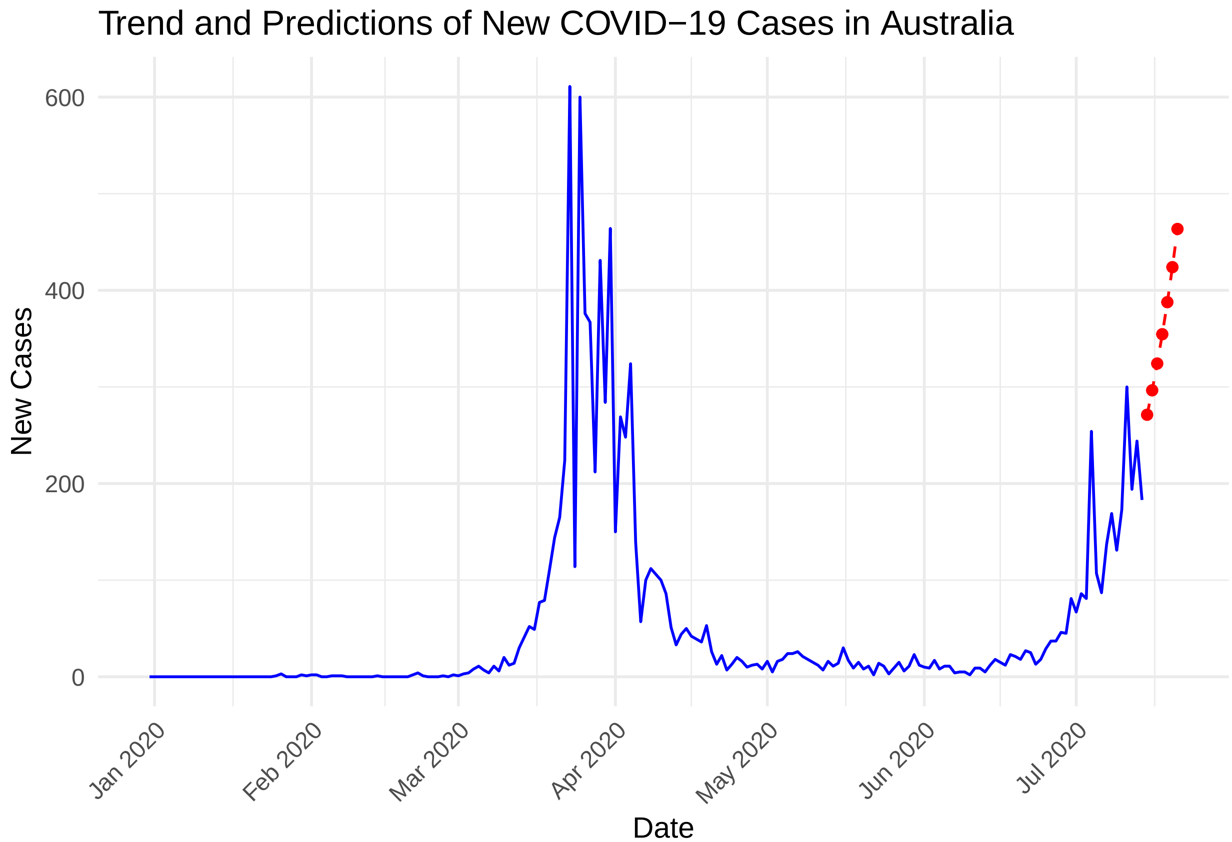
We will use **Australia data** for this predictive task just because most of us are living in Australia.

```
# Filter the data for Australia only
australia_data <- joined_data %>% filter(location == "Australia")

# Convert dates to numeric starting from 0 for poly model
australia_data$numeric_date <- as.numeric(australia_data$date - min(australia_data$date))

# Plot the data
ggplot(australia_data, aes(x = numeric_date, y = new_cases)) +
  geom_point(color = "blue") +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10), labels = scales::date_format("%b %Y")) +
  labs(title = "Trend of New COVID-19 Cases in Australia",
       x = "Date", y = "New Cases") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



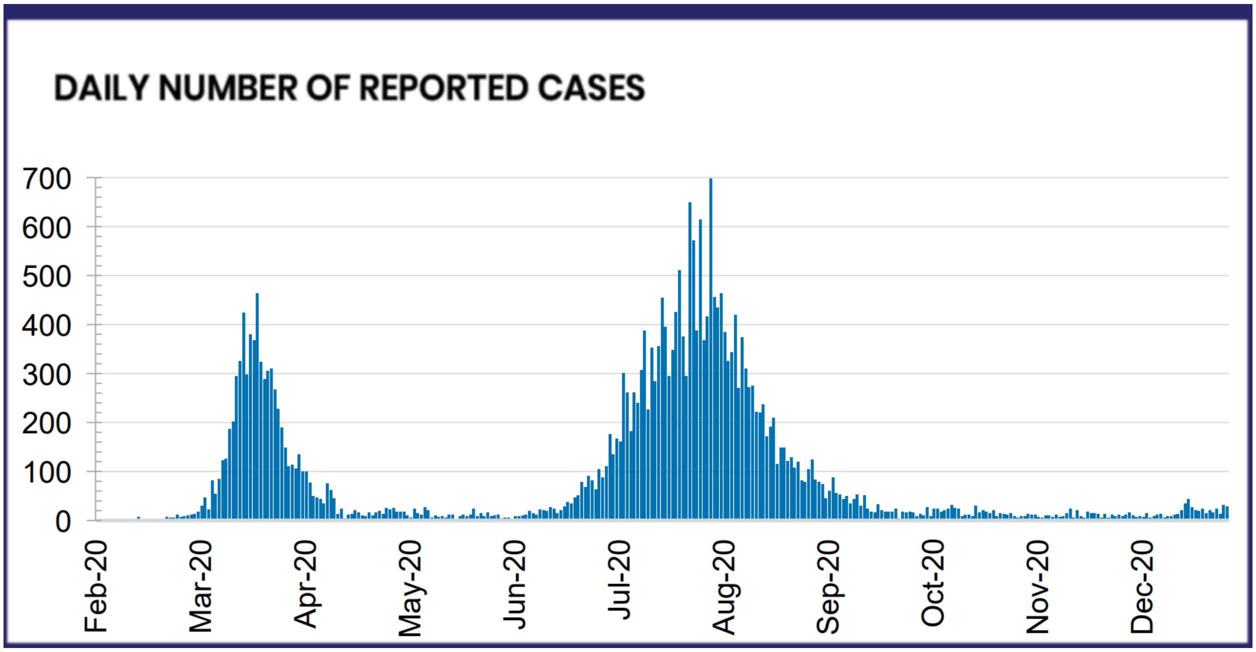
We will then make predictions on the newly infected case number or death cases for the next five to seven days

