# HarvardX Supervised Machine Learning on Covid-19 Dataset

Roan Indra

5/24/2020

# 1 1. Introduction

Covid-19 is an ongoing pandemic, which has affected more than 5 million people worldwide. This project uses the Covid-19 dataset maintained by Our World in Data. It is free for all purposes, updated daily, and includes data on confirmed cases, deaths, and testing - which will be the focus of this project.

The goal of the project is to build a machine learning model that can predict the daily number of new deaths, which is especially worth studying as it communicates how deadly the pandemic is and how successful our efforts in containing the pandemic.

To that end, we are going to: - Explore and study the Covid-19 data; - Determine the independent variables that could predict the daily number of new deaths; and - Propose and build the machine learning algorithm that has the least RSME

##Reference: Covid-19 dataset - About the data set - The data set in CSV

#2. Method / Analysis

## 1.1 a. Getting the Data

First, we have to ensure that the packages that we need for the project are installed and loaded onto the machine.

```r
# Note: this process could take a couple of minutes
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse

## -- Attaching packages --------------------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  3.0.0     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

Next, we can proceed to download the dataset.

```r
#The dataset can be obtained from the Our World in Data website and it is available in CSV format.
dl <- tempfile()
 download.file("https://covid.ourworldindata.org/data/owid-covid-data.csv", dl)
data <- read.csv(dl)
rm(dl) #we won't be needing dl anymore, so we should remove it to reduce clutter.
```

## 1.2 b. Data Exploration and Visualisation

Let's start by getting some basic understanding of the variables, the structure of the data, and basic statistics.

```r
names(data)
```

```
##  [1] "iso_code"                        "location"
##  [3] "date"                            "total_cases"
##  [5] "new_cases"                       "total_deaths"
##  [7] "new_deaths"                      "total_cases_per_million"
##  [9] "new_cases_per_million"           "total_deaths_per_million"
## [11] "new_deaths_per_million"          "total_tests"
## [13] "new_tests"                       "total_tests_per_thousand"
```

```
## [15] "new_tests_per_thousand"          "new_tests_smoothed"
## [17] "new_tests_smoothed_per_thousand" "tests_units"
## [19] "stringency_index"                "population"
## [21] "population_density"              "median_age"
## [23] "aged_65_older"                   "aged_70_older"
## [25] "gdp_per_capita"                  "extreme_poverty"
## [27] "cvd_death_rate"                  "diabetes_prevalence"
## [29] "female_smokers"                  "male_smokers"
## [31] "handwashing_facilities"          "hospital_beds_per_100k"
```

```
str(data)
```

```
## 'data.frame':    19496 obs. of  32 variables:
##  $ iso_code                       : Factor w/ 212 levels "","ABW","AFG",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ location                       : Factor w/ 212 levels "Afghanistan",..: 10 10 10 10 10 10 10 10 1
##  $ date                           : Factor w/ 146 levels "2019-12-31","2020-01-01",..: 74 81 85 86 8
##  $ total_cases                    : int  2 4 12 17 19 28 28 28 50 55 ...
##  $ new_cases                      : int  2 2 8 5 2 9 0 0 22 5 ...
##  $ 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 ...
##  $ total_cases_per_million        : num  18.7 37.5 112.4 159.2 178 ...
##  $ new_cases_per_million          : num  18.7 18.7 74.9 46.8 18.7 ...
##  $ total_deaths_per_million       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ new_deaths_per_million         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ total_tests                    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_tests                      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ total_tests_per_thousand       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_tests_per_thousand         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_tests_smoothed             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_tests_smoothed_per_thousand: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ tests_units                    : Factor w/ 6 levels "","inconsistent units (COVID Tracking Projec
##  $ stringency_index               : num  0 30.6 44.8 44.8 44.8 ...
##  $ population                     : num  106766 106766 106766 106766 106766 ...
##  $ population_density             : num  585 585 585 585 585 ...
##  $ median_age                     : num  41.2 41.2 41.2 41.2 41.2 41.2 41.2 41.2 41.2 41.2 ...
##  $ aged_65_older                  : num  13.1 13.1 13.1 13.1 13.1 ...
##  $ aged_70_older                  : num  7.45 7.45 7.45 7.45 7.45 ...
##  $ gdp_per_capita                 : num  35974 35974 35974 35974 35974 ...
##  $ extreme_poverty                : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ cvd_death_rate                 : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ diabetes_prevalence            : num  11.6 11.6 11.6 11.6 11.6 ...
##  $ female_smokers                 : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ male_smokers                   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ handwashing_facilities         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ hospital_beds_per_100k         : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
summary(data)
```

```
##     iso_code          location              date         total_cases
##  AUS    :  146   Australia:  146   2020-05-15:  211   Min.   :      0
##  AUT    :  146   Austria  :  146   2020-05-16:  211   1st Qu.:      5
##  BEL    :  146   Belarus  :  146   2020-05-17:  211   Median :     84
##  BLR    :  146   Belgium  :  146   2020-05-18:  211   Mean   :  17526
```

