Data Analysis of Worldwide COVID-19 Cases

Rafeed

Data Analysis Project using R-Studio

Purpose:

For this data analysis project, the following three datasets will be explored:

1. COVID-19 Dataset: This contains COVID-19 case data for a number of different countries, along with the GDP per capita and population of the respective countries.
2. Lockdown Dataset: This contains the dates at which lockdowns took place for the respective countries, along with categorising the type of lockdown.
3. Vaccination Dataset: This contains COVID-19 vaccination data involving doses and vaccination percentages for the different countries.

Using these datasets, we aim to:

- Gather information about how countries around the world were affected by the COVID-19 pandemic.
- Create visualisations to better illustrate the dataset for analysis.
- Identify and explore relationships between different factors and whether they contributed to a better or worse response to the virus.
- Explore the application and effectiveness of training/testing data for this COVID-19 dataset to create a polynomial regression model vs using K-Fold Cross Validaton to make predictions.

Part 1: Data Wrangling and Integration

Loading Libraries and Datasets

Load the necessary libraries.

```
# load libraries
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
##  dplyr     1.1.2       readr     2.1.4
##  forcats   1.0.0       stringr   1.5.0
##  ggplot2   3.4.2       tibble    3.2.1
##  lubridate 1.9.2       tidyr     1.3.0
##  purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
##  dplyr::filter() masks stats::filter()
```

```
##    dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ggplot2)
library(naniar)
library(dplyr)
library(lubridate)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(caret)
```

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

We'll be working with three datasets: Covid-data, CountryLockdowndates, and WorldwideVaccineData. Let's load these into three variables.

```
# load datasets
covid_data <- read.csv("Covid-data.csv")
lockdown_data <- read.csv("CountryLockdowndates.csv")
vaccine_data <- read.csv("WorldwideVaccineData.csv")
```

Data Wrangling and Tidying

To ensure that useful information can be extracted from data sets, it must first undergo data wrangling and tidying. The data cleaning and tidying demands will differ depending on the dataset, what it is trying to convey, and what we're trying to extract out of it.

Covid Dataset

This dataset consists of information related to COVID-19 cases (cumulative/new cases/deaths) in certain countries during the outbreak over a period of time, as well as some additional details.

The following code will list the attributes that are in this dataset.

```
head(covid_data)
```

```
##    location       date total_cases new_cases total_deaths new_deaths
## 1 Australia 2019-12-31           0         0            0          0
## 2 Australia 2020-01-01           0         0            0          0
## 3 Australia 2020-01-02           0         0            0          0
## 4 Australia 2020-01-03           0         0            0          0
## 5 Australia 2020-01-04           0         0            0          0
```

```
## 6 Australia 2020-01-05           0         0            0          0
##   gdp_per_capita population
## 1       44648.71   25499881
## 2       44648.71   25499881
## 3       44648.71   25499881
## 4       44648.71   25499881
## 5       44648.71   25499881
## 6       44648.71   25499881
```

First, we will check for data consistency in the dataset. We expect some attributes to be unique, such as location, and the corresponding gdp_per_capita and population, and thus can do appropriate checks to see if there is any inconsistencies present in the dataset. Attributes such as date, total_cases, new_cases, total_deaths, and new_deaths, are continuous data and are expected to change throughout this dataset.

```
# Check covid_data consistency, for variables which are expected to remain the same when group
ed.
unique(covid_data$location)
```

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

```
length(unique(covid_data$location))
```

```
## [1] 14
```

```
unique(covid_data$gdp_per_capita)
```

```
## [1] 44648.71 15308.71 38605.67 19082.62 35220.08 34272.36 39753.24 54225.45
```

```
length(unique(covid_data$gdp_per_capita))
```

```
## [1] 8
```

```
unique(covid_data$population)
```

```
## [1]   25499881 1439323774   65273512   83992953   60461828   46754783   67886004
## [8]  331002647
```

```
length(unique(covid_data$population))
```

```
## [1] 8
```

From this, we can see there are a few spelling/character inconsistencies for the location data, such as “Australia” and

“Australia”, and “Italy” and “Itly”. These should be corrected.

In total, there should be 8 ‘unique’ locations in the covid_data dataset. For the GPD per capita and population attributes, there does not appear to be any seemly inconsistent values, and there are 8 unique values for each, which corresponds with the number of unique locations. The assumption is made that the population and GDP per capita data available for this dataset will remain constant and unchanged despite the different dates.

```
# Replace inconsistent location values with correct spelling for covid_data.
covid_data$location[covid_data$location == "Australia  "] <- "Australia"
covid_data$location[covid_data$location == " China"] <- "China"
covid_data$location[covid_data$location == "iran"] <- "Iran"
covid_data$location[covid_data$location == "Itly"] <- "Italy"
covid_data$location[covid_data$location == "UnitedKingdom"] <- "United Kingdom"
covid_data$location[covid_data$location == "United Stats"] <- "United States"
unique(covid_data$location)
```

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

```
length(unique(covid_data$location))
```

```
## [1] 8
```

Now that there are 8 unique locations, and have unique corresponding GDP per capita and population values, these attributes can be considered tidied. We can check to see if these attributes are cleaned by checking to see if there is only 1 distinct gdp_per_capita and population value for each location.

```
# Check datatypes for attributes for covid_data.
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 2549
9881 25499881 25499881 ...
```

```
# Check to see only gdp_per_capita and population value for each location.
distinct_gdp_pop <- covid_data %>%
  group_by(location) %>%
  summarise(num_distinct_gdp = n_distinct(gdp_per_capita),
            num_distinct_population = n_distinct(population))
summary(distinct_gdp_pop)
```

```
##    location         num_distinct_gdp num_distinct_population
```

```
##  Length:8           Min.   :1        Min.   :1
##  Class :character   1st Qu.:1        1st Qu.:1
##  Mode  :character   Median :1        Median :1
##                     Mean   :1        Mean   :1
##                     3rd Qu.:1        3rd Qu.:1
##                     Max.   :1        Max.   :1
```

With this, we can confirm that these attributes are clean.

Missing values can also be present in datasets, and it is imperative that these are identified and appropriate action is taken prior to any analysis to ensure that they do not cause inaccurate information to be extracted.

Let's try to identify if this dataset has any missing values, and if so, which of the attributes contain missing values.

```
# Output which attributes have NA values and the NA count.
colSums(is.na(covid_data[c("location", "date", "total_cases", "new_cases", "total_deaths", "new_deaths", "gdp_per_capita", "population")]))
```

```
##       location           date    total_cases      new_cases   total_deaths
##              0              0              0              0              6
##     new_deaths gdp_per_capita     population
##              7              0              0
```

There are 6 total_deaths and 7 new_deaths missing values, and the rest of the dataset is free of missing values. It is important to acknowledge where the missing values are from in the dataset (i.e. which attributes), as it will influence the decision on how best to approach them and understanding the implications of that choice.

This is interesting to know, as total_deaths and new_deaths are related. Looking at the dataset, it can be noted that for each instance, the new_deaths value is considered first, and then added to the total_deaths value from the previous instance to give the total_deaths value for that instance. However, this is a complex task and in an overall visualisation perspective, there are not many values in this dataset with missing values and therefore is acceptable to exclude them.

Similarly, negative values may also be present in this dataset, and hence should be identified and dealt with. We will treat this the same way as we did with finding the missing values.

```
# Check which attributes have negative values and the negative value count.
colSums(covid_data[c("location", "date", "total_cases", "new_cases", "total_deaths", "new_deaths", "gdp_per_capita", "population")] < 0, na.rm = TRUE)
```

```
##       location           date    total_cases      new_cases   total_deaths
##              0              0              0              6              0
##     new_deaths gdp_per_capita     population
##              2              0              0
```

Negative values are clear outliers and their presence in this dataset do not add anything to the data, but instead only complicate it further when dealing with new_cases and new_deaths. Once again, from an overall visualisation perspective, as there are a small number of instances with negative values, it is satisfactory to disregard them.

Let's do exactly that; remove these missing and negative values.

```
# Filter dataset to exclude instances with missing values and negative values.
covid_data_2 <- covid_data %>%
```

```
  filter(!is.na(total_deaths), !is.na(new_deaths), new_cases >= 0, new_deaths >= 0)

# Missing and negative values only accounted for 18 entries total for the whole dataset.
```

Now we are left with a relatively clean covid_data dataset. Before joining, the last thing that will be done is to bring all the dates into a consistent format, and then updating the date column from a string datatype to a date datatype.

```
# Go through date column and correct all to match format YYYY-MM-DD.
covid_data_2$date <- parse_date_time(covid_data_2$date, orders = c('ymd', 'ydm'))

# Turn into date datatype
covid_data_2$date <- as.Date(covid_data_2$date, format = "%Y %m %d")
```

This dataset is now ready for initial joining with the other datasets.

Lockdown Dataset

This dataset consists of information related to the lockdown of certain countries/regions and the respective date(s) at which they went into lockdown, along with some additional information.

The following code will list the attributes and datatypes that are in this dataset.

```
head(lockdown_data)
```

```
##        Country.Region Province       Date Type
## 1         Afghanistan          24/03/2020 Full
## 2             Albania          08/03/2020 Full
## 3             Algeria          24/03/2020 Full
## 4             Andorra          16/03/2020 Full
## 5              Angola          24/03/2020 Full
## 6 Antigua and Barbuda                     None
##
                                    Reference
## 1                https://www.thestatesman.com/world/afghan-govt-imposes-lockd
own-coronavirus-cases-increase-15-1502870945.html
## 2                                                                 https://en.wikipe
dia.org/wiki/2020_coronavirus_pandemic_in_Albania
## 3 https://www.garda.com/crisis24/news-alerts/325896/algeria-government-implements-lockdown-
and-curfew-in-blida-and-algiers-march-23-update-7
## 4                                                                 https://en.wikipe
dia.org/wiki/2020_coronavirus_pandemic_in_Andorra
## 5                                                                  https://en.wikip
edia.org/wiki/2020_coronavirus_pandemic_in_Angola
## 6
```

The information in this dataset will be utilised by joining this to the main covid_data dataset. With this in mind, lockdown_data should be cleaned and tidied appropriately to ensure that only useful data is joined to covid_data. The following column from this lockdown_data will be joined: Country.Region (as the ID attribute), and Date, both of which will need adjustments to the column names to align with the main covid_data dataset criteria requirements.

Before joining, we need to establish what it is that we want to carry over to the covid_data dataset. If a country has multiple lockdown dates (whether or not it has been split into provinces), it is most logical to consider the first lockdown date as the most impactful in relation to a country's COVID-19 pandemic case numbers, and hence the presence of

multiple values is not critical and will not be joined.

First, let's transform the column names of the attributes which we plan to join to covid_data.

```
# Change column names of the attributes which are to be joined to the covid_data dataset to ma
tch criteria, filter the lockdown_date values which are blank, and select only the relevant co
lumns to be joined. Join will use location as ID attribute therefore, match datatype by conver
ting to factor.
lockdown_data_2 <- lockdown_data %>%
  mutate(location = Country.Region,
         lockdown_date = Date) %>%
  filter(lockdown_date != "") %>%
  select(location, lockdown_date)
str(lockdown_data_2)
```

```
## 'data.frame':    230 obs. of  2 variables:
##  $ location     : chr  "Afghanistan" "Albania" "Algeria" "Andorra" ...
##  $ lockdown_date: chr  "24/03/2020" "08/03/2020" "24/03/2020" "16/03/2020" ...
```

Now let's check that the locations in the covid_data dataset are also in this lockdown_data dataset.

```
# Check that required locations are present in lockdown_data.
any(lockdown_data_2$location == "Australia")
```

```
## [1] TRUE
```

```
any(lockdown_data_2$location == "China")
```

```
## [1] TRUE
```

```
any(lockdown_data_2$location == "France")
```

```
## [1] TRUE
```

```
any(lockdown_data_2$location == "Italy")
```

```
## [1] TRUE
```

```
any(lockdown_data_2$location == "Iran")
```

```
## [1] TRUE
```

```
any(lockdown_data_2$location == "Spain")
```

```
## [1] TRUE
```

```
any(lockdown_data_2$location == "United Kingdom")
```

```
## [1] TRUE
```

```
any(lockdown_data_2$location == "United States")
```

```
## [1] FALSE
```

This tells us that most of the locations from the covid_data are also in lockdown_data, except for “United States”, which indicates either an absence or an inconsistency in naming. Let’s check.

```
# Check locations that start with "U" in lockdown_data to see if there is a naming inconsisten
cy or absence of location "United States".
unique(lockdown_data_2[lockdown_data_2$location %in% grep("^U", lockdown_data_2$location, valu
e = TRUE), "location"])
```

```
## [1] "Uganda"                "Ukraine"               "United Arab Emirates"
## [4] "United Kingdom"        "Uruguay"               "US"
## [7] "Uzbekistan"
```

United States has been written as “US” in this dataset. This will be corrected to match covid_data.

```
# Correct naming inconsistency "US" to "United States".
lockdown_data_2$location[lockdown_data_2$location == "US"] <- "United States"
```

Like with the covid_data dataset, we will update the format of the string date values so that it matches the format of the main covid_data date format (YYYY-MM-DD) for consistency, and then convert from string to date datatype.

```
# Go through date column and correct all to match format YYYY-MM-DD to be consistent with the
main covid_data dataset.
lockdown_data_2$lockdown_date <- parse_date_time(lockdown_data_2$lockdown_date, orders = c('dm
y', 'ydm'))

# Turn into date datatype
lockdown_data_2$lockdown_date <- as.Date(lockdown_data_2$lockdown_date, format = "%Y %m %d")
```

Now that the column names are as expected, the newly named location column needs to have unique entries only, including their earliest lockdown date.

```
# Change location to factor datatype, then ensure each entry is distinct with the entry contai
ning the earliest lockdown date for each location. This is the information that will be carrie
d over to the covid_data dataset.
lockdown_data_3 <- lockdown_data_2 %>%
  arrange(location, lockdown_date) %>%
  group_by(location) %>%
  slice(1)
```

The cleaned lockdown_data dataset is now ready to be joined to the cleaned covid_data.

Vaccine Dataset

This dataset consists of information related to the vaccination roll-out for certain countries.

