Capstone Project - CYO

Jean-Claude de Villeres

March 19, 2020

Contents

1 Introduction

This capstone report aims to explore the Coronavirus dataset from Kaggle and do a straight-forward forecast on the number of cases for the next 50 days. We will explore the data to find various trends and patterns and also to forecast using different models future Coronavirus cases. The dataset consists of around 4,935 rows of records of daily cases from January 22 to March 11. The data is mainly sourced out from Johns Hopkins University for education and academic research purposes, where they aggregated data from variou globalresources such as, World Health Organization, US Center for Disease Control, Canadian Government, China CDC, Italy Ministry of Health and others.

The following are the column descriptions:

- Sno - Serial number
- ObservationDate - Date of the observation in MM/DD/YYYY
- Province/State - Province or state of the observation (Could be empty when missing)
- Country/Region - Country of observation
- Last Update - Time in UTC at which the row is updated for the given province or country
- Confirmed - Cumulative number of confirmed cases till that date
- Deaths - Cumulative number of of deaths till that date
- Recovered - Cumulative number of recovered cases till that date

Here is the link for the dataset https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset

2 Methods and Analysis

We will explore the data using dplyr summarize / groupby methods and ggplot to analyze patterns in the dataset. We will also try to print a global map of cases before and after the pandemic. Equipped with the resulting information, we will generate various models to forecast the number of cases for the next 50 days.

2.1 Data Exploration

We will group data to find various patterns as well as wrangle with the dataset and perform summarisations to find relevant trends.

```
head(covid_data)
```

```
##   SNo ObservationDate Province.State Country.Region     Last.Update Confirmed
## 1   1      2020-01-22          Anhui Mainland China 1/22/2020 17:00         1
## 2   2      2020-01-22        Beijing Mainland China 1/22/2020 17:00        14
## 3   3      2020-01-22      Chongqing Mainland China 1/22/2020 17:00         6
## 4   4      2020-01-22         Fujian Mainland China 1/22/2020 17:00         1
## 5   5      2020-01-22          Gansu Mainland China 1/22/2020 17:00         0
## 6   6      2020-01-22      Guangdong Mainland China 1/22/2020 17:00        26
##   Deaths Recovered
## 1      0         0
## 2      0         0
## 3      0         0
## 4      0         0
## 5      0         0
## 6      0         0
```

```
summary(covid_data)
```

```
##       SNo       ObservationDate        Province.State        Country.Region
##  Min.   :   1   Min.   :2020-01-22            :1815   Mainland China:1548
##  1st Qu.:1234   1st Qu.:2020-02-11   Gansu    :  51   US            :1003
##  Median :2468   Median :2020-02-26   Hebei    :  51   Australia     : 225
##  Mean   :2468   Mean   :2020-02-22   Anhui    :  50   Canada        : 146
##  3rd Qu.:3702   3rd Qu.:2020-03-06   Beijing  :  50   France        :  50
##  Max.   :4935   Max.   :2020-03-11   Chongqing:  50   Japan         :  50
##                                      (Other)  :2868   (Other)       :1913
##               Last.Update     Confirmed           Deaths          Recovered
##  2020-02-01T19:43:03:  63   Min.   :    0.0   Min.   :   0.00   Min.   :    0
##  2020-02-01T19:53:03:  63   1st Qu.:    1.0   1st Qu.:   0.00   1st Qu.:    0
```

```
##  2020-02-24T23:33:02:  63   Median :    9.0   Median :   0.00   Median :    1
##  1/31/2020 23:59    :  62   Mean   :  577.6   Mean   :  17.69   Mean   :  201
##  1/30/2020 16:00    :  58   3rd Qu.:   93.0   3rd Qu.:   1.00   3rd Qu.:   14
##  1/29/2020 19:30    :  54   Max.   :67773.0   Max.   :3046.00   Max.   :49134
##  (Other)            :4572
```