```
##  BRA    :  146   Brazil   :  146   2020-05-19:  211   3rd Qu.:   1135
##  CAN    :  146   Canada   :  146   2020-05-02:  210   Max.   :5273572
##  (Other):18620   (Other)  :18620   (Other)   :18231
##    new_cases       total_deaths      new_deaths       total_cases_per_million
##  Min.   : -2461   Min.   :     0   Min.   :    0.00   Min.   :    0.000
##  1st Qu.:     0   1st Qu.:     0   1st Qu.:    0.00   1st Qu.:    0.593
##  Median :     2   Median :     1   Median :    0.00   Median :   26.720
##  Mean   :   541   Mean   :  1162   Mean   :   35.06   Mean   :  499.009
##  3rd Qu.:    43   3rd Qu.:    24   3rd Qu.:    1.00   3rd Qu.:  252.172
##  Max.   :107909   Max.   :341722   Max.   :10520.00   Max.   :19594.555
##                                                       NA's   :377
##  new_cases_per_million total_deaths_per_million new_deaths_per_million
##  Min.   :-265.189      Min.   :   0.000         Min.   :  0.0000
##  1st Qu.:   0.000      1st Qu.:   0.000         1st Qu.:  0.0000
##  Median :   0.246      Median :   0.167         Median :  0.0000
##  Mean   :  13.313      Mean   :  21.855         Mean   :  0.5852
##  3rd Qu.:   5.857      3rd Qu.:   4.538         3rd Qu.:  0.0540
##  Max.   :4944.376      Max.   :1237.551         Max.   :200.0400
##  NA's   :377           NA's   :377              NA's   :377
##   total_tests         new_tests        total_tests_per_thousand
##  Min.   :       1   Min.   :     1.0   Min.   :  0.000
##  1st Qu.:    8090   1st Qu.:   538.5   1st Qu.:  0.348
##  Median :   43024   Median :  1946.0   Median :  2.454
##  Mean   :  246614   Mean   : 10274.0   Mean   : 11.130
##  3rd Qu.:  153569   3rd Qu.:  6233.2   3rd Qu.: 13.042
##  Max.   :13784786   Max.   :416546.0   Max.   :172.147
##  NA's   :14332      NA's   :14904      NA's   :14332
##  new_tests_per_thousand new_tests_smoothed new_tests_smoothed_per_thousand
##  Min.   :0.000          Min.   :     0     Min.   :0.000
##  1st Qu.:0.028          1st Qu.:   629     1st Qu.:0.030
##  Median :0.149          Median :  2111     Median :0.149
##  Mean   :0.391          Mean   :  9097     Mean   :0.357
##  3rd Qu.:0.543          3rd Qu.:  5848     3rd Qu.:0.498
##  Max.   :7.285          Max.   :389611     Max.   :4.993
##  NA's   :14904          NA's   :13866      NA's   :13866
##                                      tests_units    stringency_index
##                                            :13267   Min.   :  0.00
##  inconsistent units (COVID Tracking Project):   78   1st Qu.: 16.67
##  people tested                             : 1803   Median : 69.84
##  samples tested                            : 1003   Mean   : 55.87
##  tests performed                           : 2462   3rd Qu.: 86.11
##  units unclear                             :  883   Max.   :100.00
##                                                     NA's   :4500
##    population        population_density    median_age    aged_65_older
##  Min.   :8.090e+02   Min.   :    0.137   Min.   :15.10   Min.   : 1.144
##  1st Qu.:2.352e+06   1st Qu.:   42.729   1st Qu.:25.30   1st Qu.: 4.031
##  Median :9.660e+06   Median :   93.105   Median :32.40   Median : 7.846
##  Mean   :1.092e+08   Mean   :  428.546   Mean   :32.44   Mean   : 9.923
##  3rd Qu.:3.691e+07   3rd Qu.:  227.322   3rd Qu.:41.00   3rd Qu.:15.413
##  Max.   :7.795e+09   Max.   :19347.500   Max.   :48.20   Max.   :27.049
##  NA's   :64          NA's   :850         NA's   :1743    NA's   :1980
##  aged_70_older    gdp_per_capita     extreme_poverty cvd_death_rate
##  Min.   : 0.526   Min.   :   661.2   Min.   : 0.10   Min.   : 79.37
##  1st Qu.: 2.380   1st Qu.:  6885.8   1st Qu.: 0.50   1st Qu.:145.18
```