The following code will list the attributes and datatypes that are in this dataset.

```
head(vaccine_data)
```

```
##       Country Doses.administered.per.100.people Total.doses.administered
## 1 Afghanistan                                17                  6445359
## 2     Albania                               102                  2906126
## 3     Algeria                                35                 15205854
## 4      Angola                                64                 20397115
## 5   Argentina                               237                106474858
## 6     Armenia                                73                  2150112
##   X..of.population.vaccinated X..of.population.fully.vaccinated
## 1                          15                                13
## 2                          46                                44
## 3                          19                                16
## 4                          41                                22
## 5                          92                                84
## 6                          38                                33
```

Like the lockdown_data, the information in this vaccine_data will also be joined to the main covid_data dataset. This should be taken into consideration in the cleaning and tidying process of this data so that only important information is joined. The following columns from this table will be joined: Country (as the ID attribute), Total.doses.administered, and X..of.population.fully.vaccinated, all of which will need adjustments to the column names to align with the criteria requirements.

```
# Change column names of the attributes which are to be joined to the covid_data dataset to ma
tch criteria.  Join will use location as ID attribute therefore, match datatype by converting
to factor.
vaccine_data_2 <- vaccine_data %>%
  mutate(location = Country,
         total_doses_administered = Total.doses.administered,
         pc_of_population_fully_vaccinated = X..of.population.fully.vaccinated) %>%
  select(location, total_doses_administered, pc_of_population_fully_vaccinated)
str(vaccine_data_2)
```

```
## 'data.frame':    187 obs. of  3 variables:
##  $ location                        : chr  "Afghanistan" "Albania" "Algeria" "Angola" ...
##  $ total_doses_administered        : num  6.45e+06 2.91e+06 1.52e+07 2.04e+07 1.06e+08 ...
##  $ pc_of_population_fully_vaccinated: num  13 44 16 22 84 33 78 86 75 48 ...
```

Like the lockdown_data, we will check that the locations in the covid_data dataset are also in this vaccine_data dataset.

```
# Check that required locations are present in lockdown_data.
any(vaccine_data_2$location == "Australia")
```

```
## [1] TRUE
```

```
any(vaccine_data_2$location == "China")
```

```
## [1] FALSE
```

```
any(vaccine_data_2$location == "France")
```

```
## [1] TRUE
```

```
any(vaccine_data_2$location == "Italy")
```

```
## [1] TRUE
```

```
any(vaccine_data_2$location == "Iran")
```

```
## [1] TRUE
```

```
any(vaccine_data_2$location == "Spain")
```

```
## [1] TRUE
```

```
any(vaccine_data_2$location == "United Kingdom")
```

```
## [1] FALSE
```

```
any(vaccine_data_2$location == "United States")
```

```
## [1] TRUE
```

This tells us that most of the locations from the covid_data are also in vaccine_data, except for “China” and “United Kingdom”, which again indicates either an absence or an inconsistency in naming. Let’s check.

```
# Check locations that start with "C" and "U" in vaccine_data to see if there is a naming inco
nsistency or absence of location "China".
unique(vaccine_data_2[vaccine_data_2$location %in% grep("^C", vaccine_data_2$location, value =
 TRUE), "location"])
```

```
##  [1] "Cambodia"                "Cameroon"
##  [3] "Canada"                  "Cape Verde"
##  [5] "Central African Republic" "Chad"
##  [7] "Chile"                   "Colombia"
##  [9] "Comoros"                 "Congo"
## [11] "Costa Rica"              "Croatia"
## [13] "Cuba"                    "CuraÃ§ao"
## [15] "Cyprus"                  "Czech Republic"
```

```
# (Not found under "C". Let's try another method.)
unique(vaccine_data_2[grepl("china", vaccine_data_2$location, ignore.case = TRUE), "location"]
)
```

```
## [1] "Mainland China"
```

```
# ...and for "United Kingdom".
unique(vaccine_data_2[vaccine_data_2$location %in% grep("^U", vaccine_data_2$location, value =
 TRUE), "location"])
```

```
## [1] "U.A.E."         "U.K."           "Uganda"         "Ukraine"
## [5] "United States" "Uruguay"        "Uzbekistan"
```

From this we can gather that China and United Kingdom have been entered inconsistently. They will be corrected to match the covid_data.

```
# Correct naming inconsistency "Mainland China" to "China".
vaccine_data_2$location[vaccine_data_2$location == "Mainland China"] <- "China"

# Correct naming inconsistency "U.K." to "United Kingdom".
vaccine_data_2$location[vaccine_data_2$location == "U.K."] <- "United Kingdom"
```

We'll also check if there are any duplicate rows in this dataset.

```
# Check to see if all instances in this dataset are unique.
nrow(vaccine_data_2)
```

```
## [1] 187
```

```
nrow(distinct(vaccine_data_2))
```

```
## [1] 187
```

The vaccine_data dataset looks quite clean as it is, with all entries being unique. We can also do a simple check to see if these values are within expectation or if there are any glaring outliers.

```
# As this data looks relatively clean as is, and does not include any "date" related values, s
ummary provides us a decent overview of if there are any outlier values.
options(scipen = 999)
summary(vaccine_data_2)
```

```
##    location         total_doses_administered pc_of_population_fully_vaccinated
##  Length:187         Min.    :     17139       Min.    : 0.10
##  Class :character   1st Qu.:   1809944       1st Qu.:29.00
##  Mode  :character   Median :   8179010       Median :55.00
##                     Mean    :  64927776       Mean    :51.94
```

```
##                    3rd Qu.:  28647570       3rd Qu.:75.00
##                    Max.   :3407945000       Max.   :99.00
```

Judging by the summary of this dataset, this dataset is clean and can also be joined to the other two datasets.

Joining the three datasets

The covid_data dataset will be the main dataset, to which the lockdown_data and vaccine_data will be joined.

```
# Join the three datasets together. Covid_data_2 is the main dataset, and lockdown_data_3 and
vaccine_data_2 will be left joined.
joined_data <- left_join(covid_data_2, lockdown_data_3, by = "location") %>%
  left_join(vaccine_data_2, by = "location")

# Convert location attribute to factor.
joined_data$location <- as.factor(joined_data$location)
str(joined_data)
```

```
## 'data.frame':    1557 obs. of  11 variables:
##  $ location                         : Factor w/ 8 levels "Australia","China",..: 1 1 1 1 1
1 1 1 1 1 ...
##  $ date                             : Date, format: "2019-12-31" "2020-01-01" ...
##  $ 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 254
99881 25499881 25499881 25499881 25499881 ...
##  $ lockdown_date                    : Date, format: "2020-03-24" "2020-03-24" ...
##  $ total_doses_administered         : num  57988175 57988175 57988175 57988175 57988175 ...
##  $ pc_of_population_fully_vaccinated: num  86 86 86 86 86 86 86 86 86 86 ...
```

Check to see the expected unique locations are present, as these are the key ID attributes.

```
# Checking that required countries are present as key ID attribute.
unique(joined_data$location)
```

```
## [1] Australia      China          France         Iran           Italy
## [6] Spain          United Kingdom United States
## 8 Levels: Australia China France Iran Italy Spain ... United States
```

As the three individual datasets underwent a lot of pre-processing and cleaning, it does not appear that this joined dataset needs any more cleaning at this stage. There are currently a moderate number of zero values (i.e. total _cases, total_deaths, etc.) corresponding to dates for a number of the countries, however these have deliberately not been removed at this stage so that the dates are consistent and case numbers are comparable for each country at each date. Depending on analysis requirements in the following parts of this assignment, this joined dataset may be altered in later sections.

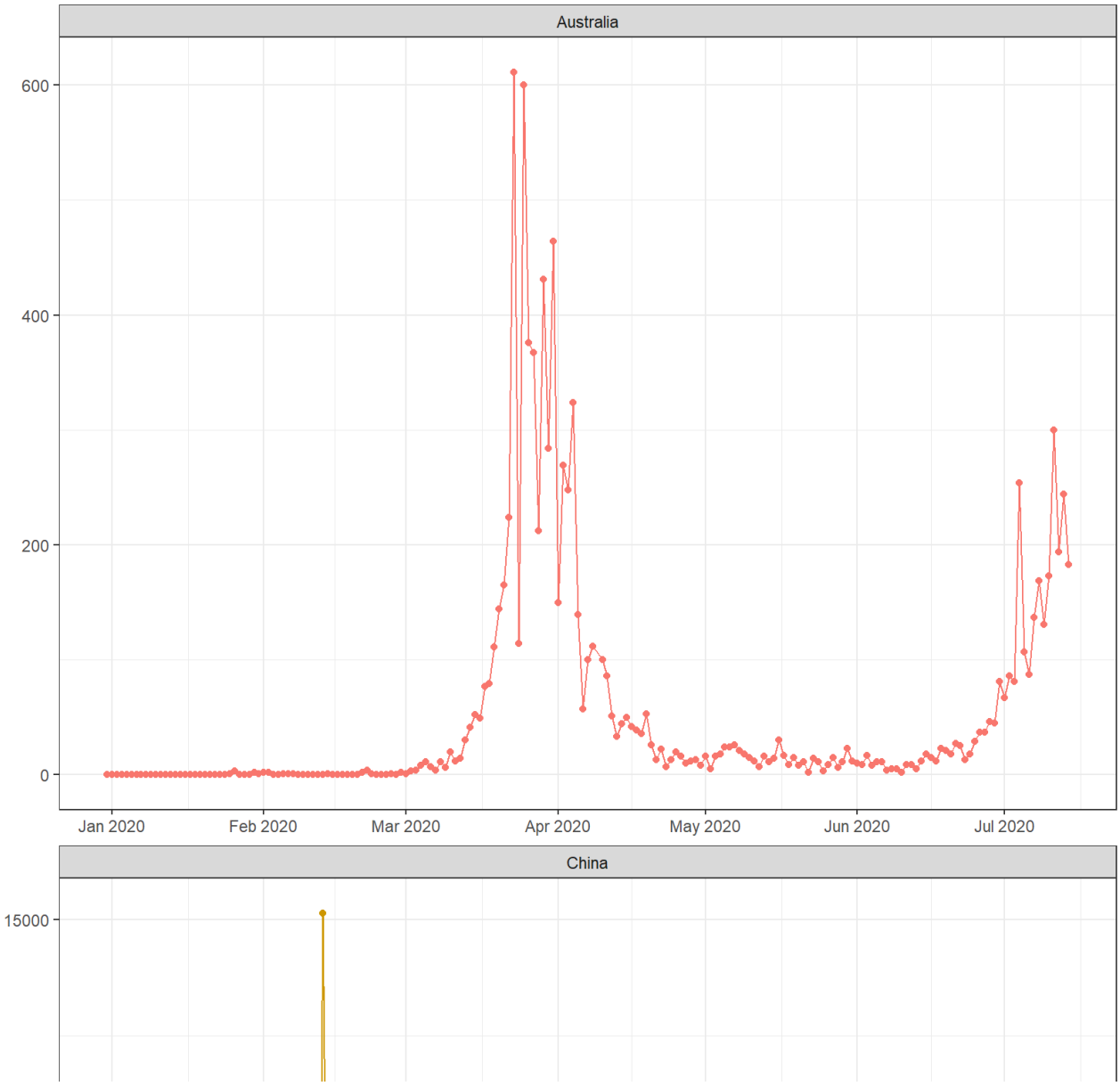
Part 2: Data Visualisation and Analysis

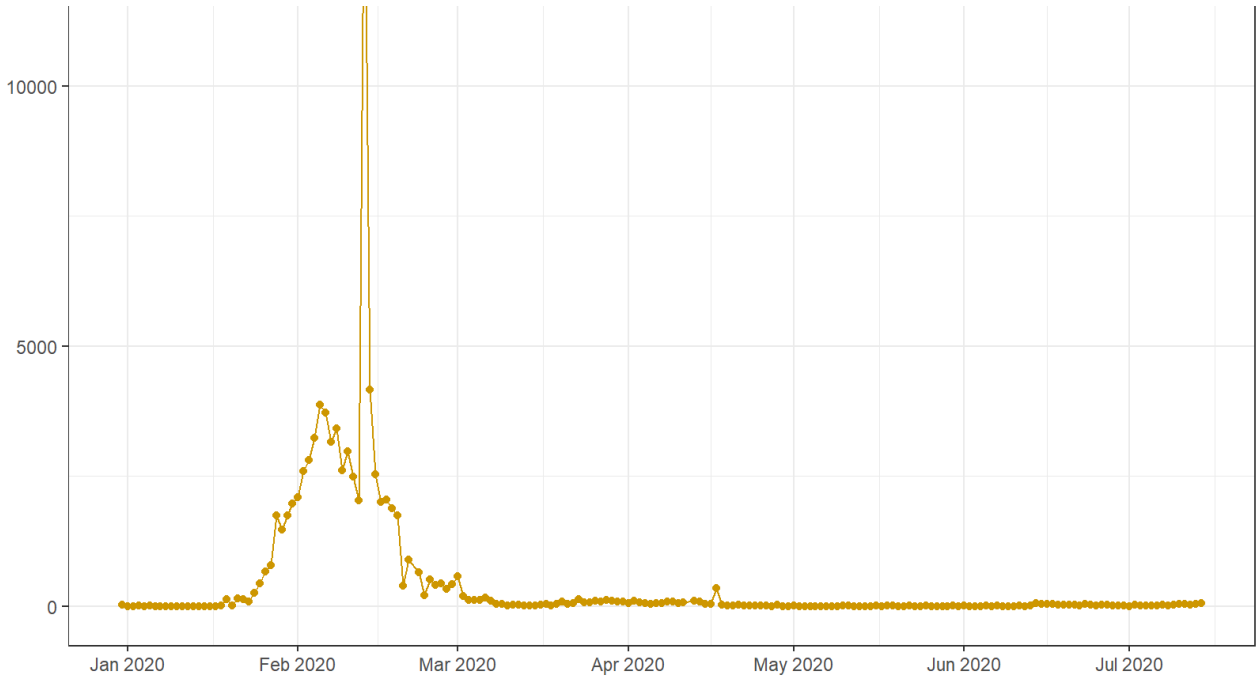
Investigating the trend of new cases for each country.