```r
# Fit a polynomial regression model
model <- lm(new_cases ~ poly(numeric_date, 7, raw = TRUE), data = australia_data)

# Generate 7 dates
last_date <- max(australia_data$date)
future_dates <- seq.Date(from = last_date + 1, by = "day", length.out = 7)
future_numeric_dates <- as.numeric(future_dates - min(australia_data$date))

# Predict new case
future_predictions <- predict(model, newdata = data.frame(numeric_date = future_numeric_dates))

# Combine future dates with predictions so we can plot the prediction data
future_predictions_df <- data.frame(date = future_dates, new_cases = future_predictions)

# Plot the data and the predictions
ggplot(australia_data, aes(x = date, y = new_cases)) +
  geom_line(color = "blue") +
  geom_point(data = future_predictions_df, aes(x = date, y = new_cases), color = "red") +
  geom_line(data = future_predictions_df, aes(x = date, y = new_cases), color = "red", linetype = "dashe
  scale_x_date(date_labels = "%b %Y", date_breaks = "1 month") +
  labs(title = "Trend and Predictions of New COVID-19 Cases in Australia",
       x = "Date", y = "New Cases") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Below is the Daily Number Reported Cases graph retrieved from: https://www.health.gov.au/sites/default/

files/documents/2020/12/coronavirus-covid-19-at-a-glance-31-december-2020_0.pdf



The prediction **seems following the actual number** according to the trend of second wave of COVID-19 approaching Australia

Apply training and testing splits to evaluate predictions

We will use 90:10 Split for balancing the representativeness of training and testing data as we only have 196 instances for the task. We also need to select day randomly instead of pick 10% continuing chunk of time to avoid data loss

```r
# Split the data into training and test set with an 90:10 split ratio
set.seed(123)
training.samples <- australia_data %>%
  dplyr::select(date) %>%
  mutate(id = row_number()) %>%
  sample_frac(0.9) %>%
  pull(id)

train.data <- australia_data[training.samples, ]
test.data <- australia_data[-training.samples, ]


# Make predictions
train.data$predictions <- predict(model, newdata = train.data)
test.data$predictions <- predict(model, newdata = test.data)

# Compute performance metrics
performance_train <- data.frame(
  RMSE = RMSE(train.data$predictions, train.data$new_cases),
  R2 = R2(train.data$predictions, train.data$new_cases)
)

performance_test <- data.frame(
  RMSE = RMSE(test.data$predictions, test.data$new_cases),
```

```
  R2 = R2(test.data$predictions, test.data$new_cases)
)


# Visualize the performance on both sets
ggplot() +
  geom_line(data = train.data, aes(x = date, y = new_cases), color = "blue") +
  geom_line(data = train.data, aes(x = date, y = predictions), color = "red", linetype = "dashed") +
  geom_line(data = test.data, aes(x = date, y = new_cases), color = "green") +
  geom_line(data = test.data, aes(x = date, y = predictions), color = "purple", linetype = "dashed") +
  scale_x_date(date_labels = "%b %Y", date_breaks = "1 month") +
  labs(title = "Model Performance on Training and Test Sets",
       x = "Date", y = "New Cases") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

Apply K-fold Cross-Validation