2.1.1 Statistics as of March 11

```
# Filter data for March 11
covid_data_recent <- covid_data %>% filter(ObservationDate =="2020-03-11")
head(covid_data_recent)
```

```
##    SNo ObservationDate Province.State Country.Region         Last.Update
## 1 4720      2020-03-11          Hubei Mainland China 2020-03-11T10:53:02
## 2 4721      2020-03-11                         Italy 2020-03-11T21:33:02
## 3 4722      2020-03-11                          Iran 2020-03-11T18:52:03
## 4 4723      2020-03-11                   South Korea 2020-03-11T21:13:18
## 5 4724      2020-03-11        France          France 2020-03-11T22:53:03
## 6 4725      2020-03-11                         Spain 2020-03-11T20:53:02
##   Confirmed Deaths Recovered
## 1     67773   3046     49134
## 2     12462    827      1045
## 3      9000    354      2959
## 4      7755     60       288
## 5      2281     48        12
## 6      2277     54       183
```

```
# Show number of confirmed
sum(covid_data_recent$Confirmed)
```

```
## [1] 125865
```

```
# Show number of recovered
sum(covid_data_recent$Recovered)
```

```
## [1] 67003
```

```
# Show number of deaths
sum(covid_data_recent$Deaths)
```

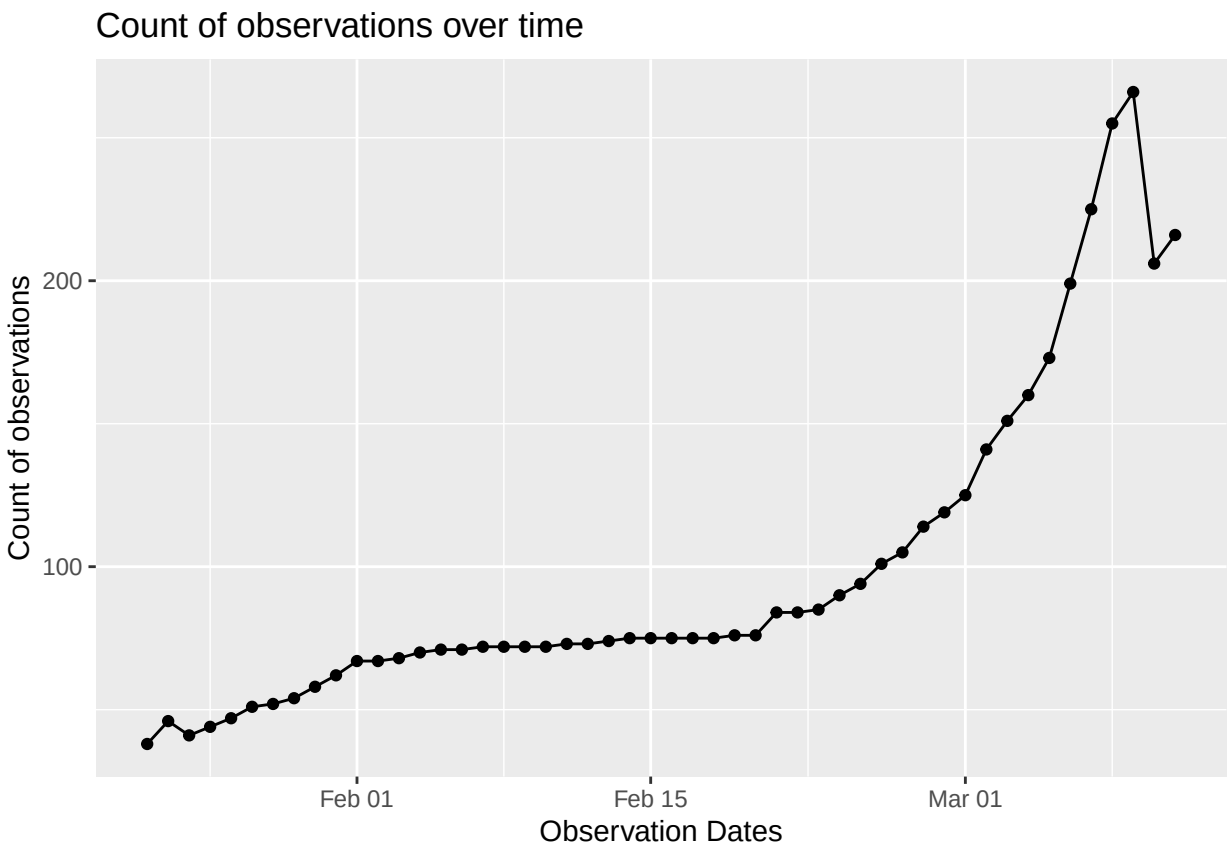
```
## [1] 4615
```

2.1.2 Plot number of cases over time

```
# Show number of observations over time
covid_observations <- covid_data %>%
  group_by(ObservationDate) %>%
  summarize(count =n())
head(covid_observations %>% arrange(desc(count)))
```

```
## # A tibble: 6 x 2
##   ObservationDate count
##   <date>          <int>
## 1 2020-03-09        266
## 2 2020-03-08        255
## 3 2020-03-07        225
## 4 2020-03-11        216
## 5 2020-03-10        206
## 6 2020-03-06        199
```

```