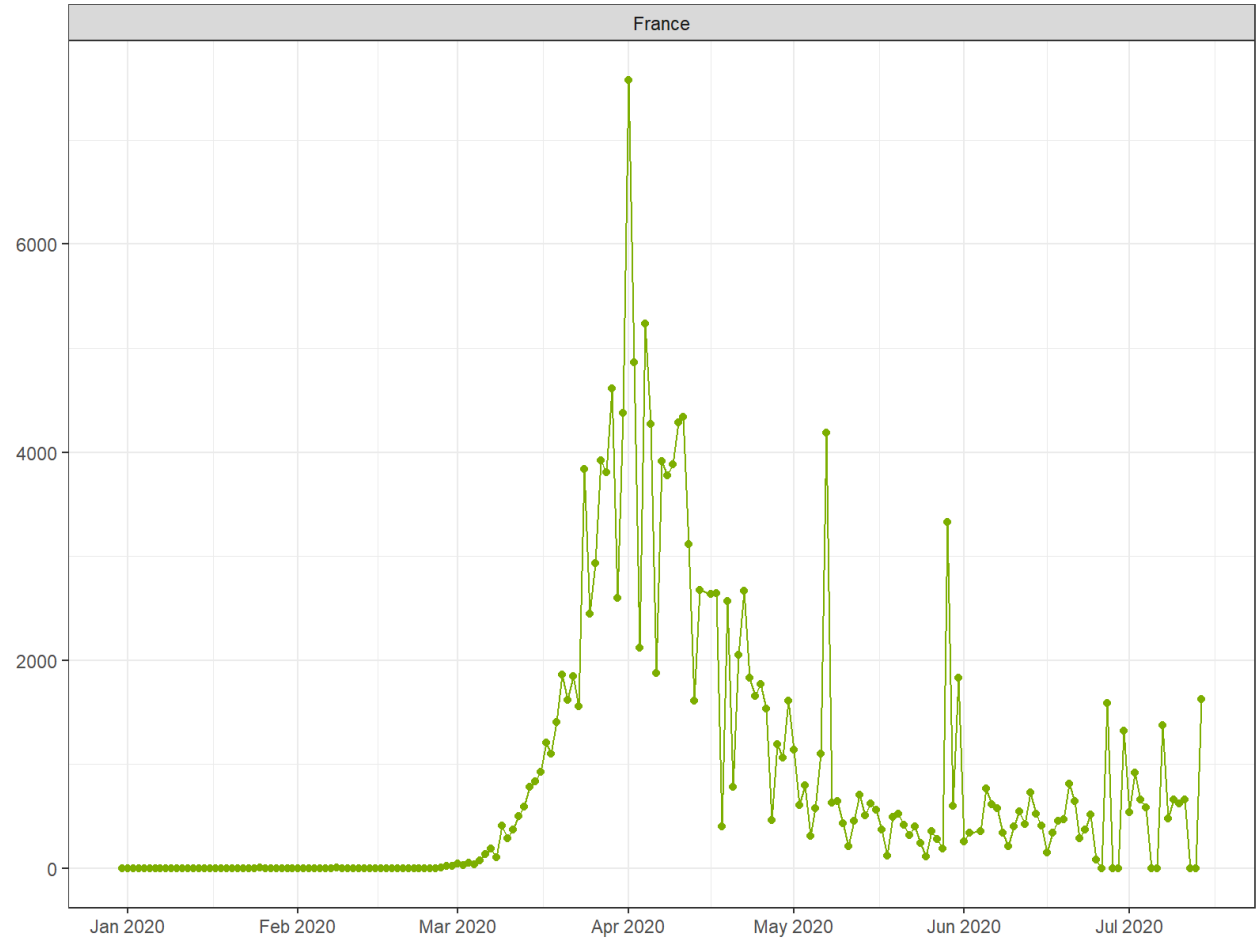
Visualisation: Choosing an appropriate plot to investigate the trend of new cases by country:

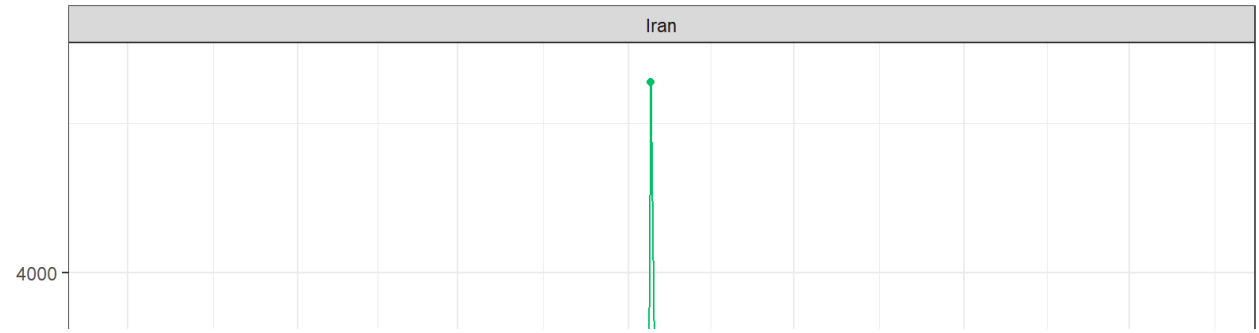
Below is a line graph plot (with points) of the new cases of COVID-19 over a period of time, for each of the countries in the joined dataset. Observable outliers will be assessed and dealt with in a case-by-case basis. Due to the inherent nature of COVID-19 and the way it spreads, sharp peaks will not automatically be considered as outliers, as there have been many cases where this has held true in the real-world COVID-19 data (such as Victoria on 8th of January 2022). Random zero-values however, will be considered and most likely removed.

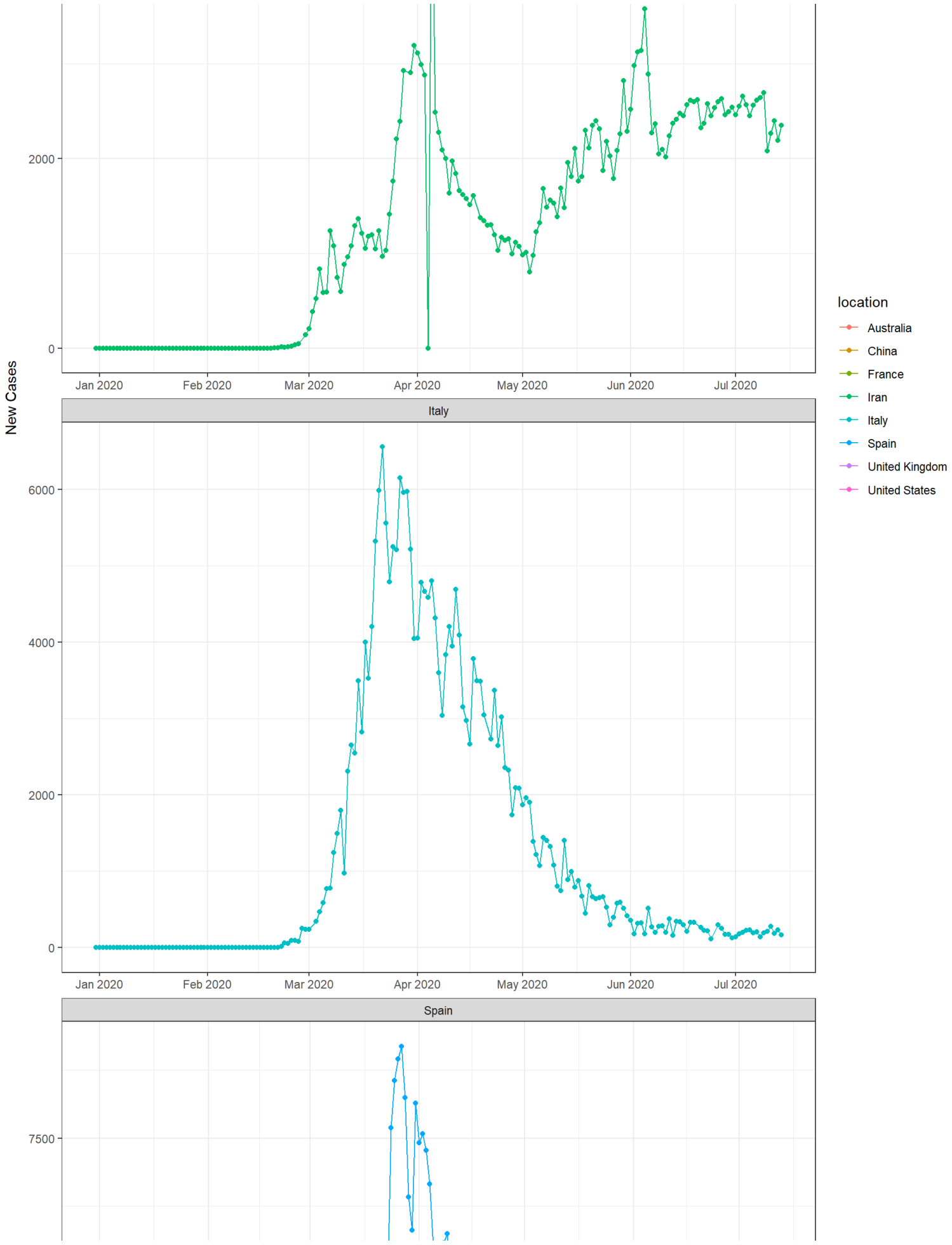
```
# Create line graph with points for new_cases vs date for each location.
ggplot(data = joined_data, aes(x = date, y = new_cases, group = location, color = location)) +
  geom_point() +
  geom_line() +
  facet_wrap(~ location, nrow = 8, ncol = 1, scales = "free") +
  labs(x = "Date", y = "New Cases") +
  theme_bw() +
  scale_x_date(date_labels = "%b %Y", date_breaks = "1 month")
```