```
#10-fold cross-validation
set.seed(123)
train_control <- trainControl(method = "cv", number = 10)
cv_model <- train(new_cases ~ poly(date, 7, raw = TRUE), data = australia_data,
                  method = "lm", trControl = train_control)

# Cross-validated performance metrics
performance_cv <- data.frame(
```

```r
  RMSE = cv_model$results$RMSE,
  R2 = cv_model$results$Rsquared
)

print("Performance on Training Set:")
```

```
## [1] "Performance on Training Set:"
```

```r
print(performance_train)
```

```
##       RMSE        R2
## 1 76.99168 0.4524675
```

```r
print("Performance on Test Set:")
```

```
## [1] "Performance on Test Set:"
```

```r
print(performance_test)
```

```
##       RMSE        R2
## 1 42.52664 0.5779517
```

```r
print("Performance with K-fold Cross-Validation:")
```

```
## [1] "Performance with K-fold Cross-Validation:"
```

```r
print(performance_cv)
```

```
##       RMSE        R2
## 1 87.97917 0.1872814
```

Model Performance Analysis

1. **Accuracy:** we used 2 metrics RMSE and R2 to evaluation the prediction. However, RMSE and R2 indicate that the prediction is **not very accurate**.
2. **Under-fitting:** comparing the metrics between 2 models, the model is more likely over-fitting which is why it shows better performance on that test set. There are quite a few reasons could explain why the model is under fitting:
   - Model complexity: our model is not appropriate may not capture the pattern of the training dataset
   - Data instances: we only have 196 instances for the training and testing which is a very small dataset which do not give enough data to feed into a model. Although the metrics on test set looks better, it might be because the test set is more representative to the over all data value in the dataset
3. **K-fold cross validation:** the model evaluated across 10 different folds should provide a more robust estimate of the model performance. Yes ! the K-fold CV can help reduce the overfitting issue because it used all data for training and testing on each of the 10 folds.
4. **Add Data:** adding more data could be an advantage to train a better model. The current data is highly skewed which make it hard for the model to learn the underlying patterns.
5. **Model Choice:** we could try different polynomial degree to find the better model for future prediction

Part A: Topic 2 Olympic Tweets

Task 1: Data wrangling, visualization and analysis

We will read and save the dataset and taking a look at what dataset we are working on

```r
# Read the CSV file
olympics_tweets <- read.csv("Olympics_tweets.csv")

str(olympics_tweets)
```

```
## 'data.frame':    114213 obs. of  13 variables:
##  $ id              : num  1.42e+18 1.42e+18 1.42e+18 1.42e+18 1.42e+18 ...
##  $ text            : chr  "Mirabai Chanu's maiden Olympic silver helped India clinch joint-12th spot
##  $ user_screen_name: chr  "ITGDsports" "dseu_official" "SutejKumar4" "NsChandapaka" ...
##  $ user_location   : chr  "Noida- India" NA "New Delhi- India" "Nalgonda- Telangana" ...
##  $ retweet_count   : int  1 0 0 1 0 0 0 1 0 0 ...
##  $ favorited       : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ favorite_count  : int  1 1 0 0 1 0 0 2 0 0 ...
##  $ user_description: chr  "Live cricket scores- news- analysis and fun facts on all your favourite sp
##  $ user_created_at : chr  "26/09/2017 10:57" "12/11/2020 9:01" "20/04/2020 16:56" "8/08/2017 1:26" .
##  $ user_followers  : int  9333 1168 72 5907 2721 1 61 7247 180 771 ...
##  $ user_friends    : int  28 4 143 60 353 84 405 185 9 1071 ...
##  $ date            : chr  "24/07/2021 15:52" "24/07/2021 15:52" "24/07/2021 15:52" "24/07/2021 15:53
##  $ language        : chr  "en" "en" "en" "en" ...
```

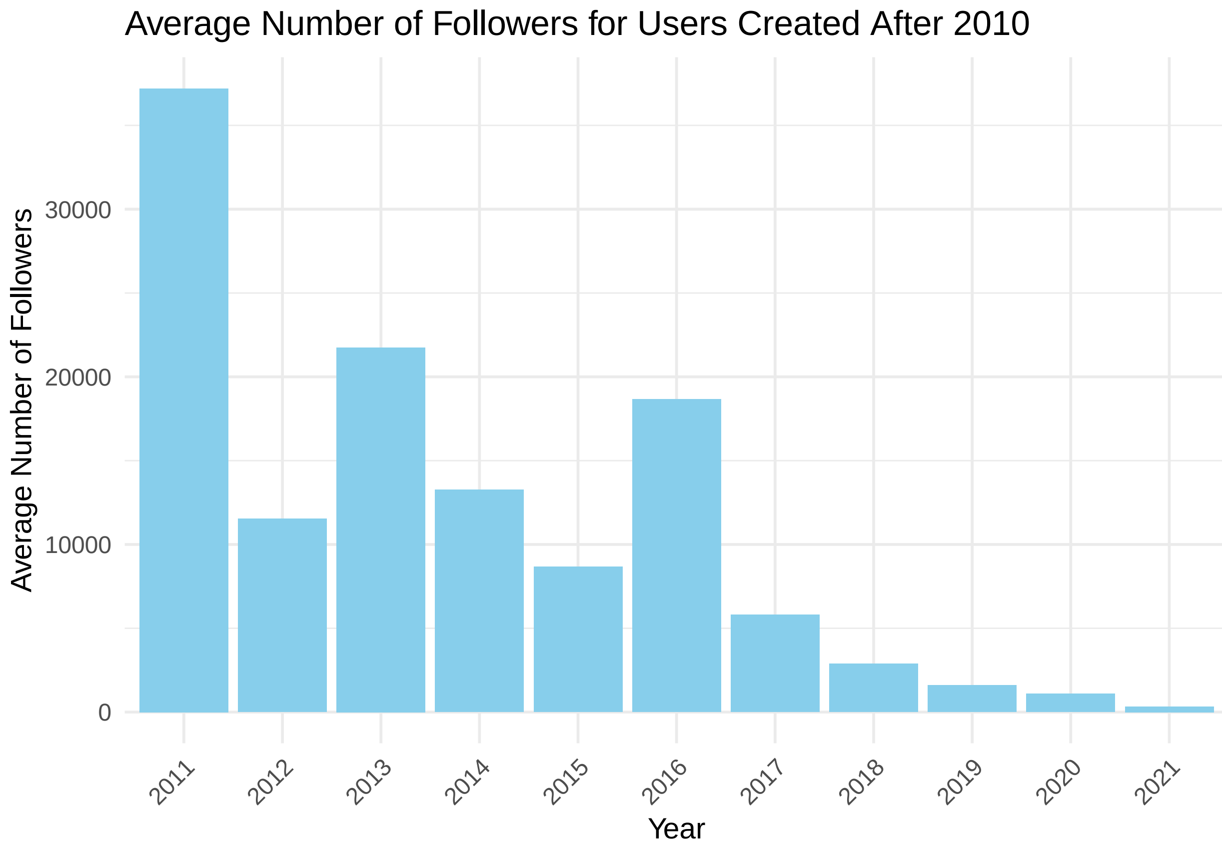
```r
summary(olympics_tweets)
```