```
##  Median : 5.021   Median : 15847.4   Median : 1.50   Median :233.07
##  Mean   : 6.322   Mean   : 23347.9   Mean   :10.01   Mean   :244.74
##  3rd Qu.: 9.842   3rd Qu.: 35938.4   3rd Qu.:10.00   3rd Qu.:311.11
##  Max.   :18.493   Max.   :116935.6   Max.   :77.60   Max.   :724.42
##  NA's   :1832     NA's   :1982       NA's   :7878    NA's   :1817
##  diabetes_prevalence female_smokers   male_smokers   handwashing_facilities
##  Min.   : 0.990      Min.   : 0.10   Min.   : 7.70   Min.   : 1.188
##  1st Qu.: 5.310      1st Qu.: 1.90   1st Qu.:21.40   1st Qu.:24.640
##  Median : 7.110      Median : 7.10   Median :31.40   Median :59.607
##  Mean   : 8.006      Mean   :11.35   Mean   :32.64   Mean   :55.563
##  3rd Qu.:10.080      3rd Qu.:20.00   3rd Qu.:40.80   3rd Qu.:84.169
##  Max.   :23.360      Max.   :44.00   Max.   :78.10   Max.   :98.999
##  NA's   :1174        NA's   :5052    NA's   :5206    NA's   :11822
##  hospital_beds_per_100k
##  Min.   : 0.100
##  1st Qu.: 1.400
##  Median : 2.600
##  Mean   : 3.238
##  3rd Qu.: 4.280
##  Max.   :13.800
##  NA's   :3160
```

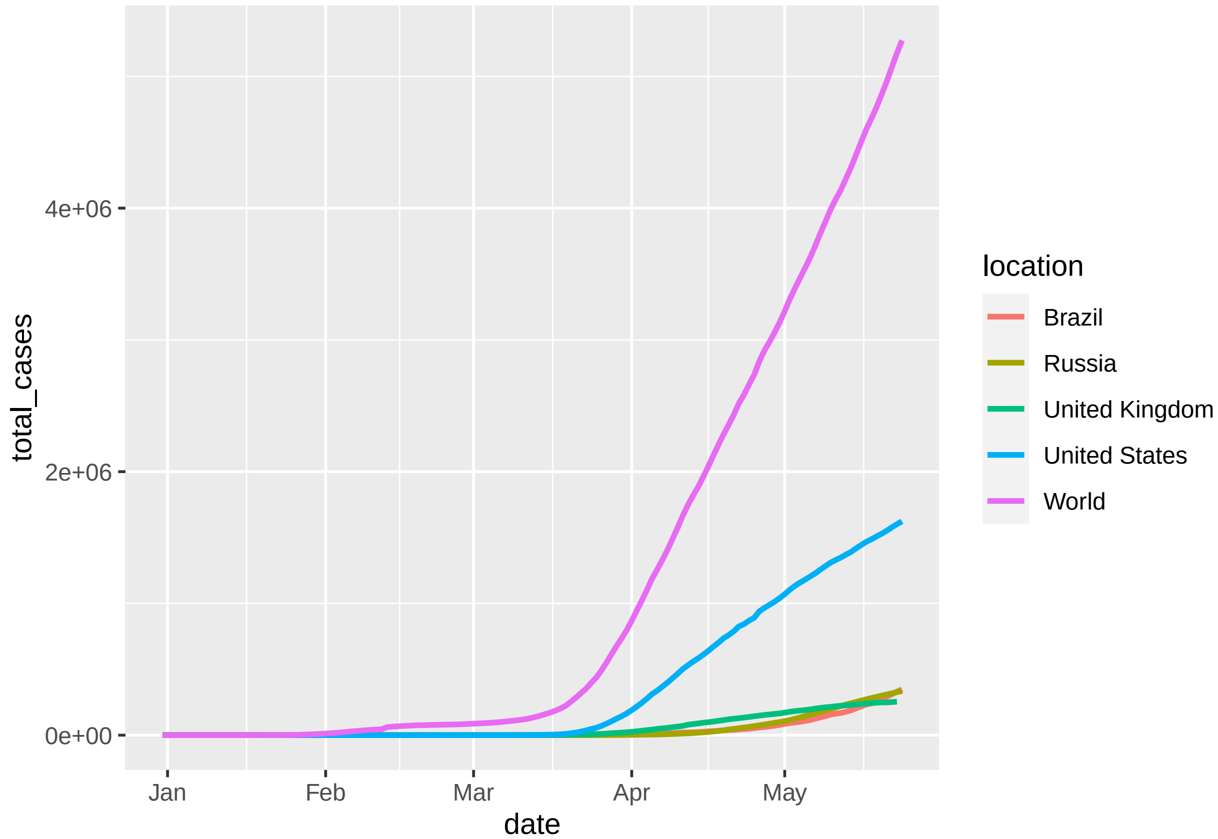
We can see that the date is not formatted correctly. Let's fix that.

```r
data$date <- as.Date(data$date)
class(data$date) #check if the date is correct
```