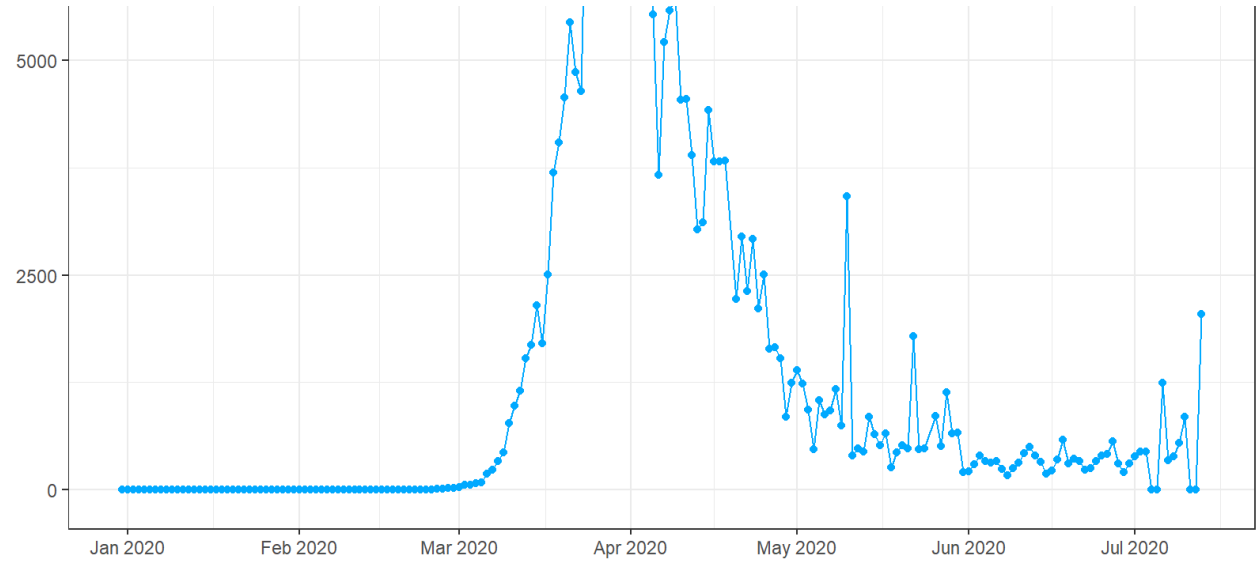


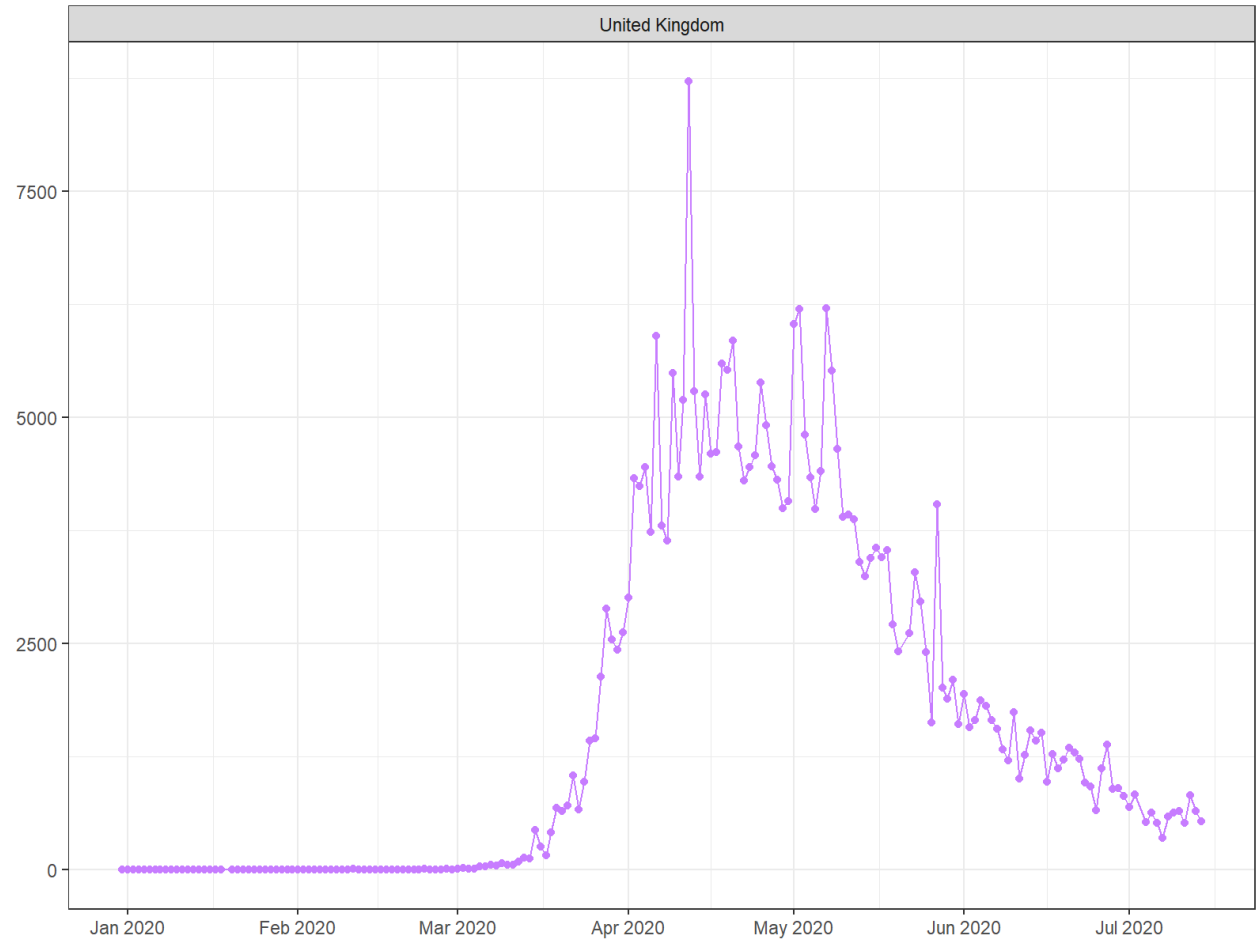

10000
5000
0
Jan 2020
Feb 2020
Mar 2020
Apr 2020
May 2020
Jun 2020
Jul 2020

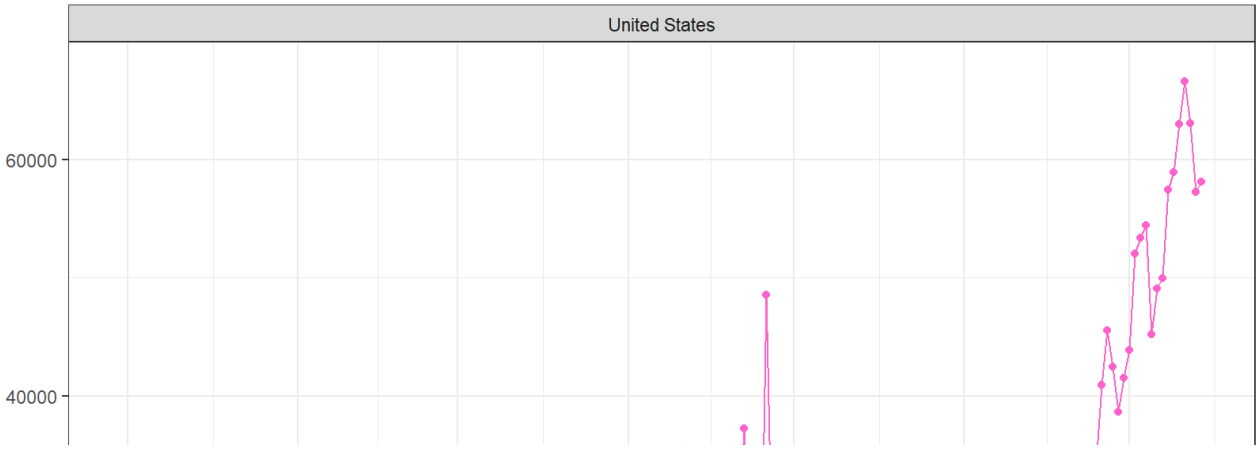

France
6000
4000
2000
0
Jan 2020
Feb 2020
Mar 2020
Apr 2020
May 2020
Jun 2020
Jul 2020

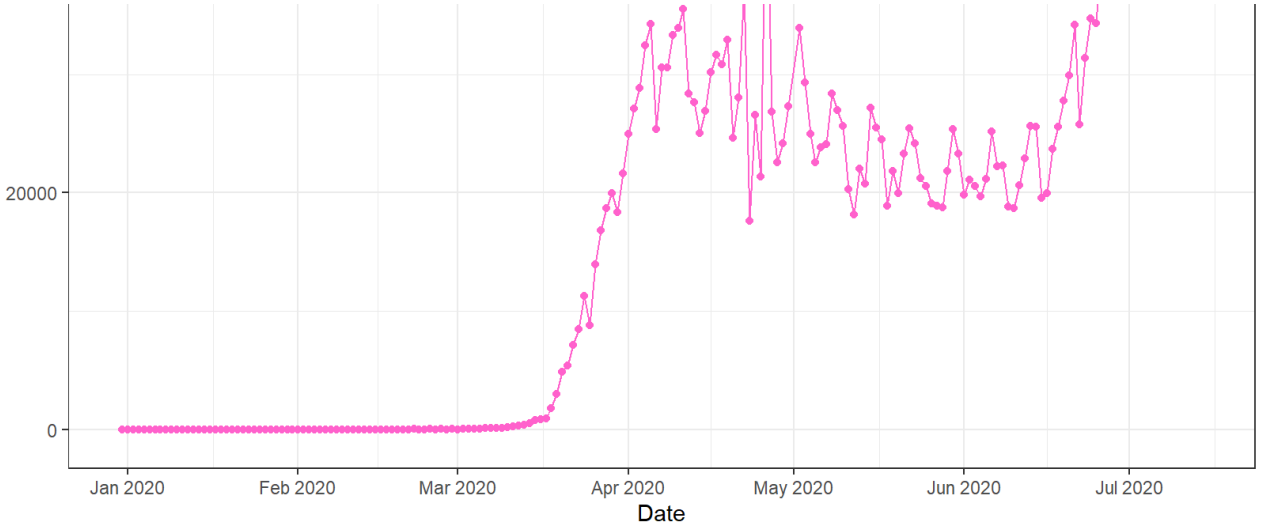

Iran
4000


New Cases
2000
0
Jan 2020
Feb 2020
Mar 2020
Apr 2020
May 2020
Jun 2020
Jul 2020
Italy
6000
4000
2000
0
Spain
7500
location
Australia
China
France
Iran
Italy
Spain
United Kingdom
United States

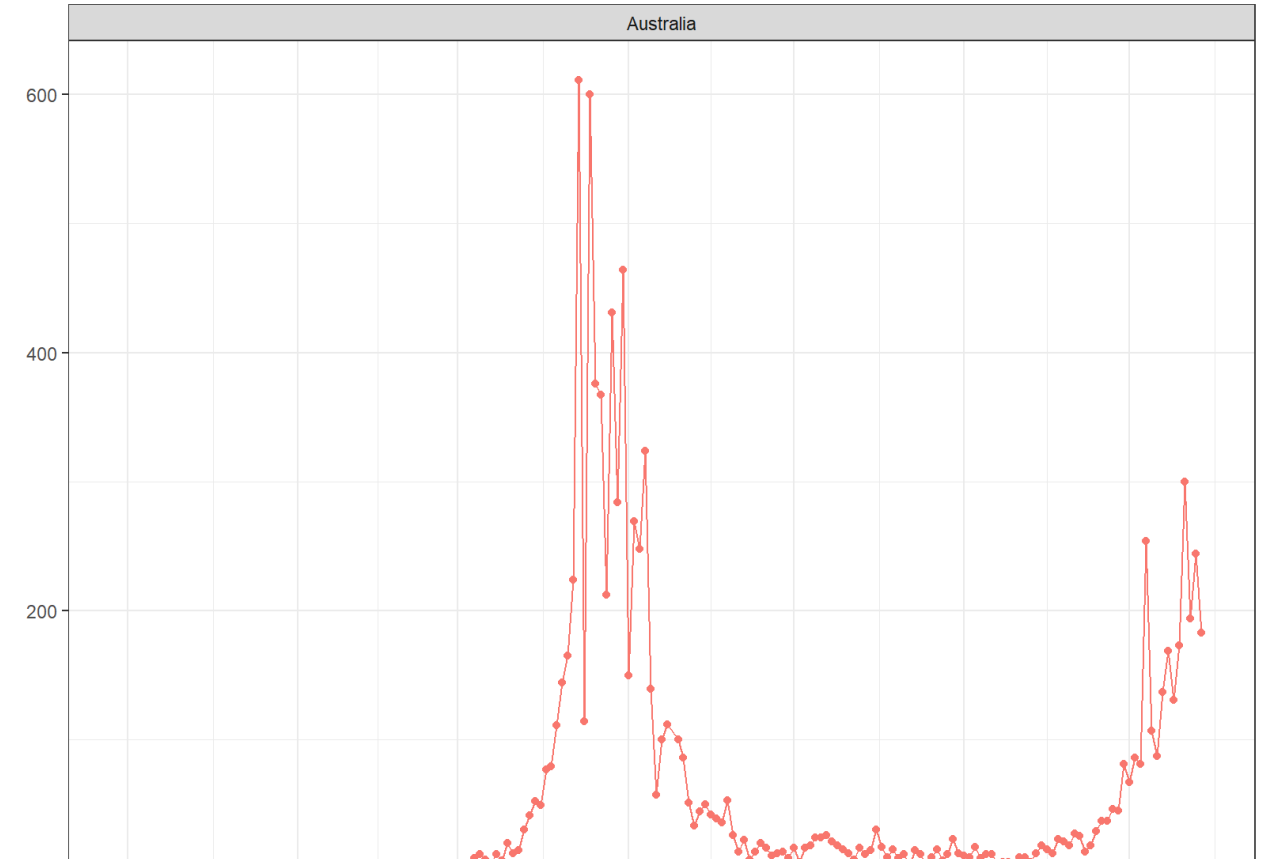

5000
2500
0
Jan 2020
Feb 2020
Mar 2020
Apr 2020
May 2020
Jun 2020
Jul 2020

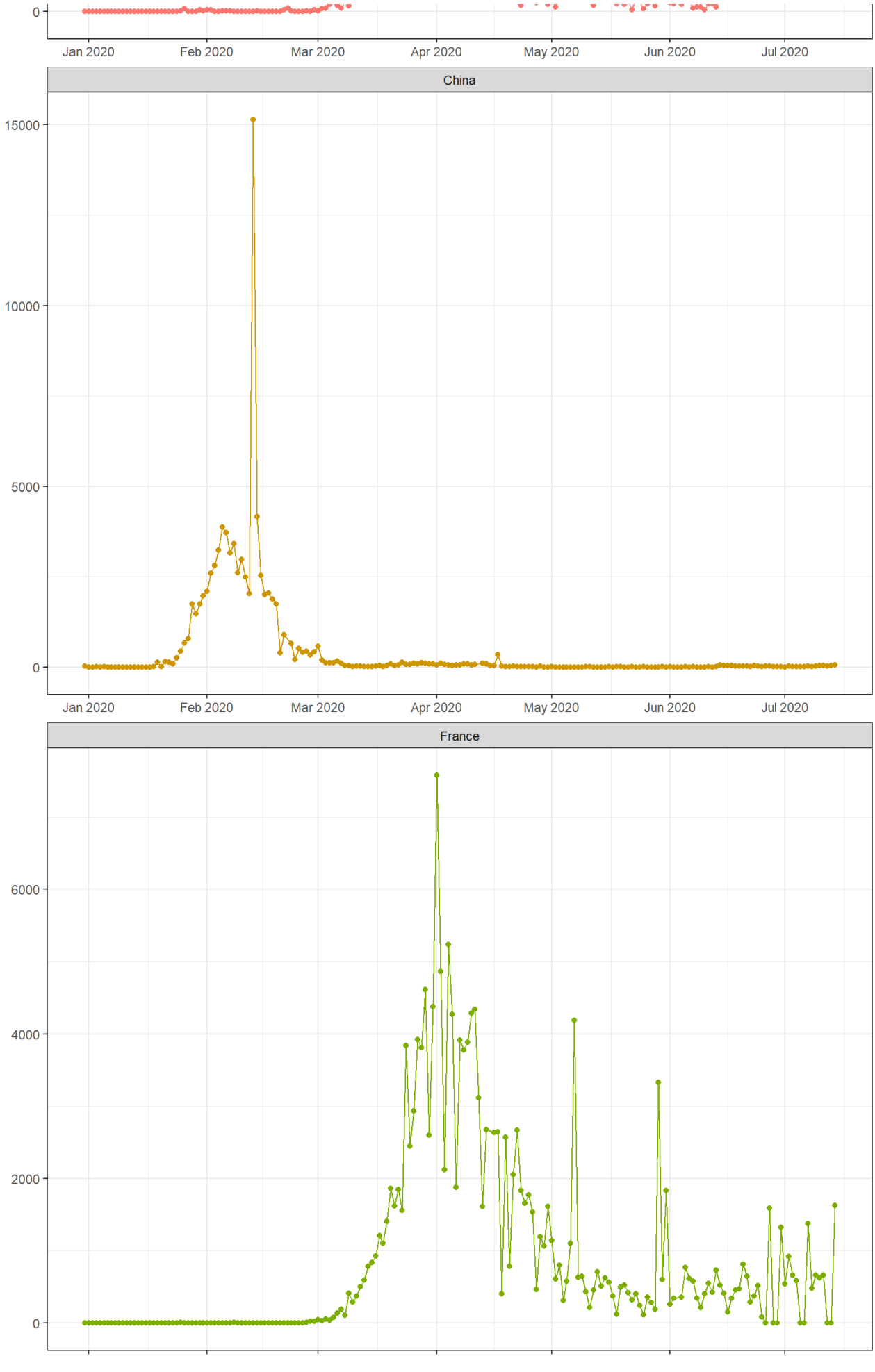

United Kingdom
7500
5000
2500
0
Jan 2020
Feb 2020
Mar 2020
Apr 2020
May 2020
Jun 2020
Jul 2020