# Plot the number of observations over time
covid_observations %>% ggplot(aes(x=ObservationDate, y=count, group=1)) +
  geom_point() + geom_line() +  labs(x = "Observation Dates", y = "Count of observations",
              title = "Count of observations over time")
```

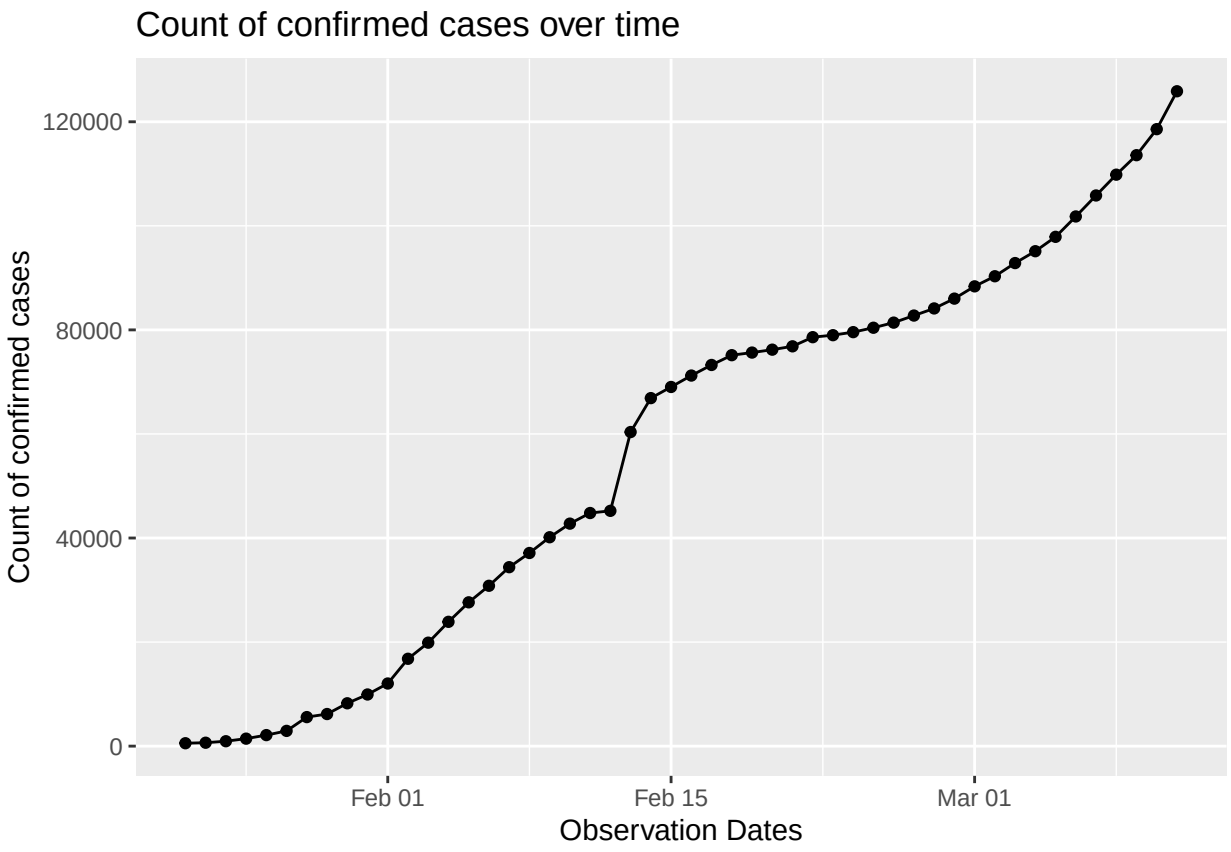



```
# Show number of confirmed cases over time
covid_confirmed <- covid_data %>%
  group_by(ObservationDate) %>%
  summarize(total_confirmed = sum(Confirmed))
head(covid_confirmed %>% arrange(desc(total_confirmed)))
```

```
## # A tibble: 6 x 2
##   ObservationDate total_confirmed
##   <date>                    <int>
## 1 2020-03-11               125865
## 2 2020-03-10               118582
## 3 2020-03-09               113582
## 4 2020-03-08               109835
```

```
## 5 2020-03-07                105836
## 6 2020-03-06                101800
```