```
## [1] "Date"
```

We can now do some data visualisations to better understand the relationships among the variables.First, let's visualise the total number of cases as of date in the world as well as in Brazil, Russia, United Kingdom, and United States - four countries with the highest number of cases (as of 25 May 2020).

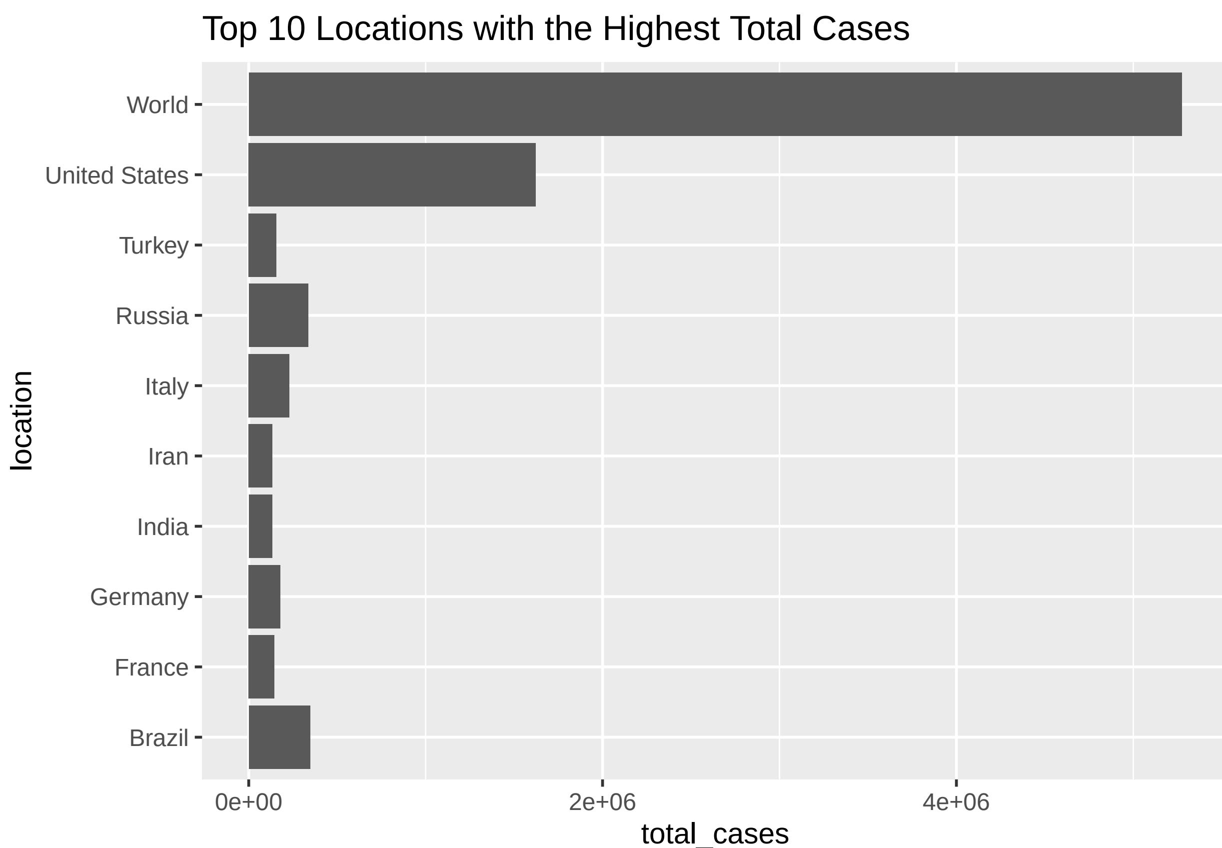
```r
data %>% filter(location %in% c("World", "United States", "Brazil", "Russia", "United Kingdom")) %>% gg
  geom_line(size = 1)
```




Aside from the staggering and rising total number of cases in the world, the chart indicates that United States has, by far, the fastest and highest rising total number of cases in the world (as of 25 May 2020).

Let's use a bar chart to take a closer look on the top 10 locations with the highest total cases.

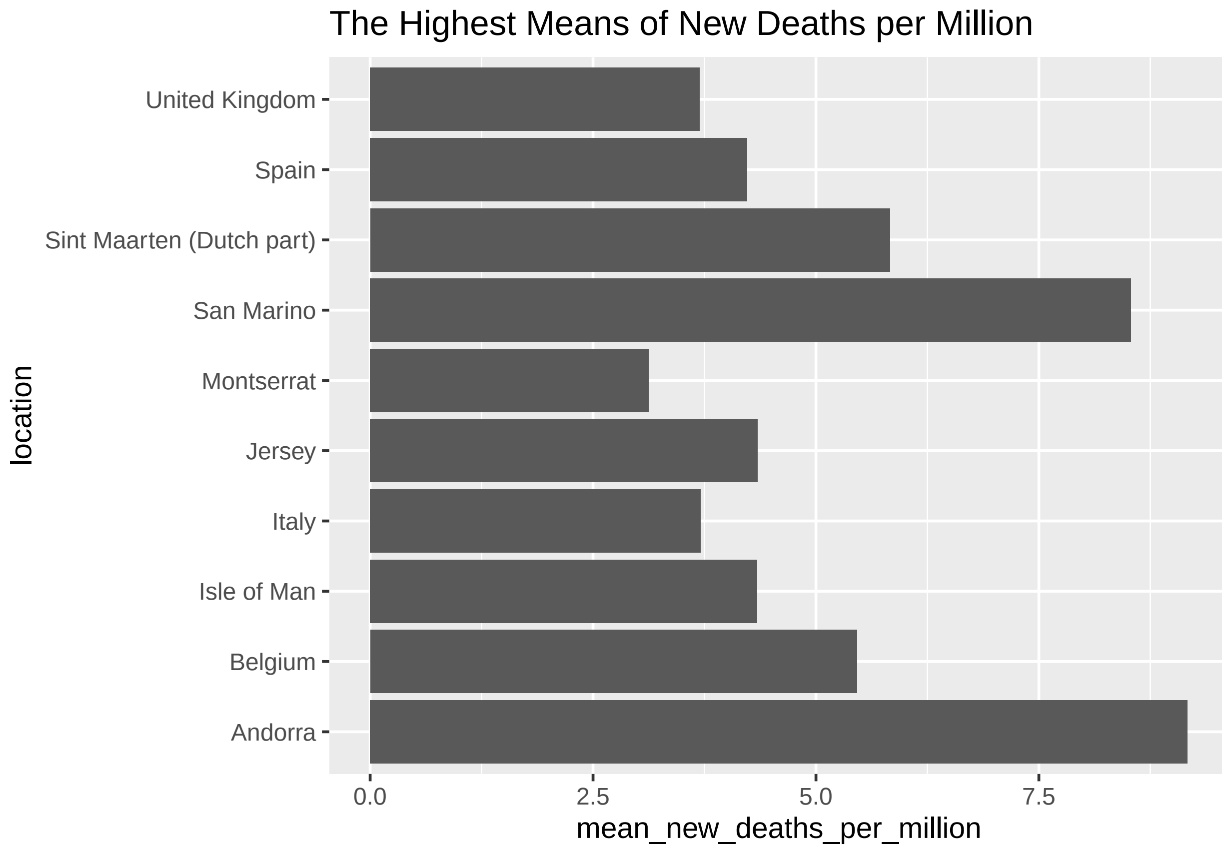
```
#Top 10 locations with the highest total cases since the start of the pandemic till today (i.e. Sys.Dat
data %>% filter(date == Sys.Date() - 1) %>% select(date, location, total_cases) %>% arrange(desc(total_
```




With that insight, let's now examine the variables of interest for this project. Let's begin by visualising the top 10 locations with the highest means of new deaths per million. For comparison purposes, the number of new deaths per million is preferred over the total (absolute) number of new deaths.