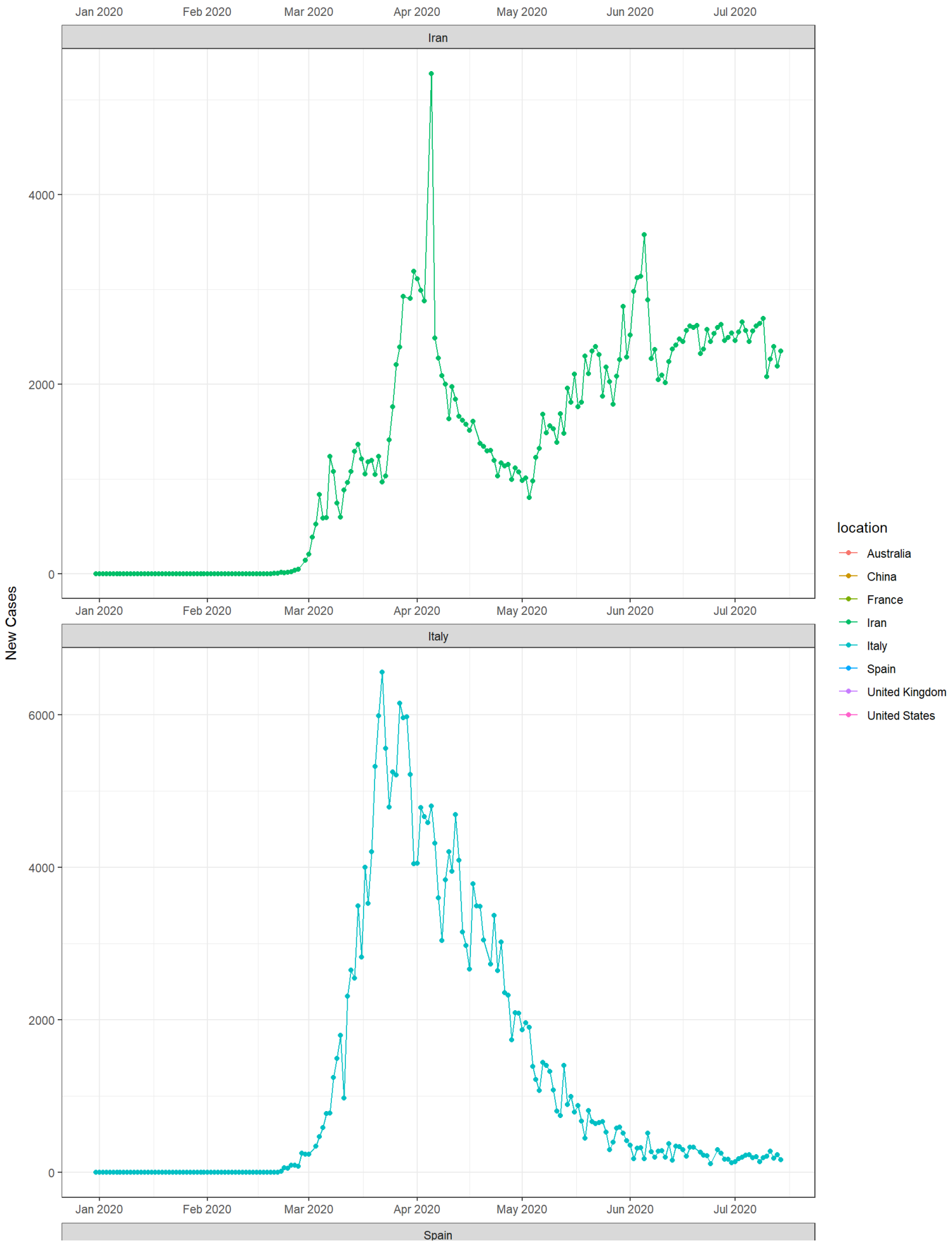

United States
60000
40000

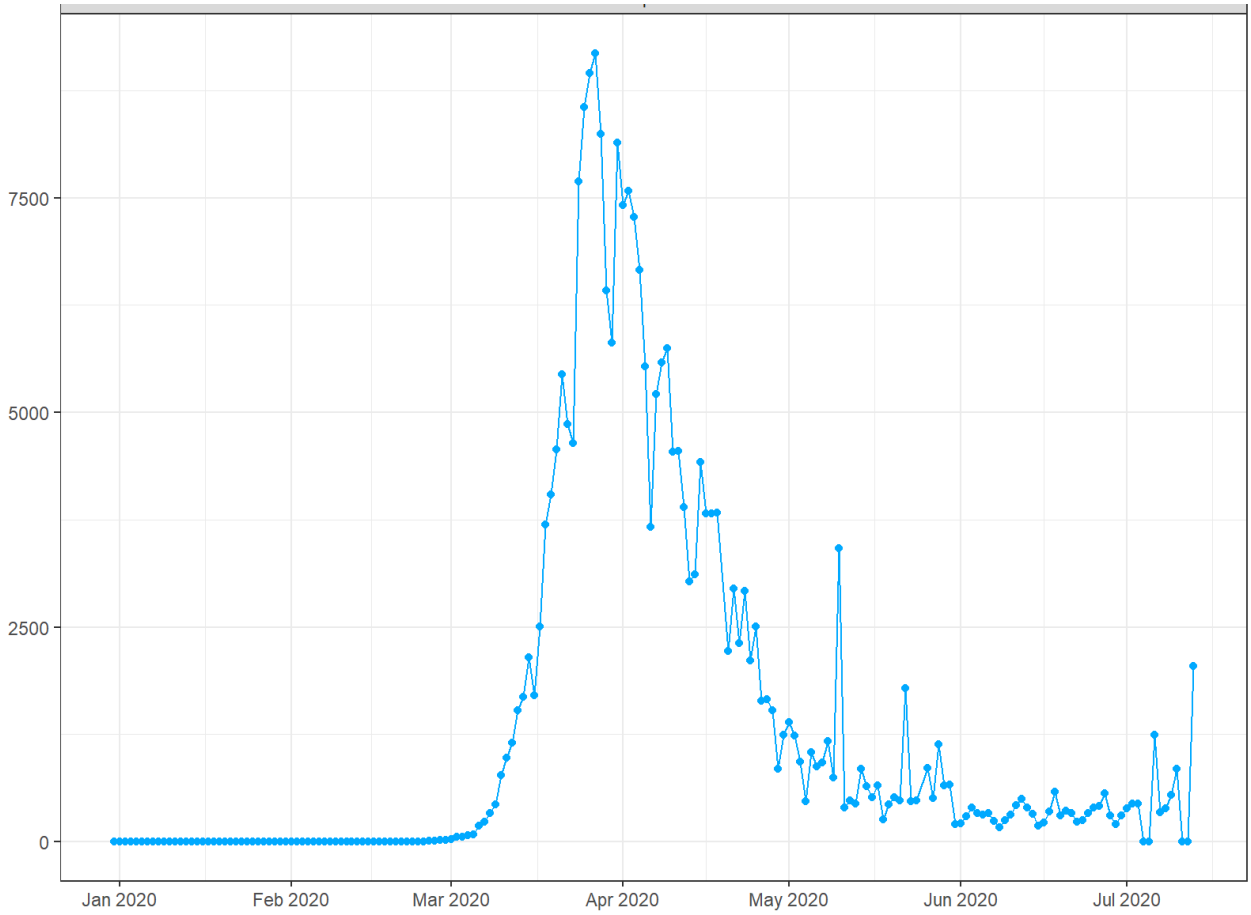


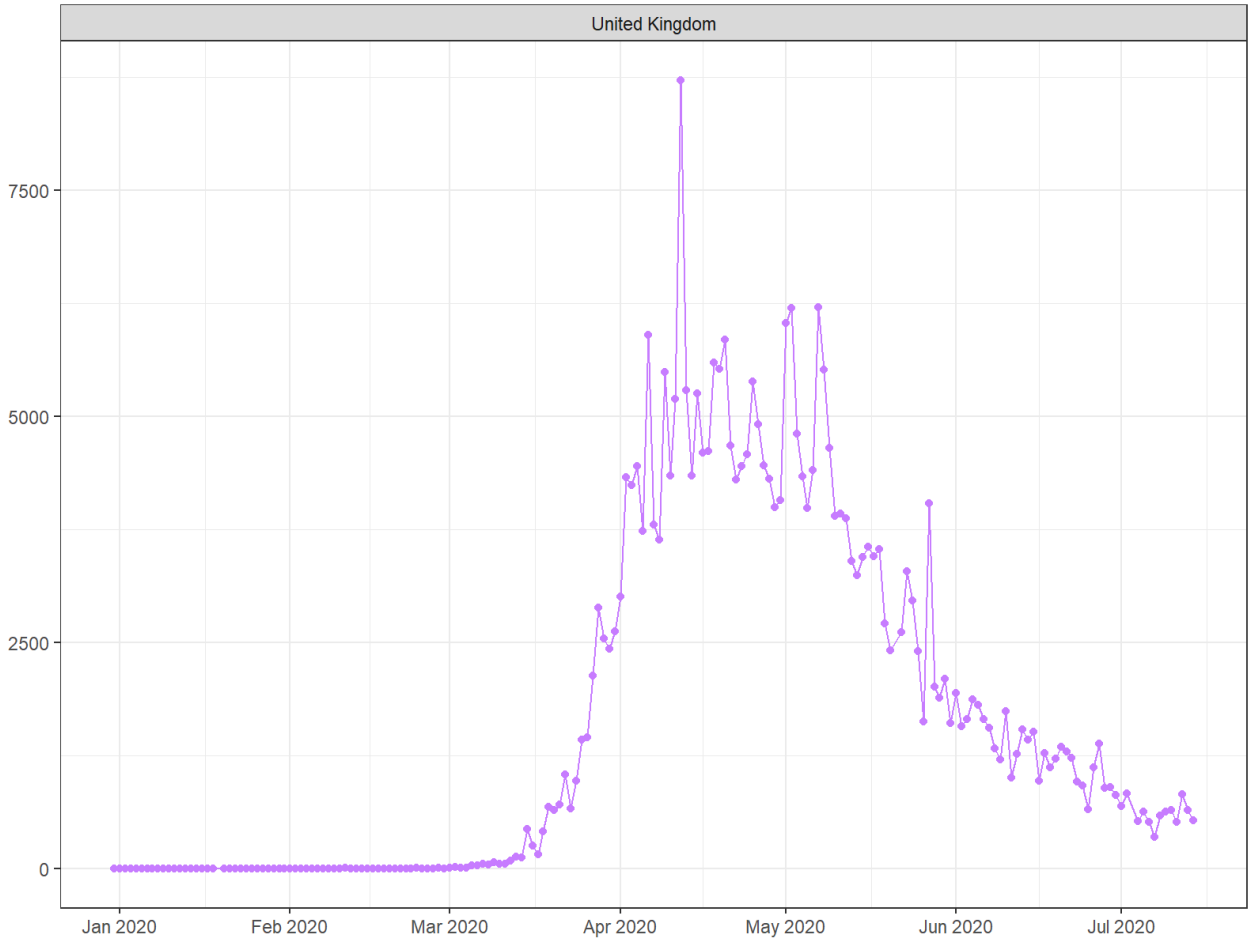

```
# Upon initial visualisation, outlier observed in Iran on 04 April 2020. Remove outlier instan
ce (0 cases and deaths during outbreak peak).
joined_data <- joined_data %>%
  filter(!(location == "Iran" & date == as.Date("2020-04-04")))

# Recreate line graph but with the removed outlier.
ggplot(data = joined_data, aes(x = date, y = new_cases, group = location, color = location)) +
  geom_point() +
  geom_line() +
  facet_wrap(~ location, nrow = 8, ncol = 1, scales = "free") +
  labs(x = "Date", y = "New Cases") +
  theme_bw() +
  scale_x_date(date_labels = "%b %Y", date_breaks = "1 month")
```

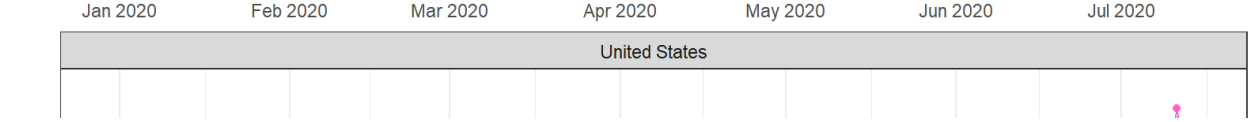


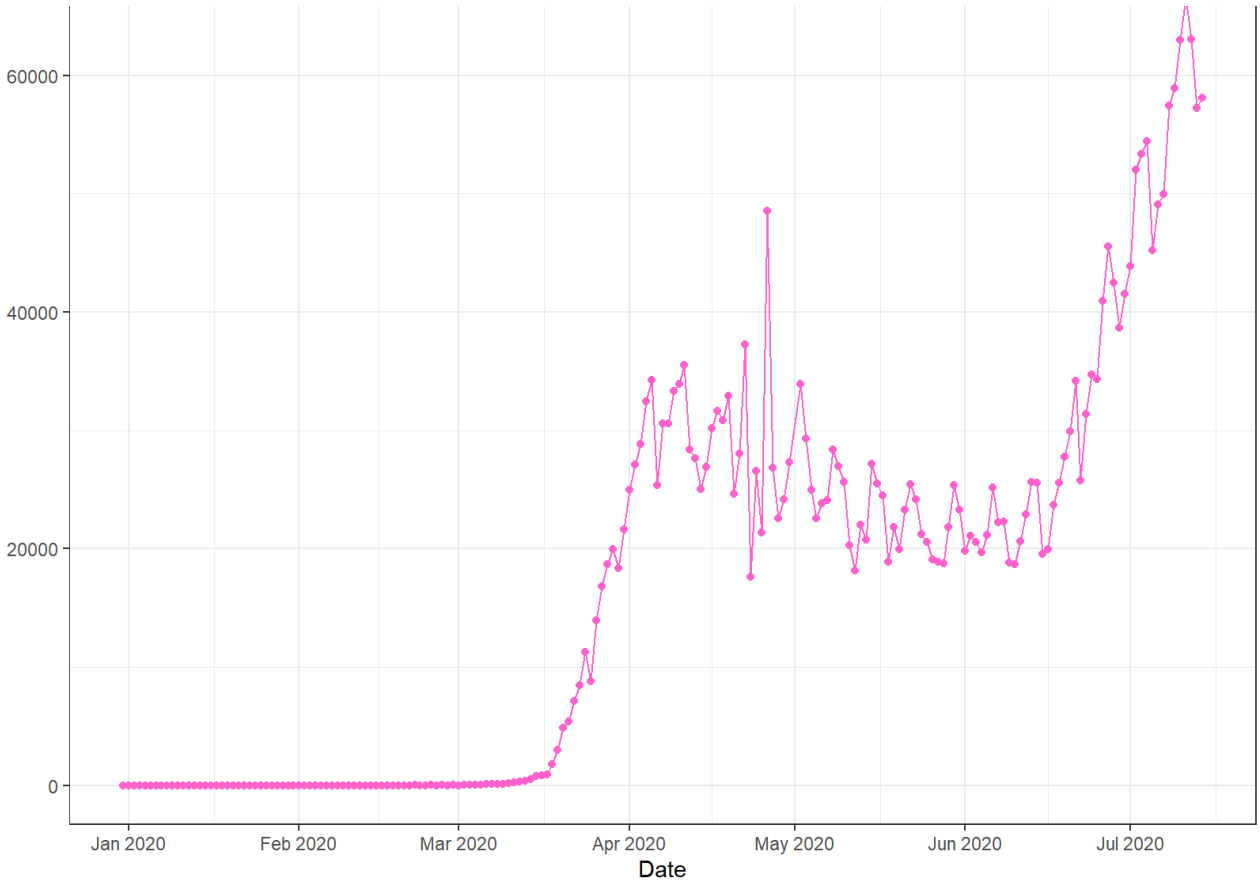

0
Jan 2020
Feb 2020
Mar 2020
Apr 2020
May 2020
Jun 2020
Jul 2020
China
15000
10000
5000
0
Jan 2020
Feb 2020
Mar 2020
Apr 2020
May 2020
Jun 2020
Jul 2020
France
6000
4000
2000
0


Jan 2020
Feb 2020
Mar 2020
Apr 2020
May 2020
Jun 2020
Jul 2020
Iran
Italy
Spain
New Cases
0
2000
4000
6000
location
Australia
China
France
Iran
Italy
Spain
United Kingdom
United States


7500
5000
2500
0
Jan 2020
Feb 2020
Mar 2020
Apr 2020
May 2020
Jun 2020
Jul 2020


United Kingdom
7500
5000
2500
0
Jan 2020
Feb 2020
Mar 2020
Apr 2020
May 2020
Jun 2020
Jul 2020


United States



Exploratory findings: Conduct real-world research into the events surrounding significant points on the plots, such as notable events that may have caused sharp peaks/reductions in cases within a country. Is the trend similar among different countries? What is your understanding of the changes in the trend?