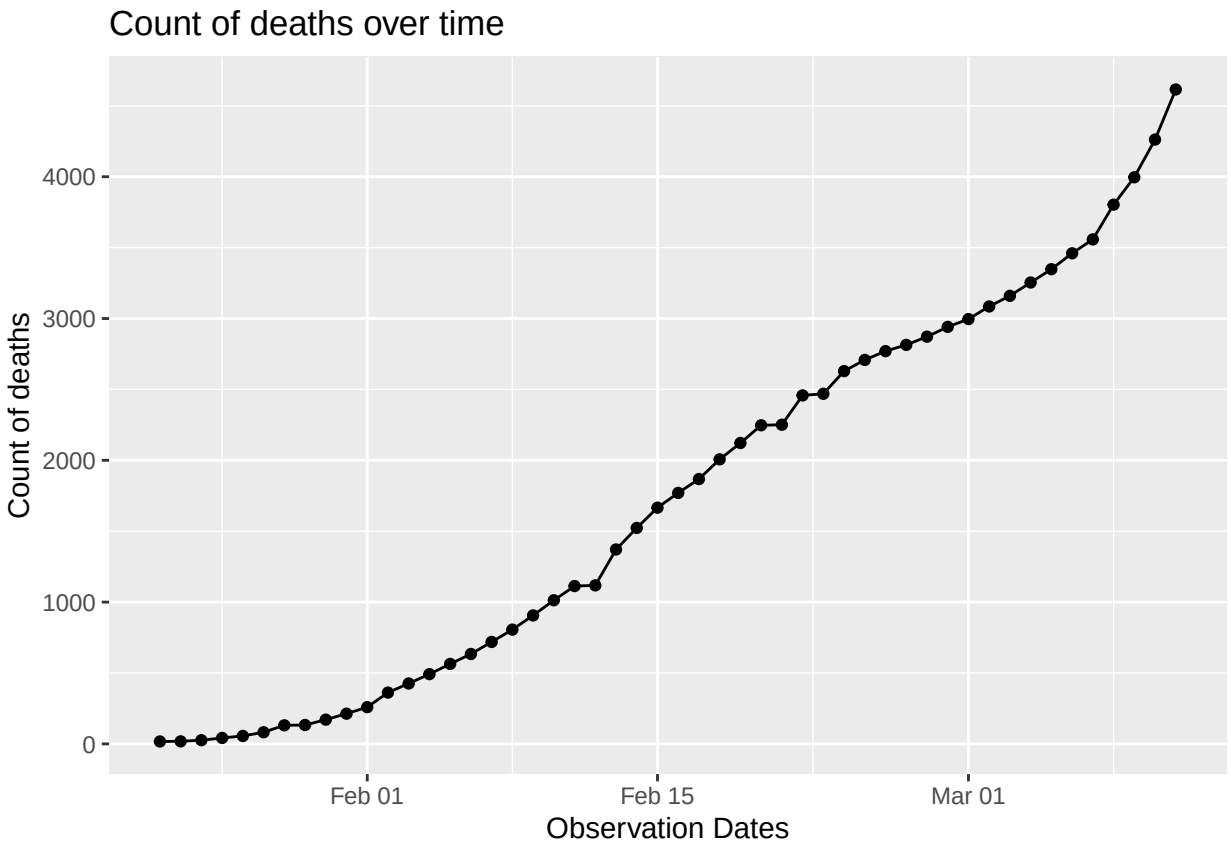
```
# Plot the number of confirmed cases over time
covid_confirmed %>% ggplot(aes(x=ObservationDate, y=total_confirmed, group=1)) +
  geom_point() + geom_line() +  labs(x = "Observation Dates", y = "Count of confirmed cases",
              title = "Count of confirmed cases over time")
```




```
# Show number of deaths cases over time
covid_deaths <- covid_data %>%
  group_by(ObservationDate) %>%
  summarize(total_deaths = sum(Deaths))
head(covid_deaths %>% arrange(desc(total_deaths)))
```

```
## # A tibble: 6 x 2
##   ObservationDate total_deaths
##   <date>                 <int>
## 1 2020-03-11              4615
## 2 2020-03-10              4262
## 3 2020-03-09              3996
## 4 2020-03-08              3803
## 5 2020-03-07              3558
## 6 2020-03-06              3460
```