```
#top 10 location with the highest mean of new deaths per million, omitting NA values
data %>% group_by(location) %>% summarise(mean_new_deaths_per_million = mean(new_deaths_per_million, na
```

```
## Selecting by mean_new_deaths_per_million
```

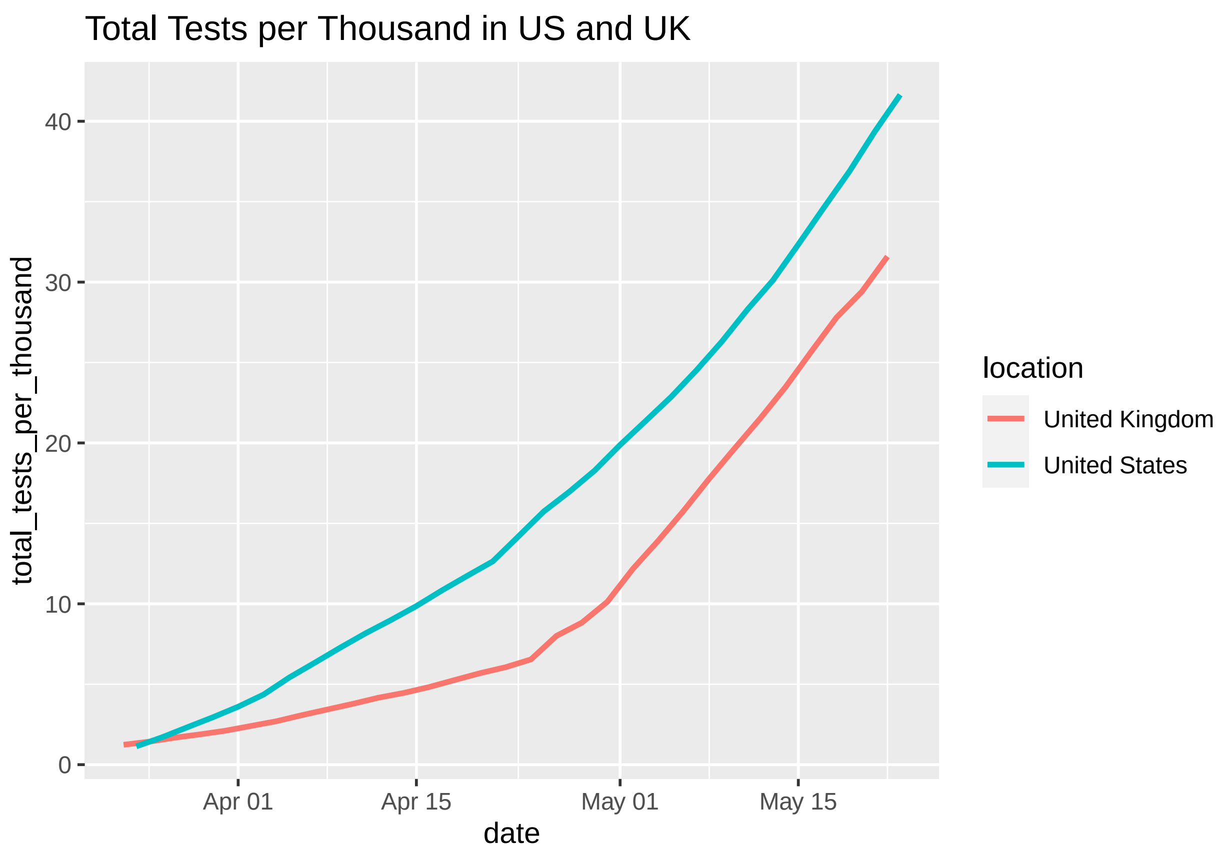



Andorra and San Marino are two awful places to live in due to the relatively high averages of new deaths per million (as of 25 May 2020).

We can surmise that the total number of new deaths (per million) is affected by not only the location (which has certain demographic charateristics such as median age), but also the number of tests that are conducted and the number of hospital beds that are available.

Let's quickly visualise the aforementioned variables.