```
##        id                text           user_screen_name   user_location     
##  Min.   :1.419e+18   Length:114213      Length:114213      Length:114213     
##  1st Qu.:1.420e+18   Class :character   Class :character   Class :character  
##  Median :1.420e+18   Mode  :character   Mode  :character   Mode  :character  
##  Mean   :1.420e+18                                                           
##  3rd Qu.:1.421e+18                                                           
##  Max.   :1.422e+18                                                           
##                                                                              
##  retweet_count       favorited       favorite_count    user_description  
##  Min.   :   0.0000   Mode :logical   Min.   :   0.00   Length:114213     
##  1st Qu.:   0.0000   FALSE:114210    1st Qu.:   0.00   Class :character  
##  Median :   0.0000   NA's :3         Median :   0.00   Mode  :character  
##  Mean   :   0.3502                   Mean   :   2.13                     
##  3rd Qu.:   0.0000                   3rd Qu.:   1.00                     
##  Max.   :2404.0000                   Max.   :9572.00                     
##  NA's   :3                           NA's   :3                           
##  user_created_at    user_followers      user_friends         date          
##  Length:114213      Min.   :       0   Min.   :     0   Length:114213     
##  Class :character   1st Qu.:     106   1st Qu.:   179   Class :character  
##  Mode  :character   Median :     456   Median :   498   Mode  :character  
##                     Mean   :   92012   Mean   :  1429                     
##                     3rd Qu.:    2007   3rd Qu.:  1188                     
##                     Max.   :78061816   Max.   :772902                     
##                     NA's   :9          NA's   :9                          
##    language        
##  Length:114213     
##  Class :character  
##  Mode  :character  
```

```
##
##
##
##
```

To extract the YEAR info and create a require plot, we will convert $user_created_at to Date-Time format. Then we will extract the YEAR and save it to a new variable.

```r
# Convert user_created_at to Date-Time
olympics_tweets$user_created_at <- dmy_hm(olympics_tweets$user_created_at)

# extract the year and add it as a new variable
olympics_tweets$user_created_at_year <- year(olympics_tweets$user_created_at)

# Filter data for users created after 2010
filtered_data <- olympics_tweets %>%
  filter(user_created_at_year > 2010)

# Calculate the average number of user_followers for each year
average_followers_per_year <- filtered_data %>%
  group_by(user_created_at_year) %>%
  summarise(average_followers = mean(user_followers, na.rm = TRUE))

# we plot the average number of user_followers across different year
ggplot(average_followers_per_year, aes(x = factor(user_created_at_year), y = average_followers)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Average Number of Followers for Users Created After 2010",
       x = "Year", y = "Average Number of Followers") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# Count occurrences of different location values, excluding NA
location_counts <- olympics_tweets %>%
  filter(!is.na(user_location)) %>%
  count(user_location, sort = TRUE)