Most of the countries experienced a rapid rate of new daily COVID-19 cases early to mid-March 2020, except for China in mid-January. March 2020 was the period in which COVID-19 had begun significantly spreading worldwide. The country of origin, China, had experienced their peak of new daily cases in mid-February, and with the virus having had an incubation period of around 14 days during its first wave, suggests that many people were infected in China and travelling internationally and therefore spreading the virus to other countries. Of the countries outside of China, Australia was able to suppress and maintain the COVID-19 outbreak the best at early stages, with their implementation of the harshest lockdown measures compared to the rest of the globe, however this was not able to stop the second wave (as observed in July 2020). European nations had similar trends for the extent of COVID-19 exposures, slowly falling over time, which could be attributed to them being neighbouring nations and having similar COVID-19 response strategies. United Kingdom had a similar trend to Europe, but slightly more dragged out. United States however did not see a steady fall in new case number, but rather stayed relatively constant up until the second wave where another rise followed. This could possibly be due to United States having a poor response to the COVID-19 pandemic, with lax lockdown rules, far less enforcement relative to other countries, and early lifting of lockdowns during the first wave, alongside a lackluster healthcare system, making it unaffordable for many to access. Iran was a nation which was also unable to reduce their case number, however this could be due to it being a developing country, resulting in the population having less access to healthcare and basic needs. China's new case data suggests that it was able to swiftly suppress the spreading of COVID-19 after the initial genesis of the virus and maintain extremely low case numbers, however it is uncertain as to how reliable the COVID-19 data provided by China is due to the controversy surrounding their data transparency, and also when taking into consideration that China is the highest populated country in the world. All of these trends observed have been explored on a relative scale, rather than absolute, as each country has different populations.

Observing the relationship between new_deaths or new_cases with

GDP of each country.

Calculating the mean GDP of all the countries.

GDP (Gross Domestic Product) can be defined as a measure of monetary value to the total goods and services, or economic output, produced in a nation over a period of time, i.e. annually. This metric could also be used to draw some correlations on other aspects of a country, such as their access to healthcare and quality of life, and relevantly, their population response to COVID-19. The mean GDP will be calculated and this will then be used to determine whether the GDP of individual countries can be an indicator of COVID-19 cases and deaths, relative to the mean.

To calculate the mean GDP of all the countries, the GDP of each country must first be calculated. This is done with the following equation: GDP per capita * population = GDP. The unique GDP values for each country will then be used to calculate the mean GDP.

```
# Calculate the GDP of each country.
joined_data_2 <- joined_data %>%
  group_by(location) %>%
  mutate(gdp = population*gdp_per_capita)
unique(joined_data_2$gdp)
```

```
## [1]  1138536791804 22034193130919  2519927729287  1602805604777  2129470660954
## [6]  1602396754698  2698688881197 17948766160756
```

```
# Mean GDP of all countries. Save as variable for later use.
mean_gdp <- mean(unique(joined_data_2$gdp))
mean_gdp
```

```
## [1] 6459348214299
```

Adding a column in the joined dataset as ‘GDP_Status’, and determing whether each country has a: higher (and equal to), or lower than average GDP.

The GDP for each country will be compared against the average GDP of all the countries, and a value of “above average” or “below average” will be added to the dataset under a new column named “gdp_status”.

```
# Addition of attribute "gdp_status" to check if gdp of each country is above or below the ave
rage gdp in this dataset.
joined_data_3 <- joined_data_2 %>%
  mutate(gdp_status = case_when(
    gdp >= mean_gdp ~ "above/equal to average",
    gdp < mean_gdp ~ "below average"))
```

Calculating the daily infected case rate and daily death rate (new_case or new_death divided by population in each day for each country).

To calculate the rate of daily infection and daily deaths, the new_cases and new_deaths values must be divided by the population for each date, for each country.

```
# Calculate daily infection and daily death rates for each country.
joined_data_4 <- joined_data_3 %>%
  mutate(daily_infection_rate = new_cases/population,
         daily_death_rate = new_deaths/population)
```

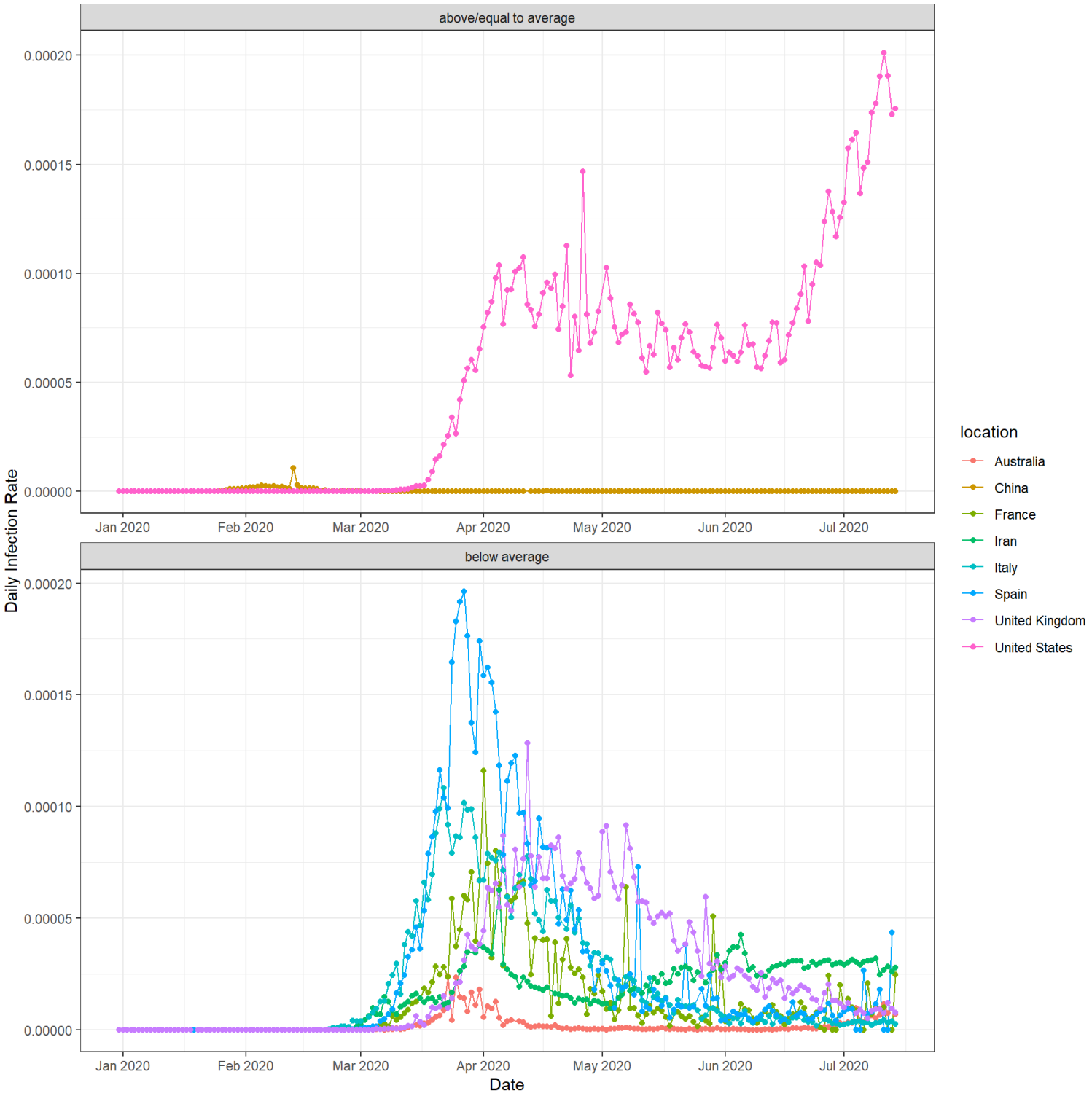
The dataset can be further split by their respective GDP status and then the individual countries can be plotted.

```
# Split the joined dataset to only include countries with above or equal to average GDP.
above_avg_gdp <- joined_data_4 %>%
  filter(gdp_status == "above/equal to average")

# Split the joined dataset to only include countries with below average GDP.
below_avg_gdp <- joined_data_4 %>%
  filter(gdp_status == "below average")
```

Creating a plot to show the relationship between the daily infected case rate within different GDP groups (greater or equal to the average, or less than the average).

Relationship between daily_infection_rate vs date, for countries split by above/equal to average GDP, and lower than average GDP.

```
# Create line graph with points for daily_infection_rate vs date for each country grouped by a
bove/equal to average GDP and lower than average GDP.
ggplot(data = joined_data_4, aes(x = date, y = daily_infection_rate, group = location, color =
 location)) +
  geom_line() +
  geom_point() +
  facet_wrap(~ gdp_status, nrow = 2, ncol = 1, scales = "free") +
  labs(x = "Date", y = "Daily Infection Rate") +
  theme_bw() +
  scale_x_date(date_labels = "%b %Y", date_breaks = "1 month")
```

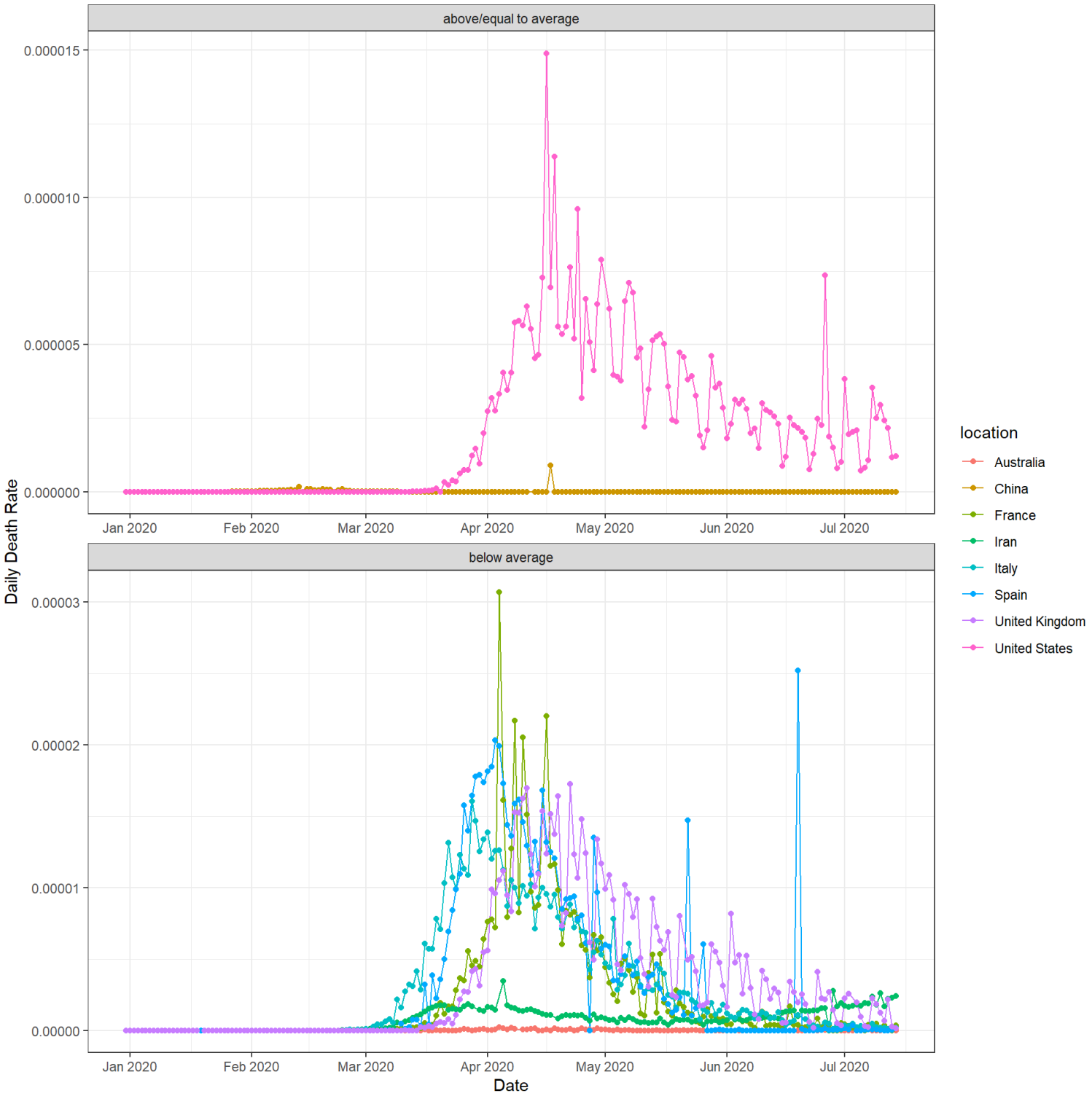


Creating a plot to show the relationship between the daily death rate within different GDP groups (greater or equal to the average, or less than the average).

Relationship between daily_infection_rate vs date, for countries split by above/equal to average GDP, and lower than average GDP.

```
# Create line graph with points for daily_death_rate vs date for each country grouped by above
/equal to average GDP, and below average GDP.
ggplot(data = joined_data_4, aes(x = date, y = daily_death_rate, group = location, color = loc
ation)) +
  geom_line() +
```

```
  geom_point() +
  facet_wrap(~ gdp_status, nrow = 2, ncol = 1, scales = "free") +
  labs(x = "Date", y = "Daily Death Rate") +
  theme_bw() +
  scale_x_date(date_labels = "%b %Y", date_breaks = "1 month")
```



Interpreting and justifying these findings. Are the number of newly infected cases and death cases higher in the High GDP group or the Low GDP group? And why?

When considering the newly infected cases and death cases for the two GDP groups, it should be noted that the average GDP value is heavily influenced by two countries, United States and China, which have been categorised as “above/equal

to average GDP", whereas the other six countries categorised as "below average". This makes interpretation quite challenging, particularly with the large differences in COVID-19 data between United States and China.

The GDP category of a country does not seem to have a significant impact on its daily infection rate. There is a great level of variation in the data for countries within the same GDP category, i.e. United States vs China, and Spain vs Australia, where the countries have much higher and lower rates respectively, despite being in the same GDP group. When examining the y-axis of the two GDP category plots, the scale is quite comparable. This may suggest that the daily infection rate is influenced more greatly by the intrinsic virality of COVID-19 alongside the lack of information surrounding the virus at the time for people to take appropriate preventative measures, and potentially other distinguishing factors unique to the country, such as extent and duration of lockdown restrictions. Almost all countries demonstrated a peak and then a slow and steady decline, beside United States, which demonstrated a relatively constant rate following the initial peak and then a second wave which exceeded the initial peak, which could be attributed to their poor response to the pandemic, rather than their GDP position.