```
#Let's focus on the total number of tests per thousand in the US and the UK - two great countries - sin
data %>% filter(date %between% c(as.Date("2020-03-23"), Sys.Date() - 1) & location == c("United States"
```

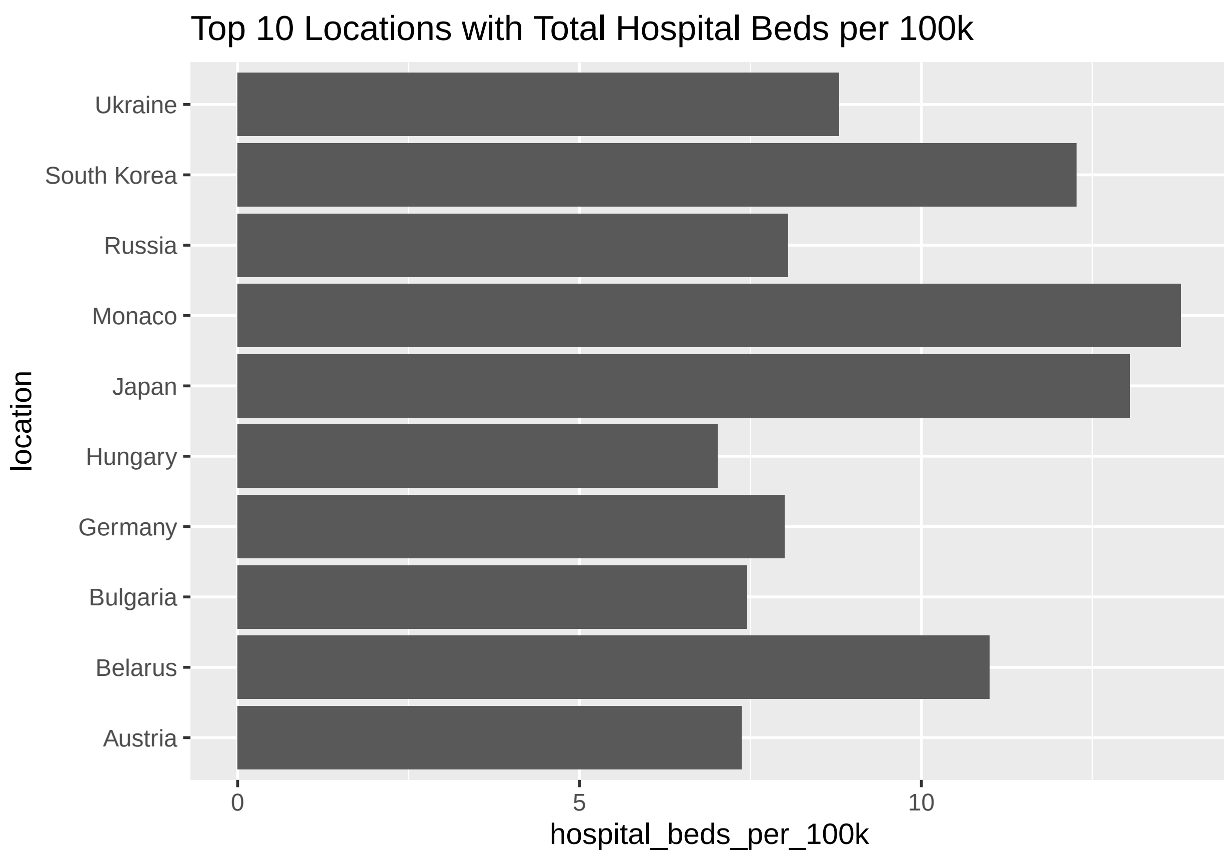



Nothing surprising here. Both the United States and the United Kingdom have ramped up total number of tests per thousand since 23 March 2020. The United States has conducted more tests than the United Kingdom! (as of 25 May 2020)

Let's find out top 10 locations with the highest number of hospital beds per 100k.

```
#The total numbers of hospital beds are mostly consistent across dates. To avoid the risk of having an 
data %>% filter(date == as.Date("2020-03-23") & !is.na(hospital_beds_per_100k)) %>% arrange(desc(hospita
```

```
## Selecting by hospital_beds_per_100k
```




The United States and the United Kingdom are not in the chart! It is surprising to know that Ukraine, Belarus, and Russia - communist countries (or ex-communist countries) - are among the top 10 countries with the highest number of hospital beds per 100k.

### 1.3 c. Data Cleaning and Processing

Based on our surmise, the number of new deaths per million (the dependent variable of interest) can be predicted by: - location (which includes demographic characteristics) - total cases per million (some of earlier confirmed cases can result in new deaths) - total tests per thousand (total tests are preferred over new tests as there is a lag time between the test and its results that eventually would contribute to the total number of cases) - hospital beds per 100k (the lack of hospital beds availability contributes directly to mortality)

So our machine learning formula is: new_deaths_per_million ~ location + total_cases_per_million + total_tests_per_thousand + hospital_beds_per_100k

Let's check for NA values for those variables of interest.

```
#check for NA values
sum(is.na(data$new_deaths_per_million))
```

```
## [1] 377
```

```
sum(is.na(data$location))
```

```
## [1] 0
```

```r
sum(is.na(data$total_cases_per_million))
```

```
## [1] 377
```

```r
sum(is.na(data$total_tests_per_thousand))
```

```
## [1] 14332
```

```r
sum(is.na(data$hospital_beds_per_100k))
```

```
## [1] 3160
```

It seems that we need to do some data cleaning to remove those NA values. But first, we don't need the whole dataset. There is too much noise in the dataset if we were to do that. Let's instead focus on the past one week data, filtering out "International" and "World" as those are total figures.

```r
#Select today's date >=7 and remove International and World
#Select the variables of interest: date, location, population, total_cases_per_million, new_deaths, new_
todaydata <- data %>% filter(date >= Sys.Date() - 7 & location != c("International", "World")) %>% sele

#Take a peek at the dataset of interest
head(todaydata)
```

```
##         date location population total_cases_per_million new_deaths
## 1 2020-05-18    Aruba     106766                 945.994          0
## 2 2020-05-19    Aruba     106766                 945.994          0
## 3 2020-05-20    Aruba     106766                 945.994          0
## 4 2020-05-21    Aruba     106766                 945.994          0
## 5 2020-05-22    Aruba     106766                 945.994          0
## 6 2020-05-23    Aruba     106766                 945.994          0
##   new_deaths_per_million total_tests_per_thousand hospital_beds_per_100k
## 1                      0                       NA                     NA
## 2                      0                       NA                     NA
## 3                      0                       NA                     NA
## 4                      0                       NA                     NA
## 5                      0                       NA                     NA
## 6                      0                       NA                     NA
```

Let's check for NA values for this dataset of interest.