# Display the top 10 most frequent location values
top_10_locations <- location_counts %>%
  top_n(10, n)

# Print the top 10 locations and their counts
print(top_10_locations)
```

```
##       user_location    n
## 1             India 1172
## 2   London- England 1087
## 3            London 1067
## 4     United States  822
## 5         Australia  583
## 6   Los Angeles- CA  577
## 7    United Kingdom  572
## 8       New York- NY  547
## 9  New Delhi- India  520
## 10          she/her  486
```

```
# Calculate the total number of tweets associated with the top 10 locations
total_tweets_top_10 <- sum(top_10_locations$n)
```

```r
# Print the total number of tweets
print(total_tweets_top_10)
```

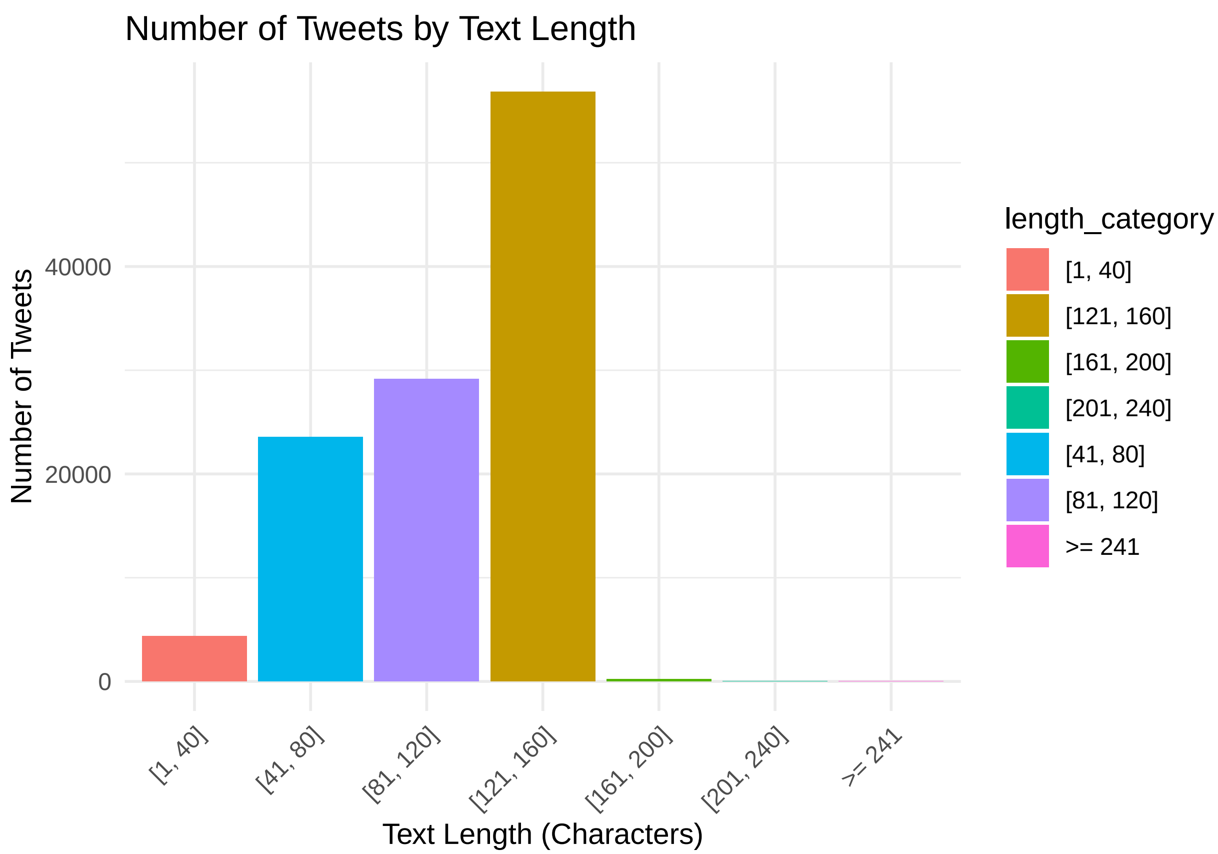
```
## [1] 7433
```

Odd values from the data frame:

- **she/her:** this is not a location but gender preference which is irrelevant in the dataset
- **Inconsistency:** some locations are city names, or city within a country (London, Los ANgeles - CA, New Delhi- India) while some contains only countries (India, Australia, . . .).

Calculate the length of the text contained in each tweet (measured in characters) and produce a bar chart to show the number of tweets of the following length

```r
# calculation of the length of the text in each tweet
olympics_tweets <- olympics_tweets %>%
  mutate(text_length = nchar(text))

# Define length categories
olympics_tweets <- olympics_tweets %>%
  mutate(length_category = case_when(
    text_length >= 1 & text_length <= 40 ~ "[1, 40]",
    text_length >= 41 & text_length <= 80 ~ "[41, 80]",
    text_length >= 81 & text_length <= 120 ~ "[81, 120]",
    text_length >= 121 & text_length <= 160 ~ "[121, 160]",
    text_length >= 161 & text_length <= 200 ~ "[161, 200]",
    text_length >= 201 & text_length <= 240 ~ "[201, 240]",
    text_length >= 241 ~ ">= 241"
  ))