However, the GDP category of a country had a greater impact on its daily death rate. Despite the variation in the data of countries within the same GDP category observed similarly to the daily death rate, the x-axis scale of the two plots here differ quite significantly. There appears to be a greater death rate for countries with a below average GDP, where the observable peak and following decline is close to double that of United States (and China has barely any deaths at all, though it is unclear how accurate this information is). An explanation for this could be that though COVID-19 is a highly contagious virus with a low mortality rate, the countries with higher GDP are more likely to have greater medical facilities, resources and accessibility to healthcare, allowing those infected to get proper care and treatment, which leads to an increased survival rate. Countries with lower GDP are more likely to experience resource shortages and reaching capacity in medical facilities, leading those in need to go without adequate treatment and hence greater likelihood of death.

Part 3: Analysis and Interpretation of Feature Correlations among Attributes

Exploring the relationship among newly infected cases, newly deaths, total doses administered, and % of population fully vaccinated:

Plotting a correlation matrix for the above attributes, and justifying the findings.

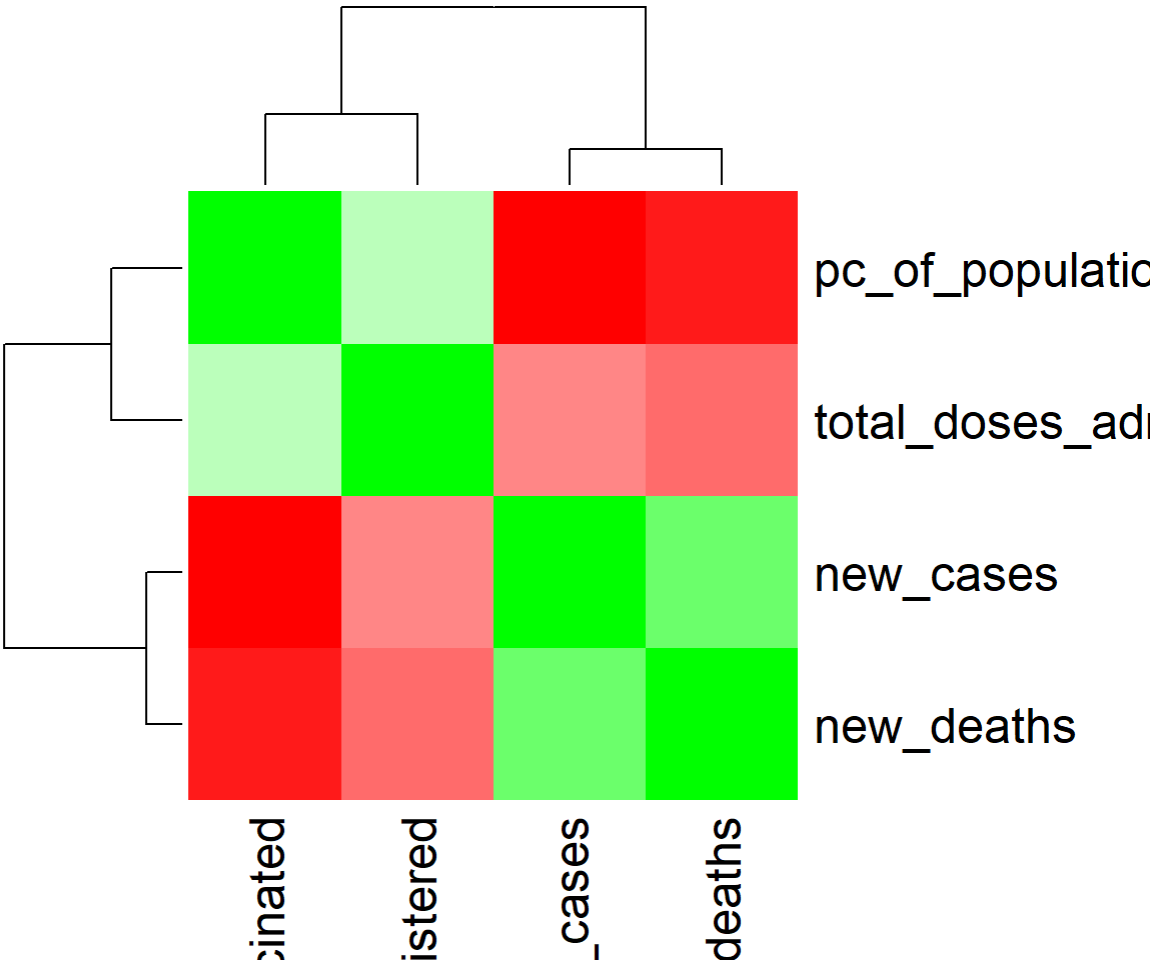
A subset of the joined_data must be created to only include values for the four required attributes, which will be used to create a correlation matrix that can be plotted.

```
# Create dataset with the four required attributes to explore correlation.
corr_data <- joined_data_4 %>%
  ungroup() %>%
  select(new_cases, new_deaths, total_doses_administered, pc_of_population_fully_vaccinated)

# Create correlation matrix.
corr_matrix <- cor(corr_data)
corr_matrix
```

```
##                                    new_cases  new_deaths
## new_cases                         1.00000000  0.68560614
## new_deaths                        0.68560614  1.00000000
## total_doses_administered         -0.02379376 -0.08176947
## pc_of_population_fully_vaccinated -0.42580129 -0.33911797
##                                   total_doses_administered
```

```
## new_cases                                    -0.02379376
## new_deaths                                   -0.08176947
## total_doses_administered                      1.00000000
## pc_of_population_fully_vaccinated             0.44324590
##                                  pc_of_population_fully_vaccinated
## new_cases                                          -0.4258013
## new_deaths                                         -0.3391180
## total_doses_administered                            0.4432459
## pc_of_population_fully_vaccinated                   1.0000000
```

```
# Plot correlation matrix.
palette = colorRampPalette(c("red", "white", "green")) (20)
heatmap(x = corr_matrix, col = palette, symm = TRUE)
```



Which features are strongly correlated to each other? What hypotheses can be drawn from these results? Which features are the least influential with each other? Are these results surprising?

In the above correlation matrix and associated heatmap, the four relevant attributes are plotted to illustrate their positive (green) and negative (red) correlations to one another. with the intensity of the respective colours showcasing the strength of the correlation between two attributes.

There is a relatively strong positive correlation between new_cases and new_deaths. There are moderately negative correlations between pc_of_population_fully_vaccinated and new_cases, and pc_of_population_fully_vaccinated and new_deaths. There is also a moderately positive correlation between pc_of_population_fully_vaccinated and total_doses_administered. There are weak negative correlations between total_doses_administered and new_cases, and total_doses_administered and new_deaths.

There are a few hypotheses that can be drawn from this correlation matrix. The relatively strong positive correlation between new_cases and new_deaths is self-explanatory, as with more new cases of COVID-19 infection, there will be a higher number of COVID-19-related deaths. The moderately negative correlation between pc_of_population_fully_vaccinated and new_cases, and new_deaths, is likely due to the percentage of full vaccination directly relating to the level of immunity for a given country's population, meaning more COVID-19 vaccination will result in greater immunity and therefore less number of new_cases and new_deaths, and vice versa. The moderately positive correlation between pc_of_population_fully_vaccinated and total_doses_administered is also quite self-explanatory, as more vaccine doses are required for a higher percentage of full vaccination, though the population will also play a part in the full vaccination percentage figure (i.e. a country with a higher population will require more doses administered to achieve the same percentage of full vaccination, compared to a country with lower population). The weak negative correlations between total_doses_administered and new_cases, and new_deaths, could be due to the total doses also taking into account a number of single vaccination doses to people, which would provide sub-optimal immunity compared to full vaccination and therefore are less effective in preventing new cases and deaths. However, this is also dependent on how recently a vaccine was rolled out within a country, as there is a waiting period before the first and second dose (i.e. a country that has recently rolled out their vaccinations would mean that the population will only be eligible for the first dose, and therefore have far less immunity compared to a country which has had the vaccinations for a longer period and have begun administering the second dose to the population).

Most of these correlations are not surprising, and are quite logical. Two of the attributes are related to vaccinations, and the other two are related to COVID-19 cases/mortality. As vaccinations are administered with the function of building immunity within the population to help prevent COVID-19 infections and deaths, the observed correlations are reasonably predictable. However, the correlation for total_doses_administered and new_cases, and new_deaths, was surprising to a degree as the correlation coefficient was lower than anticipated and quite close to zero.

Part 4: Fitting a Model and Making Prediction using Polynomial Regression.

Choose a potential country for further analysis:

For this polynomial regression model we will be exploring the dataset for Italy.

Plotting the chosen country and drawing a polynomial/linear line.

A polynomial regression model will be applied for this analysis. First, lets filter the data to only show instances related to Italy. The dataset will also need some tidying to allow for proper regression modelling, such as removing the earlier dates in the dataset with zero or close to zero new case values. and the later dates demonstrating tailing in the values. The earlier dates will be removed as cases were not recorded nearly as consistently in the early stages of COVID-19 compared to the actual outbreak time period where data was more representative, and the tail end of the dataset as the inconsistent fluctuations in the lower end values, otherwise they will negatively impact the model if left in. Also, the date attribute will need to be in a numeric datatype for the model to work.

```
# Filter dataset to only include data for Italy.
italy_data <- joined_data_4 %>%
  filter(location == "Italy")

# We will start from 2020-02-21 to disregard most previous zero-like values, and finish at 202
0-05-31 to disregard the tail end values.
italy_data_model <- italy_data[italy_data$date >= as.Date("2020-02-21"),]
italy_data_model <- italy_data_model[italy_data_model$date <= as.Date("2020-05-31"),]
```

```
# We can remove other columns which are of no benefit.
italy_data_model <- italy_data_model %>%
  select(location, date, new_cases, total_cases, new_deaths, total_deaths)

# Date cannot be date datatype, and needs to be converted to numeric for regression model. Day
 0 will begin on 2020-02-21, and each day from that reference date will be x days as a number.
italy_data_model$date <- as.numeric(difftime(italy_data_model$date, min(italy_data_model$date)
, units = "days"))
```

Making predictions on the newly infected case number for the next five to seven days.

Applying training and testing splits for evaluating predictions. Which training and testing split ratio (7:3, 8:2, 9:1) has been followed?

The model we will be building will apply training and testing splits to evaluate predictions. We will try three different types of splits.

Firstly, the 7:3 split.

```
# Split the data into test into training and testing sets.
set.seed(123)
training.samples <- italy_data_model$new_cases %>%
  createDataPartition(p = 0.7, list = FALSE)
train.data <- italy_data_model[training.samples,]
test.data <- italy_data_model[-training.samples,]

# Build the model.
model <- lm(new_cases ~ poly(date, 5, raw = TRUE),
                            data = train.data)

# Produce predictions.
predictions_test <- model %>% predict(test.data)
predictions_train <- model %>% predict(train.data)


# Performance of model - testing.
modelPerformance_test73 = data.frame(
  RMSE = RMSE(predictions_test, test.data$new_cases),
  R2 = R2(predictions_test, test.data$new_cases)
)

# Performance of model - training.
modelPerformance_train73 = data.frame(
  RMSE = RMSE(predictions_train, train.data$new_cases),
  R2 = R2(predictions_train, train.data$new_cases)
)

# Assess error for model.
model
```

```
##
## Call:
## lm(formula = new_cases ~ poly(date, 5, raw = TRUE), data = train.data)
##
```

```
## Coefficients:
##                 (Intercept)  poly(date, 5, raw = TRUE)1
##                  62.00601616                -165.20190718
## poly(date, 5, raw = TRUE)2  poly(date, 5, raw = TRUE)3
##                  27.95218574                  -0.82500099
## poly(date, 5, raw = TRUE)4  poly(date, 5, raw = TRUE)5
##                   0.00898193                  -0.00003363
```

```
modelPerformance_test73
```

```
##       RMSE        R2
## 1 581.9745 0.9213252
```

```
modelPerformance_train73
```

```
##       RMSE        R2
## 1 552.9204 0.9011289
```

Next the 8:2 split.

```
# Split the data into test into training and testing sets.
set.seed(123)
training.samples <- italy_data_model$new_cases %>%
  createDataPartition(p = 0.8, list = FALSE)
train.data <- italy_data_model[training.samples,]
test.data <- italy_data_model[-training.samples,]

# Build the model.
model <- lm(new_cases ~ poly(date, 5, raw = TRUE),
                            data = train.data)

# Produce predictions.
predictions_test <- model %>% predict(test.data)
predictions_train <- model %>% predict(train.data)


# Performance of model - testing.
modelPerformance_test82 = data.frame(
  RMSE = RMSE(predictions_test, test.data$new_cases),
  R2 = R2(predictions_test, test.data$new_cases)
)

# Performance of model - training.
modelPerformance_train82 = data.frame(
  RMSE = RMSE(predictions_train, train.data$new_cases),
  R2 = R2(predictions_train, train.data$new_cases)
)

# Assess error for the model.
model
```

```
##
## Call:
## lm(formula = new_cases ~ poly(date, 5, raw = TRUE), data = train.data)
##
## Coefficients:
##                 (Intercept)  poly(date, 5, raw = TRUE)1
##                   79.7723827                -186.0236344
## poly(date, 5, raw = TRUE)2  poly(date, 5, raw = TRUE)3
##                   29.9388931                  -0.8774361
## poly(date, 5, raw = TRUE)4  poly(date, 5, raw = TRUE)5
##                    0.0095160                  -0.0000355
```

```
modelPerformance_test82
```

```
##       RMSE        R2
## 1 667.4107 0.8805642
```

```
modelPerformance_train82
```

```
##       RMSE        R2
## 1 531.0021 0.9146836
```

And finally the 9:1 split.

```
# Split the data into test into training and testing sets.
set.seed(123)
training.samples <- italy_data_model$new_cases %>%
  createDataPartition(p = 0.9, list = FALSE)
train.data <- italy_data_model[training.samples,]
test.data <- italy_data_model[-training.samples,]

# Build the model on the training data.
model <- lm(new_cases ~ poly(date, 5, raw = TRUE),
                           data = train.data)

# Produce predictions.
predictions_test <- model %>% predict(test.data)
predictions_train <- model %>% predict(train.data)


# Performance of model - testing.
modelPerformance_test91 = data.frame(
  RMSE = RMSE(predictions_test, test.data$new_cases),
  R2 = R2(predictions_test, test.data$new_cases)
)

# Performance of model - training.
modelPerformance_train91 = data.frame(
  RMSE = RMSE(predictions_train, train.data$new_cases),
  R2 = R2(predictions_train, train.data$new_cases)
```

```
)

# Assess error for model.
model
```

```
##
## Call:
## lm(formula = new_cases ~ poly(date, 5, raw = TRUE), data = train.data)
##
## Coefficients:
##                 (Intercept)  poly(date, 5, raw = TRUE)1
##               170.37724994                -212.23545002
## poly(date, 5, raw = TRUE)2  poly(date, 5, raw = TRUE)3
##                30.98277302                  -0.89288159
## poly(date, 5, raw = TRUE)4  poly(date, 5, raw = TRUE)5
##                 0.00961202                  -0.00003571
```

```
modelPerformance_test91
```

```
##       RMSE        R2
## 1 489.9845 0.9167801
```

```
modelPerformance_train91
```

```
##       RMSE        R2
## 1 560.9182 0.9062136
```