```r
#Check for NA values
sum(is.na(todaydata$new_deaths_per_million))
```

```
## [1] 2
```

```r
sum(is.na(todaydata$location))
```

```
## [1] 0
```

```r
sum(is.na(todaydata$total_cases_per_million))
```

```
## [1] 2
```

```r
sum(is.na(todaydata$total_tests_per_thousand))
```

```
## [1] 1096
```

```r
sum(is.na(todaydata$hospital_beds_per_100k))
```

```
## [1] 317
```

Let's first do two things: - replace NA values with 0 for new_deaths_per_million and total_cases_per_million. It makes sense to do this as 0 is the likely value for blank (NA) new_deaths_per_million and the total_cases_per_million for the past one week. - impute values for total_tests_per_thousand with mean values as it is unlikely that the missing values are 0.

```r
#Let's first replace na values with 0 for new_deaths_per_million. (It seems that blank new_deaths_per_m
#Let's also replace na values with 0 for total_tests_per_thousand to enable us to compute the mean valu
#Store the values in todaydata2.
todaydata2 <- todaydata %>% drop_na(new_deaths_per_million) %>% group_by(location) %>% mutate(total_tes

#compute the mean for each location for total_tests_per_thousand and store the values in todaydata3.
todaydata3 <- todaydata2 %>% group_by(location) %>% summarise(mean_total_tests_per_thousand = mean(tota
head(todaydata3)
```

```
## # A tibble: 6 x 2
##   location    mean_total_tests_per_thousand
##   <fct>                               <dbl>
## 1 Afghanistan                             0
## 2 Albania                                 0
## 3 Algeria                                 0
## 4 Andorra                                 0
## 5 Angola                                  0
## 6 Anguilla                                0
```

```r
#Replace 0 values for total_tests_per_thousand with the mean values and store these values in todaydata
todaydata4 <- left_join(todaydata2, todaydata3, by = "location") %>% group_by(location) %>% mutate(total

#Store the dataset back to the original dataset, which is "todaydata"
todaydata <- todaydata4 %>% select(-mean_total_tests_per_thousand)
todaydata
```

```
## # A tibble: 1,463 x 8
## # Groups:   location [210]
##    date       location population total_cases_per~ new_deaths new_deaths_per_~
##    <date>     <fct>         <dbl>            <dbl>      <int>            <dbl>
##  1 2020-05-18 Aruba        106766             946.          0                0
##  2 2020-05-19 Aruba        106766             946.          0                0
##  3 2020-05-20 Aruba        106766             946.          0                0
```

```
##   4 2020-05-21 Aruba        106766              946.          0             0
##   5 2020-05-22 Aruba        106766              946.          0             0
##   6 2020-05-23 Aruba        106766              946.          0             0
##   7 2020-05-24 Aruba        106766              946.          0             0
##   8 2020-05-18 Afghani~   38928341              171.          1             0.026
##   9 2020-05-19 Afghani~   38928341              182.          4             0.103
##  10 2020-05-20 Afghani~   38928341              197.          5             0.128
## # ... with 1,453 more rows, and 2 more variables:
## #   total_tests_per_thousand <dbl>, hospital_beds_per_100k <dbl>
```

```
#remove the intermediary variables to avoid clutter
rm(todaydata2, todaydata3, todaydata4)
```

Check for NA values for the dataset of interest.

```
#Check for NA values
sum(is.na(todaydata$new_deaths_per_million))
```

```
## [1] 0
```

```
sum(is.na(todaydata$location))
```

```
## [1] 0
```

```
sum(is.na(todaydata$total_cases_per_million))
```

```
## [1] 0
```

```
sum(is.na(todaydata$total_tests_per_thousand))
```

```
## [1] 0
```

```
sum(is.na(todaydata$hospital_beds_per_100k))
```

```
## [1] 315
```

Finally, let's replace missing values with 0 for hospital_beds_per_100k. This is a reasonable estimate as countries with few hospital beds are the ones that are most likely not reporting the figures for hospital beds.

```
#replace missing values for hospital_beds_per_100k with 0
todaydata <- todaydata %>% mutate(hospital_beds_per_100k = replace_na(hospital_beds_per_100k, 0))
todaydata
```

```
## # A tibble: 1,463 x 8
## # Groups:   location [210]
##    date       location population total_cases_per~ new_deaths new_deaths_per_~
##    <date>     <fct>         <dbl>            <dbl>      <int>            <dbl>
##  1 2020-05-18 Aruba        106766             946.          0             0
##  2 2020-05-19 Aruba        106766             946.          0             0
```

```
##   3 2020-05-20 Aruba        106766              946.          0             0
##   4 2020-05-21 Aruba        106766              946.          0             0
##   5 2020-05-22 Aruba        106766              946.          0             0
##   6 2020-05-23 Aruba        106766              946.          0             0
##   7 2020-05-24 Aruba        106766              946.          0             0
##   8 2020-05-18 Afghani~   38928341              171.          1             0.026
##   9 2020-05-19 Afghani~   38928341              182.          4             0.103
##  10 2020-05-20 Afghani~   38928341              197.          5             0.128
## # ... with 1,453 more rows, and 2 more variables:
## #   total_tests_per_thousand <dbl>, hospital_beds_per_100k <dbl>
```

Make sure that we do not have any NA values in the dataset of interest, i.e. "todaydata".