# Count the number of tweets in each length category
length_counts <- olympics_tweets %>%
  count(length_category, sort = TRUE)

# Plot the bar chart
ggplot(length_counts, aes(x = length_category, y = n, fill = length_category)) +
  geom_bar(stat = "identity") +
  scale_x_discrete(limits = c("[1, 40]", "[41, 80]", "[81, 120]", "[121, 160]", "[161, 200]", "[201, 24
  labs(title = "Number of Tweets by Text Length",
       x = "Text Length (Characters)",
       y = "Number of Tweets") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
# Identify tweets containing at least one username (@)
olympics_tweets <- olympics_tweets %>%
  mutate(has_username = str_detect(text, "@"))

# we then count the number of tweets with usernames
num_tweets_with_username <- olympics_tweets %>%
  filter(has_username) %>%
  nrow()

# we will count the number of tweets with at least three different usernames
olympics_tweets <- olympics_tweets %>%
  mutate(num_usernames = str_count(text, "@") - str_count(text, "\\s@")) # Count '@' symbols to estimat

# Identify tweets with at least three usernames
num_tweets_with_three_or_more_usernames <- olympics_tweets %>%
  filter(num_usernames >= 3) %>%
  nrow()

# Print results
print(paste("number of tweets containing at least one username:", num_tweets_with_username))
```

```
## [1] "number of tweets containing at least one username: 42827"
```

```r
print(paste("number of tweets containing at least three different usernames:", num_tweets_with_three_or
```

```
## [1] "number of tweets containing at least three different usernames: 15"
```

Outcome analysis

Some observation we have in the first plot: **users who create their tweet accounts earlier tend to have more followers and friends**

- The 2011 accounts have the highest number of followers
- The 2013 and 2016 accounts also enjoy the relatively high number of followers
- The overall number of followers tend to drop gradually over the years

There are quite a few reasons could support why earlier accounts tend to have more followers. Firstly, the early adopters had more time to build their content and create their network in the platform. The gaining followers should be easier due to less competitive content in the platform. Secondly, post 2017 users entered the platform when it is already mature, there are highly competitive content creation making harder for the new users to reach their followers.

Some observation we have in the second plot: ***As users write longer tweets, they tend to have more followers and friends***

-
- [81, 120] & [41, 80]: content within this length also indicate many users write a medium-short content that are clear to the point

The tweets are in [121, 160] range might contain more detailed information or insights which potentially more engaging and high-quality than the shorter ranges. High-quality content tweets also more likely to be retweeted and shared with friends and help users gain more followers. Furthermore, the platform algorithms might also prioritize these content with higher engagement which led to more likes and followers. Thus, more visibility for this range of tweets.

Task 2: Predictive data analysis

Build a classification tree model We will first load the dataset and summary the variables

```r
# Read the CSV file
df <- read.csv("predictive_twitter_data.csv")

summary(df)
```

```
##    text_score      text_score_expansion    hashtag          hasURL
##  Min.   :-16.000   Min.   :-16.000      Min.   :0.0000   Min.   :0.0000
##  1st Qu.:-16.000   1st Qu.:-16.000      1st Qu.:0.0000   1st Qu.:0.0000
##  Median :-16.000   Median :-16.000      Median :0.0000   Median :1.0000
##  Mean   :-14.144   Mean   :-13.982      Mean   :0.1933   Mean   :0.5612
##  3rd Qu.:-10.963   3rd Qu.:-10.500      3rd Qu.:0.0000   3rd Qu.:1.0000
##  Max.   : -5.588   Max.   : -4.501      Max.   :1.0000   Max.   :1.0000
##  NA's   :60
##     isReply            length       tweet_topic_time_diff semantic_overlap
##  Min.   :0.0000   Min.   :  0.00   Min.   : 0.000        Min.   :0.00000
##  1st Qu.:0.0000   1st Qu.: 58.00   1st Qu.: 0.000        1st Qu.:0.00000
##  Median :0.0000   Median : 96.00   Median : 2.000        Median :0.00000
##  Mean   :0.1339   Mean   : 87.99   Mean   : 3.633        Mean   :0.06112
##  3rd Qu.:0.0000   3rd Qu.:116.00   3rd Qu.: 6.000        3rd Qu.:0.00000
##  Max.   :1.0000   Max.   :255.00   Max.   :16.000        Max.   :1.00000
##
##  X.entityTypes      X.entities     organization_entities person_entities
##  Min.   :0.0000   Min.   : 0.000   Min.   :0.0000        Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.: 0.000   1st Qu.:0.0000        1st Qu.:0.0000
##  Median :0.0000   Median : 2.000   Median :0.0000        Median :0.0000
##  Mean   :0.6113   Mean   : 1.916   Mean   :0.1973        Mean   :0.1881
##  3rd Qu.:1.0000   3rd Qu.: 3.000   3rd Qu.:0.0000        3rd Qu.:0.0000
```