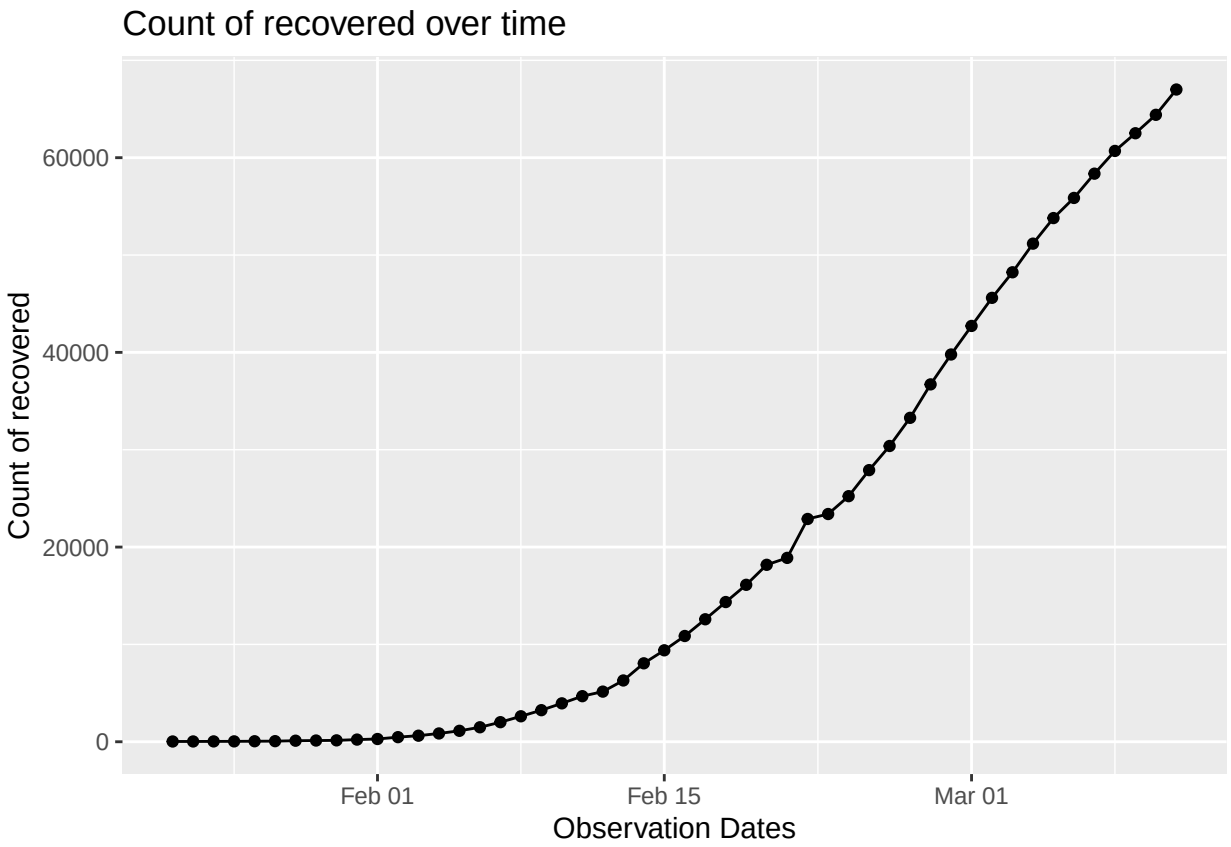
```
# Plot the number of deaths cases over time
covid_deaths %>% ggplot(aes(x=ObservationDate, y=total_deaths, group=1)) +
  geom_point() + geom_line() +  labs(x = "Observation Dates", y = "Count of deaths",
              title = "Count of deaths over time")
```




```
# Show number of recovered cases over time
covid_recovered <- covid_data %>%
  group_by(ObservationDate) %>%
  summarize(total_recovered = sum(Recovered))
head(covid_recovered %>% arrange(desc(total_recovered)))
```

```
## # A tibble: 6 x 2
##   ObservationDate total_recovered
##   <date>                    <int>
## 1 2020-03-11                67003
## 2 2020-03-10                64404
## 3 2020-03-09                62512
## 4 2020-03-08                60695
## 5 2020-03-07                58359
## 6 2020-03-06                55866
```

```
# Plot the number of recovered cases over time
covid_recovered %>% ggplot(aes(x=ObservationDate, y=total_recovered, group=1)) +
  geom_point() + geom_line() +
  labs(x = "Observation Dates", y = "Count of recovered",
               title = "Count of recovered over time")
```