```
#Check for NA values
sum(is.na(todaydata$new_deaths_per_million))
```

```
## [1] 0
```

```
sum(is.na(todaydata$location))
```

```
## [1] 0
```

```
sum(is.na(todaydata$total_cases_per_million))
```

```
## [1] 0
```

```
sum(is.na(todaydata$total_tests_per_thousand))
```

```
## [1] 0
```

```
sum(is.na(todaydata$hospital_beds_per_100k))
```

```
## [1] 0
```

Ok, we are good to go for machine learning!

#3. Results Let's start by partitioning our dataset of interest (todaydata) into training and test sets.

```
#We need to set seed to ensure consistent partitioning of dataset
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)` instead
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
#Test set to be 20% of the data
test_index <- createDataPartition(y = todaydata$new_deaths_per_million, times = 1, p = 0.2, list = FALSE
train_set <- todaydata[c(-test_index),]
test_set <- todaydata[c(test_index),]
```

Alright, now we have our training and test sets.

Just to recap our machine learning formula is new_deaths_per_million ~ location + total_cases_per_million + total_tests_per_thousand + hospital_beds_per_100k

Random forest and linear regression are proposed to be the machine learning algorithms to predict the new_deaths_per_million. Random forest is proposed due to its built-in ensembling capacity, which is suitable in examining and predicting the values in the dataset. We are going to compare its performance with that of linear regression using the RMSE method.

Let's start with random forest algorithm.

```
#rerun the set seed again to ensure a consistent result
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
#fit the random forest algorithm
#tune the parameters by minimising the RMSE metric and ntree = 285 - which was discovered by trial and
#warning: this may take some time - around 30 minutes depending on the computer specifications.
fit <- train(new_deaths_per_million ~ location + total_cases_per_million + total_tests_per_thousand + ho

#Let's see how accurate the prediction is by checking the result for one observation in the test set.
y_hat_rf_1 <- predict(fit, data.frame(location = "Malaysia", total_cases_per_million = 215.597, total_te
y_hat_rf_1
```

```
##          1
## 0.03503344
```

```
#Given that the actual result for the observation is 0.031, the predicted result: 0.03503344 seems quit
```

Let's now use linear regression algorithm.

```
#rerun the set seed again to ensure a consistent result
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
#fit the linear regression algorithm
#ignoring the rank deficient warnings are we are using standardised figures, i.e. per million, per thou
fit_lm <- train(new_deaths_per_million ~ location + total_cases_per_million + total_tests_per_thousand +
```

```
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading
```

```
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading
```

```
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading
```

```
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading
```

```
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading
```

```
#Let's see how accurate the prediction is by checking the result for one observation in the test set.
y_hat_lm_1 <- predict(fit_lm, data.frame(location = "Malaysia", total_cases_per_million = 215.597, total
```

```
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading
```

```
y_hat_lm_1
```

```
##          1
## 0.03764966
```

```
#Given that the actual result for the observation is 0.031, the predicted result: 0.03764966 seems clos
```

We need a more objective way to evaluate the performance of the two proposed algorithms. This can be done via RMSE, which is taught in the course.

```
#RMSE formula
RMSE <- function(true_ratings, predicted_ratings){
sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

Let's see how well our random forest algorithm performs

```
#Random forest performance
y_hat_rf <- predict(fit, test_set)

#RMSE of the random forest algorithm
RMSE(y_hat_rf, test_set$new_deaths_per_million)
```

```
## [1] 0.7913566
```

And compare that with the performance of the linear regression model.

```
#Linear regression performance
y_hat_lm <- predict(fit_lm, test_set)
```

```
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading
```

```r
#RMSE of the random forest algorithm
RMSE(y_hat_lm, test_set$new_deaths_per_million)
```

```
## [1] 1.565675
```

#4. Conclusion We have shown that new_deaths_per_million is correlated with location, total_cases_per_million, total_tests_per_thousand, and hospital_beds_per_100k. And it is easy to see why. Location captures most of the information pertaining to the demographic characteristics that contribute to the number of deaths. Number of total cases, tests, and hospital beds are directly correlated with mortality too.

With that understanding, we have also proceeded to predict the proposed dependent variable - new_deaths_per_million - using random forest and linear regression algorithms. The random forest algorithm with ntree=285 gives a far more accurate prediction than the linear regression algorithm, as we can see from the RMSE of the random forest algorithm - 0.7913566 - which is almost twice better than the RMSE of the linear regression algorithm - 1.565675. Given that lives are potentially at stake here, it is clear that the random forest algorithm should be used for predicting the number of new deaths instead of the linear regression algorithm.

There is scope to further build on this project to better predict the total number of new deaths. This project is limited by one week worth of data, and by the proposed random forest algorithm which is slow and prone to inconsistent results induced by the daily update of the dataset (if we were to run the prediction model on a daily basis). As such, the random forest algorithm has to be tuned regularly based on the new update of the dataset to derive accurate predictions - which takes time and could cause frustrations given the high stakes.

Future work can overcome such limitations by using time series analysis (e.g. ARIMA) that allows the whole dataset - not just one week worth of data - to be digested and incorporated into a machine learning algorithm suitable for time series. One should also consider using a machine learning algorithm that can outperform the random forest algorithm, such as XGBoost - an algorithm that is quite popular in Kaggle.