```
##  Max.   :4.0000   Max.   :11.000   Max.   :8.0000        Max.   :8.0000
##
##  work_entities     event_entities     species_entities  places_entities
##  Min.   : 0.0000   Min.   :0.000000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.: 0.0000   1st Qu.:0.000000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median : 0.0000   Median :0.000000   Median :0.00000   Median :0.0000
##  Mean   : 0.2413   Mean   :0.004005   Mean   :0.01146   Mean   :0.1207
##  3rd Qu.: 0.0000   3rd Qu.:0.000000   3rd Qu.:0.00000   3rd Qu.:0.0000
##  Max.   :12.0000   Max.   :2.000000   Max.   :5.00000   Max.   :9.0000
##
##    nFollowers         nFriends         nFavorties          nListed
##  Min.   :      0   Min.   :     0   Min.   :     0.0   Min.   :    0.0
##  1st Qu.:    152   1st Qu.:    78   1st Qu.:     0.0   1st Qu.:    2.0
##  Median :    481   Median :   292   Median :     2.0   Median :    9.0
##  Mean   :   4327   Mean   :  1305   Mean   :   185.9   Mean   :  108.7
##  3rd Qu.:   1470   3rd Qu.:   936   3rd Qu.:    24.0   3rd Qu.:   38.0
##  Max.   :4853601   Max.   :561555   Max.   :551473.0   Max.   :97531.0
##                                                        NA's   :44
##    isVerified        isGeoEnabled      twitterAge     X.tweetsPosted
##  Min.   :0.000000   Min.   :0.0000   Min.   :0.000   Min.   :      0
##  1st Qu.:0.000000   1st Qu.:0.0000   1st Qu.:1.559   1st Qu.:   2512
##  Median :0.000000   Median :0.0000   Median :2.227   Median :  10291
##  Mean   :0.005456   Mean   :0.2457   Mean   :2.218   Mean   :  28961
##  3rd Qu.:0.000000   3rd Qu.:0.0000   3rd Qu.:2.841   3rd Qu.:  30263
##  Max.   :1.000000   Max.   :1.0000   Max.   :5.624   Max.   :1399152
##                                      NA's   :12      NA's   :11
##  relevanceJudge
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.0705
##  3rd Qu.:0.0000
##  Max.   :1.0000
##
```

We will then create a function which help spliting train and test sets

```
create_train_test <- function(data, size = 0.8, train = TRUE) {
  n_row <- nrow(data)
  total_row <- floor(size * n_row)
  train_indices <- sample(1:n_row, total_row, replace = FALSE)

  if (train) {
    # Return training data
    return(data[train_indices, ])
  } else {
    # Return testing data
    return(data[-train_indices, ])
  }
}
```

```
# Create training and testing data
train_data <- create_train_test(df, size = 0.8, train = TRUE)
test_data <- create_train_test(df, size = 0.8, train = FALSE)
```

Train a decision tree model using rpart library

```
tree_model_1 <- rpart(relevanceJudge ~ ., data = train_data, method = "class")
```

Make predictions on the test set

```
y_pred <- predict(tree_model_1, test_data, type = "class")
```

We will then evaluate the model using accuracy and F1 score

```
# Convert the true labels in the test data to factor
y_test <- as.factor(test_data$relevanceJudge)

# Convert the predicted labels to factor
y_pred <- factor(y_pred, levels = levels(y_test))

# Evaluate the model
conf_matrix <- confusionMatrix(y_pred, y_test)
accuracy <- conf_matrix$overall['Accuracy']
f1_score <- conf_matrix$byClass['F1']


# Print the evaluation metrics
print(paste("Accuracy:", round(accuracy, 4)))
```

```
## [1] "Accuracy: 0.9356"
```

```
print(paste("F1 Score:", round(f1_score, 4)))
```

```
## [1] "F1 Score: 0.9659"
```

```
print("Confusion Matrix:")
```

```
## [1] "Confusion Matrix:"
```

```
print(conf_matrix$table)
```

```
##           Reference
## Prediction    0    1
##          0 7301  426
##          1   89  175
```

Improve the performance of Model 1 There are quite a few methods we could use to improve the model, in this section we will use a below technique

- **Deal with errors** (dealing with missing values): notice in the summary part, our dataset includes some NA values, consider it only shows in a very small part of data, we will remove any rows with NA values

```
df_clean <- na.omit(df)
```

- **Feature selection:** this technique requires picking the most relating features for the tree model based on its importance score. In the process of training the new model, we will include the features within a certain threshold and exclude those below it. This help simplify the model by avoid learning the unnecessary features which make a model prone to over-fitting.