Out of these three test:split options, 9:1 train:test split demonstrates the best RMSE and R^2 pair, with the lowest testing and training RMSEs of 489.98 and 560.92, and R^2 of 0.92 and 0.91 respectively. Additionally, the polynomial to the power of 5 was deemed to be the best degree: 1-3 being far too biased to be considered (and therefore the assessment will not be included below), and 4 and 6 offering slightly higher RMSE and lower R2 than 5 (RSME and R^2 assessments below). It should be noted that the training error is slightly larger than the testing error, which is conventionally strange, but the difference is not of significant concern and is likely just due to the training data consisting of more slightly deviated values than the testing data by chance.

```
# Set seed
set.seed(123)

# Build the model on the training data to the 4th degree.
model4deg <- lm(new_cases ~ poly(date, 4, raw = TRUE),
                            data = train.data)

# Produce predictions.
predictions_test4deg <- model4deg %>% predict(test.data)
predictions_train4deg <- model4deg %>% predict(train.data)


# Performance of model - testing.
modelPerformance_test4deg = data.frame(
  RMSE = RMSE(predictions_test4deg, test.data$new_cases),
```

```
  R2 = R2(predictions_test4deg, test.data$new_cases)
)

# Performance of model - training.
modelPerformance_train4deg = data.frame(
  RMSE = RMSE(predictions_train4deg, train.data$new_cases),
  R2 = R2(predictions_train4deg, train.data$new_cases)
)

# Assess error for model.
model4deg
```

```
##
## Call:
## lm(formula = new_cases ~ poly(date, 4, raw = TRUE), data = train.data)
##
## Coefficients:
##                 (Intercept)  poly(date, 4, raw = TRUE)1
##               -1059.3661676                 201.9981859
## poly(date, 4, raw = TRUE)2  poly(date, 4, raw = TRUE)3
##                   1.4014361                  -0.0994868
## poly(date, 4, raw = TRUE)4
##                   0.0006757
```

```
modelPerformance_test4deg
```

```
##       RMSE        R2
## 1 685.2559 0.8495376
```

```
modelPerformance_train4deg
```

```
##       RMSE        R2
## 1 721.6511 0.8447629
```

```
# Set seed
set.seed(123)

# Build the model on the training data to the 6th degree.
model6deg <- lm(new_cases ~ poly(date, 6, raw = TRUE),
                            data = train.data)

# Produce predictions.
predictions_test6deg <- model6deg %>% predict(test.data)
predictions_train6deg <- model6deg %>% predict(train.data)


# Performance of model - testing.
modelPerformance_test6deg = data.frame(
  RMSE = RMSE(predictions_test6deg, test.data$new_cases),
  R2 = R2(predictions_test6deg, test.data$new_cases)
```

```
)

# Performance of model - training.
modelPerformance_train6deg = data.frame(
  RMSE = RMSE(predictions_train6deg, train.data$new_cases),
  R2 = R2(predictions_train6deg, train.data$new_cases)
)

# Assess error for model.
model6deg
```

```
##
## Call:
## lm(formula = new_cases ~ poly(date, 6, raw = TRUE), data = train.data)
##
## Coefficients:
##                 (Intercept)  poly(date, 6, raw = TRUE)1
##             758.5823076779             -508.4837004323
## poly(date, 6, raw = TRUE)2  poly(date, 6, raw = TRUE)3
##              61.6430675150               -2.1295652834
## poly(date, 6, raw = TRUE)4  poly(date, 6, raw = TRUE)5
##               0.0327965833               -0.0002390184
## poly(date, 6, raw = TRUE)6
##               0.0000006743
```

```
modelPerformance_test6deg
```

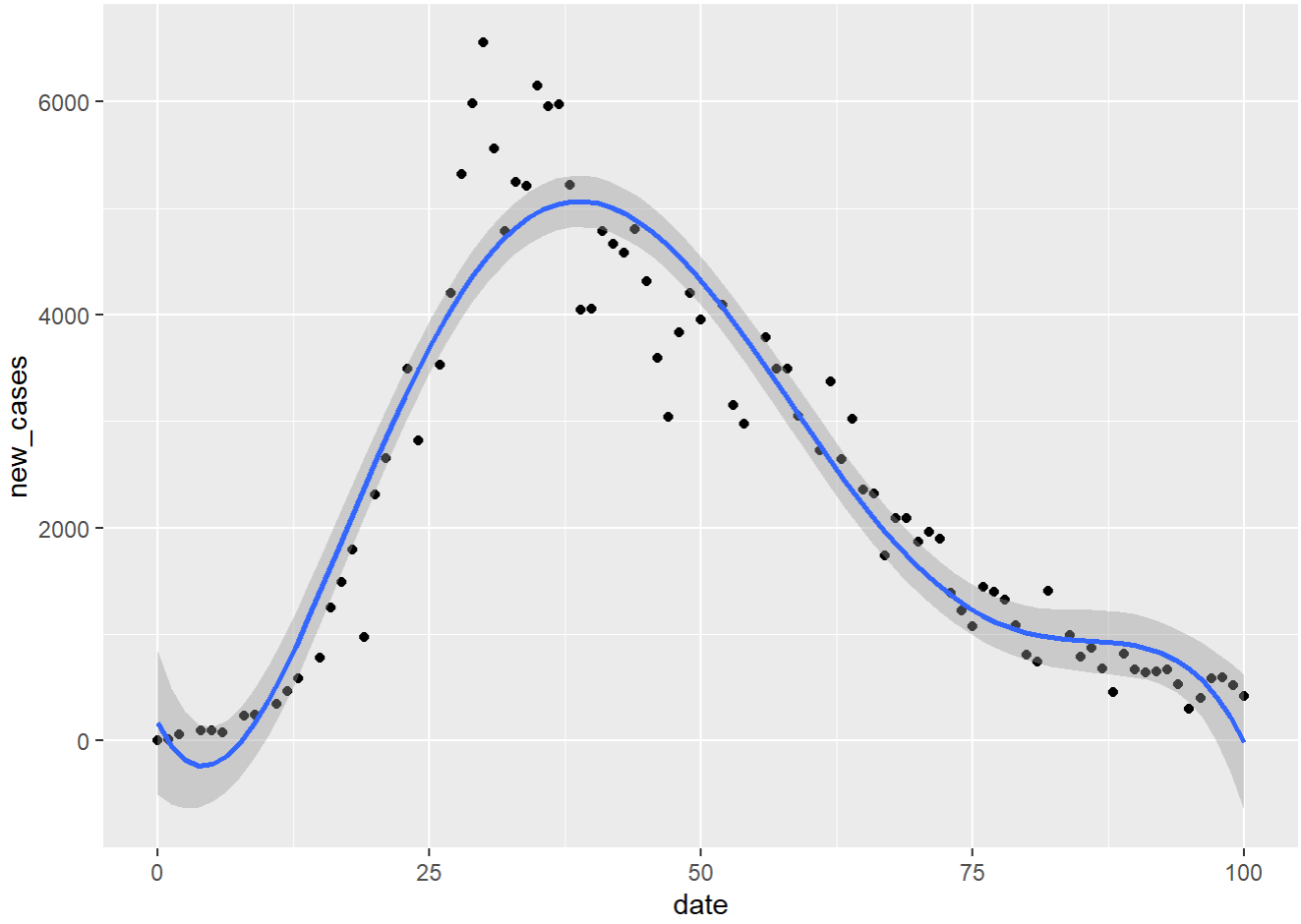
```
##      RMSE        R2
## 1 537.123 0.9049307
```

```
modelPerformance_train6deg
```

```
##       RMSE        R2
## 1 518.3732 0.9199012
```

The chosen model, using 9:1 training/testing split, and as a polynomial to the 5th degree, is plotted below.

```
# Plot the model.
ggplot(train.data, aes(date, new_cases) ) + geom_point() + stat_smooth(method = lm, formula =
y ~ poly(x, 5, raw = TRUE))
```



Using this model, we will predict the next 5 days worth of data.

```
# Predict values for next 5 days using the 9:1 training/testing split model.
new_data <- data.frame(date = max(italy_data_model$date) + 1:5)
predictions_traintest <- predict(model, newdata = new_data)
predictions_traintest
```

```
##          1          2          3          4          5
##  -244.5145  -510.4390  -818.7575 -1173.6937 -1579.6820
```

Application of K-Fold Cross Validation (CV) for evaluating predictions. Is there any difference between the results generated from the simple training and testing split compared to the CV results?

Next, we will explore the K-Fold Cross Validation method.

```
# Specify the cross-validation method
set.seed(123)
ctrl <- trainControl(method = "cv", number = 5)

# Fit a regression model and use K-Fold CV to evaluate performance
model_kfcv <- train(new_cases ~ poly(date, 5, raw = TRUE),
               data = italy_data_model, method = "lm", trControl = ctrl)

# View summary of K-Fold CV
model_kfcv
```

```
## Linear Regression
##
## 99 samples
##  1 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 79, 79, 79, 80, 79
## Resampling results:
##
##   RMSE     Rsquared   MAE
##   566.207  0.9089274  432.1978
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
# View predictions for each fold.
model_kfcv$resample
```

```
##       RMSE  Rsquared      MAE Resample
## 1 471.6426 0.9314664 364.4706    Fold1
## 2 730.5125 0.8740691 530.8762    Fold2
## 3 512.3076 0.9339136 408.1000    Fold3
## 4 669.0444 0.8735178 476.1202    Fold4
## 5 447.5279 0.9316703 381.4220    Fold5
```

Let's also predict the cases for the next 5 days using this K-Fold Cross Validation method.

```
# Predict new_cases for the next 5 days using K-Fold Cross Validation.
predictions_kfcv <- predict(model_kfcv, newdata = new_data)
predictions_kfcv
```

```
##          1          2          3          4          5
##  -265.3142  -538.8984  -856.1174 -1221.2995 -1638.9888
```

If we compare the 9:1 training/testing model with the K-Fold Cross Validation model output metrics, which give us an indication of their performance, it can be observed that the RMSE and R^2 values are slightly better for the simple 9:1 training/testing model, with an RMSE of 489.98 and R^2 of 0.92, versus the RMSE of 566.21 and R^2 of 0.91 in the K-Fold CV method (5 folds was used, but 4-7 folds were trialed outside of the above provided code and yielded slightly worse RMSE and R^2 combinations in comparison). This is most likely due to the dataset containing temporal data, where there is a significance in the order of observations (i.e. the pattern of the new cases as the date increases), which can assist the model in making more accurate predictions for future observations using past data. K-Fold CV however involves data shuffling during the process of cross-validation, and therefore the temporal pattern is not captured as well which makes it less suitable for this dataset.

Observing the results predicted by the model. Conduct some research to see if the results aligned with the actual situation. Justify reasoning from the perspective of overfitting and underfitting phenomena:

Predicted results for new_cases for the next five days using the 9:1 training/testing model.

```
# Predictions for new_cases from training/testing model.
predictions_traintest
```

```
##          1          2          3          4          5
##  -244.5145  -510.4390  -818.7575 -1173.6937 -1579.6820
```

Predicted results for new_cases for the next five days using the K-Fold Cross Validation model.

```
# Predictions for new_cases using K-Fold CV model.
predictions_kfcv
```

```
##          1          2          3          4          5
##  -265.3142  -538.8984  -856.1174 -1221.2995 -1638.9888
```

Actual results for new_cases for the next five day (italy_data).

```
# Actual new_cases for the next five days.
actual_new_cases <- italy_data[italy_data$date >= as.Date("2020-06-01"),]
actual_new_cases <- actual_new_cases[actual_new_cases$date <= as.Date("2020-06-05"),]
actual_new_cases$new_cases
```

```
## [1] 355 178 318 321 177
```

We can see that our models trained with both the testing/training data and with K-Fold CV both do not predict the values very accurately for new_cases in the next five days when compared to actual new_cases, and instead perform poorly. Each of the predicted values for the next five days are negative. As we seen in the plot from earlier, the last direction of the polynomial curves downwards towards below zero, which aligns with the predicted values.

Which category does this model belong to?

This model slightly belongs more towards the underfitting category. It can be seen that the actual values from the Italy dataset align relatively well with the polynomial regression plot. Though there is a slight difference between testing error and training error, with the training error being slightly larger than the testing error, the small difference in RMSE and R^2 between the two suggests underfitting according to the model complexity vs error relationship theory.

Does the CV technique help to reduce the overfitting issue? Justify your findings.

As the CV technique is generally implemented to address overfitting, it has not helped in this case. It is beneficial to apply on models which have greater complexity, where the CV technique evaluates performance on numerous subsets of data to determine if there are any underlying patterns in the different groups. This is evident by the observed increase in RMSE and R^2 values when CV is applied, thereby reducing a model's effectiveness. Additionally as mentioned earlier, the use of CV technique on temporal data also results in losing the element of prediction related to the order of observations, due to the data shuffling aspect of CV.

Are there any assumptions to be made for enhancing the model's performance (e.g., add more data? Apply feature selection?).

The performance of a model which is underfitting can be enhanced in a number of ways. Underfitting models may not have enough data to sufficiently capture an underlying trend, so by adding more data there is a greater chance of learning more complex patterns to improve performance. The complexity of an underfitting model can be increased to include more polynomial degrees so that the plot is more fitted to training data. In the case of the chosen model above, this option is not appropriate in isolation as it has marginally greater errors when raised from the fifth to the sixth degree and would not

improve the model. However, if we included more of the new_cases values from the original Italy dataset (the section where new_cases tails and most values remain low), and increased the complexity of the model in addition, there would be an improvement in the model's performance and new_cases predictability for future dates. Another way to improve performance would be to add more features (variables) to the model, by determining which variables most significantly contribute to its predictive performance. For example, the % of population fully vaccinated figure (if it were to be date-accurate, that is, tracked the vaccination population increase over time), could be implemented as to allow the model to learn that as the vaccinated population increases, the number of new cases should be lower, resulting in more accurate future predictions.