Let us combine our confirmed, deaths, and recovered.

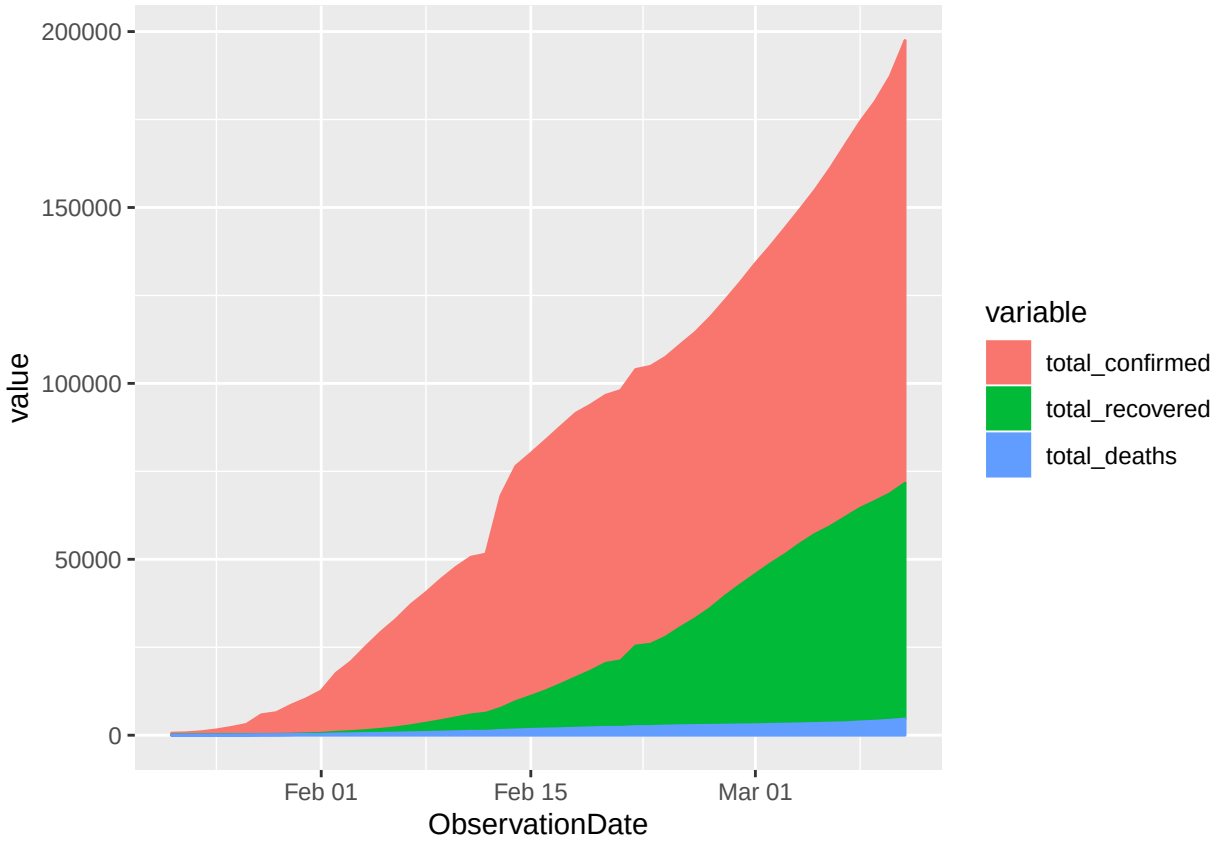
```
# Show number of all cases over time
covid_all <- covid_data %>%
  group_by(ObservationDate) %>%
  summarize(total_confirmed=sum(Confirmed), total_recovered=sum(Recovered), total_deaths=sum(Deaths))
head(covid_all)
```

```
## # A tibble: 6 x 4
##   ObservationDate total_confirmed total_recovered total_deaths
##   <date>                    <int>           <int>        <int>
## 1 2020-01-22                  555              28           17
## 2 2020-01-23                  653              30           18
## 3 2020-01-24                  941              36           26
## 4 2020-01-25                 1438              39           42
## 5 2020-01-26                 2118              52           56
## 6 2020-01-27                 2927              61           82
```

```
covid_all_melted <- melt(covid_all, id.var='ObservationDate')
head(covid_all_melted)
```

```
##   ObservationDate        variable value
## 1      2020-01-22 total_confirmed   555
## 2      2020-01-23 total_confirmed   653
## 3      2020-01-24 total_confirmed   941
## 4      2020-01-25 total_confirmed  1438
## 5      2020-01-26 total_confirmed  2118
## 6      2020-01-27 total_confirmed  2927
```

```
covid_all_melted %>% ggplot(aes(x=ObservationDate, y=value, col=variable)) +
  geom_area(aes(fill=variable))
```



Based on the plots generated we can see that:

- There has been a large spike in the number of confirmed cases between February 8 to 10. The trajectory is also continually increasing.
- The number of deaths increased daily. We can also see the a smaller version of the spike aforementioned.
- On the number of recovered cases, there has been a minimal number from January 22 to February 3 despite the increase in the number of infected cases. Bare in mind that the incubation period is 14-days.