```
#feature importance
importance_scores <- tree_model_1$variable.importance

# convert importance scores to a data frame
importance_df <- data.frame(Feature = names(importance_scores), Importance = importance_scores)
```

```r
importance_df <- importance_df[order(-importance_df$Importance),]

print("feature Importance:")
```

```
## [1] "feature Importance:"
```

```r
print(importance_df)
```

```
##                                       Feature   Importance
## text_score                         text_score 605.25210163
## text_score_expansion   text_score_expansion 461.28475220
## hasURL                                 hasURL 109.91294646
## tweet_topic_time_diff tweet_topic_time_diff  67.11319631
## semantic_overlap           semantic_overlap  43.35195559
## length                                 length  34.51029384
## nFavorties                         nFavorties  22.13221554
## isReply                               isReply  19.45141197
## organization_entities organization_entities  18.10388846
## X.entityTypes                   X.entityTypes   6.17734737
## X.entities                         X.entities   5.67698322
## event_entities                 event_entities   0.40320034
## person_entities               person_entities   0.27496320
## twitterAge                         twitterAge   0.22033314
## work_entities                   work_entities   0.21045863
## nFollowers                         nFollowers   0.14030575
## places_entities               places_entities   0.14030575
## X.tweetsPosted                 X.tweetsPosted   0.08064007
```

```r
# select top features bases on threshold = 0.2
top_features <- rownames(importance_df[importance_df$Importance > 0.2, ])
```

```r
# Create training data with selected features
train_data_2 <- create_train_test(df_clean[, c(top_features, "relevanceJudge")], size = 0.8, train = TRU

# Create testing data with selected features
test_data_2 <- create_train_test(df_clean[, c(top_features, "relevanceJudge")], size = 0.8, train = FALS
```

```r
names(train_data_2)
```

```
##  [1] "text_score"            "text_score_expansion"  "hasURL"
##  [4] "tweet_topic_time_diff" "semantic_overlap"      "length"
##  [7] "nFavorties"            "isReply"               "organization_entities"
## [10] "X.entityTypes"         "X.entities"            "event_entities"
## [13] "person_entities"       "twitterAge"            "work_entities"
## [16] "relevanceJudge"
```

```r
names(test_data_2)
```

```
##  [1] "text_score"            "text_score_expansion"  "hasURL"
##  [4] "tweet_topic_time_diff" "semantic_overlap"      "length"
##  [7] "nFavorties"            "isReply"               "organization_entities"
## [10] "X.entityTypes"         "X.entities"            "event_entities"
## [13] "person_entities"       "twitterAge"            "work_entities"
## [16] "relevanceJudge"
```

- **Feature engineering:** this technique will improve the model by creating new features. These feature could potentially capture the complex relationship of the twitter data

1. We can create a new feature to capture the i**nteraction between text__score and text__score__expansion** to see might be the impact of the text__score depending on how it scales
2. We create **interaction__score__hasURL** as an assumption that any tweets includes URL might bring more meaningful implication compare to the one without URL
3. Since the length of a tweet is considered an important aspect from the previous analysis, we create a **squared__length** variable that might capture the non-linear relationship with the target.

```r
# calculate new variables
# variable: interaction between text_score and text_score_expansion
train_data_2$interaction_text_score_expansion <- train_data_2$text_score * train_data_2$text_score_expa
test_data_2$interaction_text_score_expansion <- test_data_2$text_score * test_data_2$text_score_expansi


# variable: interaction has URL
train_data_2$interaction_score_hasURL <- train_data_2$text_score * train_data_2$hasURL
test_data_2$interaction_score_hasURL <- test_data_2$text_score * test_data_2$hasURL

# variable: squared_length
train_data_2$length_squared <- train_data_2$length^2
test_data_2$length_squared <- test_data_2$length^2
```

Now we will train the second model to see if above application could somehow improve the model

```r
# training new model and make prediction
tree_model_2 <- rpart(relevanceJudge ~ ., data = train_data_2, method = "class")
y_pred_2 <- predict(tree_model_2, test_data_2, type = "class")

# convert the true labels in the test data to factor
y_test_2 <- as.factor(test_data_2$relevanceJudge)
y_pred_2 <- factor(y_pred_2, levels = levels(y_test_2))

# Evaluate new model
conf_matrix_2 <- confusionMatrix(y_pred_2, y_test_2)
accuracy_2 <- conf_matrix_2$overall['Accuracy']
f1_score_2 <- conf_matrix_2$byClass['F1']

#evaluation metrics
print(paste("Accuracy:", round(accuracy_2, 4)))
```

```
## [1] "Accuracy: 0.9395"
```

```r
print(paste("F1 Score:", round(f1_score_2, 4)))
```

```
## [1] "F1 Score: 0.9684"
```

```r
print("Confusion Matrix:")
```

```
## [1] "Confusion Matrix:"
```

```r
print(conf_matrix_2$table)
```

```
##           Reference
## Prediction    0    1
##          0 7385  432
##          1   50  100
```

Summary: compare the result from tree__model__1 and tree__model__2, we can tell the approaches we made did improve our model performance. Additionally, I'd like to propose further improvement/other technique could be made outside of this notebook:

- **Dealing with outliers**: the technique to remove outliers from dataset could help the algorithm focusing on the major participant of the data, avoid over-fitting scenerios
- **Try another algorithms:** Random Forest is a more complex algorithm that can capture the hidden patterns in the dataset comparing to a single tree and often it leads to better result
- Using **K-fold CV**: this technique will split dataset in K subsets and essentially it uses all the data we have to train the model which often create a more robust and reliable result.