2.1.3 Countries with most cases

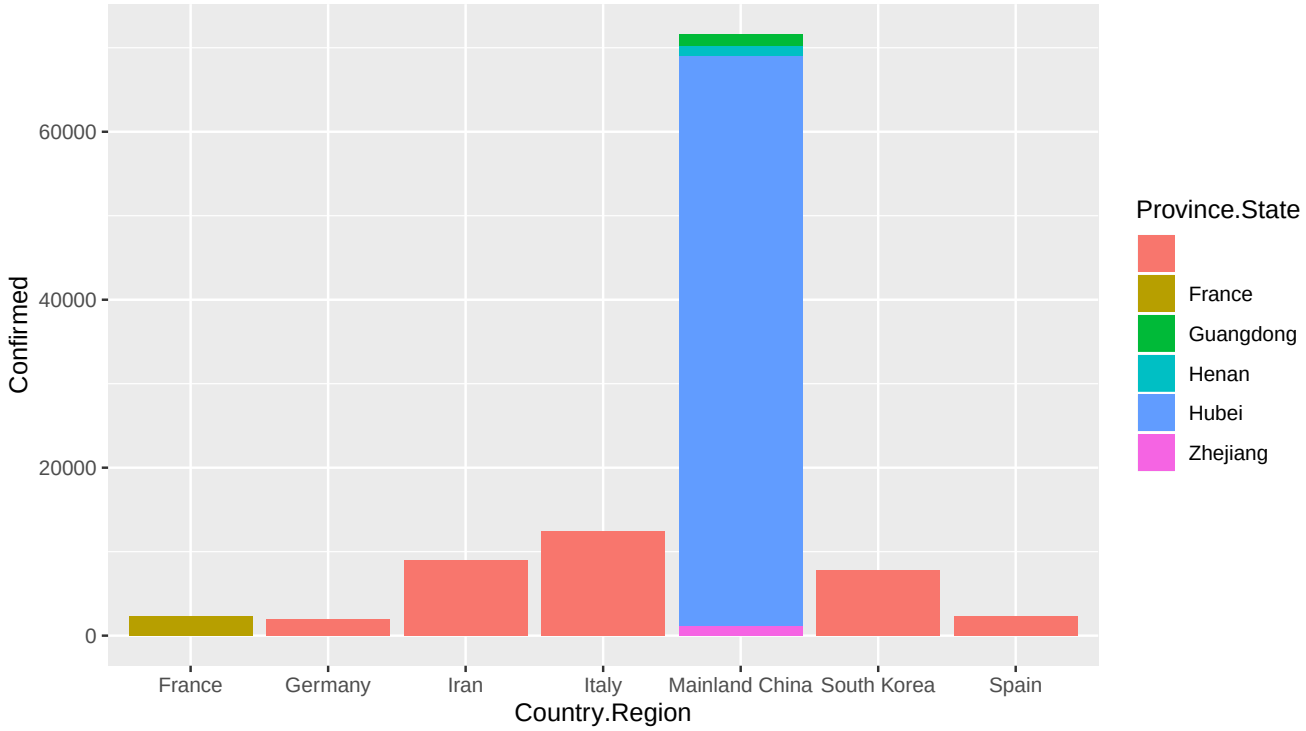
Which 10 countries have the most confirmed cases?

```
# Show top 10 Country.Region with cases
covid_country_top10 <- covid_data_recent %>%
  group_by(Country.Region, Province.State, Confirmed) %>%
    summarize(total_confirmed = sum(Confirmed)) %>% arrange(desc(total_confirmed))
covid_country_top10 <- covid_country_top10[1:10,1:3]
covid_country_top10
```

```
## # A tibble: 10 x 3
## # Groups:   Country.Region, Province.State [10]
##    Country.Region Province.State Confirmed
##    <fct>          <fct>              <int>
##  1 Mainland China "Hubei"            67773
```

```
##  2 Italy          ""                12462
##  3 Iran           ""                 9000
##  4 South Korea    ""                 7755
##  5 France         "France"           2281
##  6 Spain          ""                 2277
##  7 Germany        ""                 1908
##  8 Mainland China "Guangdong"        1356
##  9 Mainland China "Henan"            1273
## 10 Mainland China "Zhejiang"         1215
```

```
# Plot the number of country cases over time
covid_country_top10 %>% ggplot(aes(Country.Region, Confirmed)) +
  geom_bar(stat="identity", aes(fill=Province.State)) +
  theme(legend.position = "right")
```



Majority of the cases in China are from Hubei. Italy, Iran and South Korea also succumbed to the pandemic. France, Germany and Spain also reports few hundreds of cases.

2.1.4 Spatial analysis using ggplot and map_data

Note that the following plot: * It is much better viewed in .html version (uploaded in Github) or upon running the .Rmd file * If both fails, you can still use the PDF file but zoom in on the map to detect the changes.

```
# Load our time series data
covid_data_timeseries <- read_csv(str_c('time_series_covid_19_confirmed.csv'))
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   `Province/State` = col_character(),
##   `Country/Region` = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
# Due to plotting issues of all 50 maps, we will only get 10% and 90% percentile of date points (i.e., Day
covid_data_timeseries <- covid_data_timeseries[,-15:-44]
# Set the daily entries count
n_times <- ncol(covid_data_timeseries) - 4
n_times
```

```
## [1] 20
```

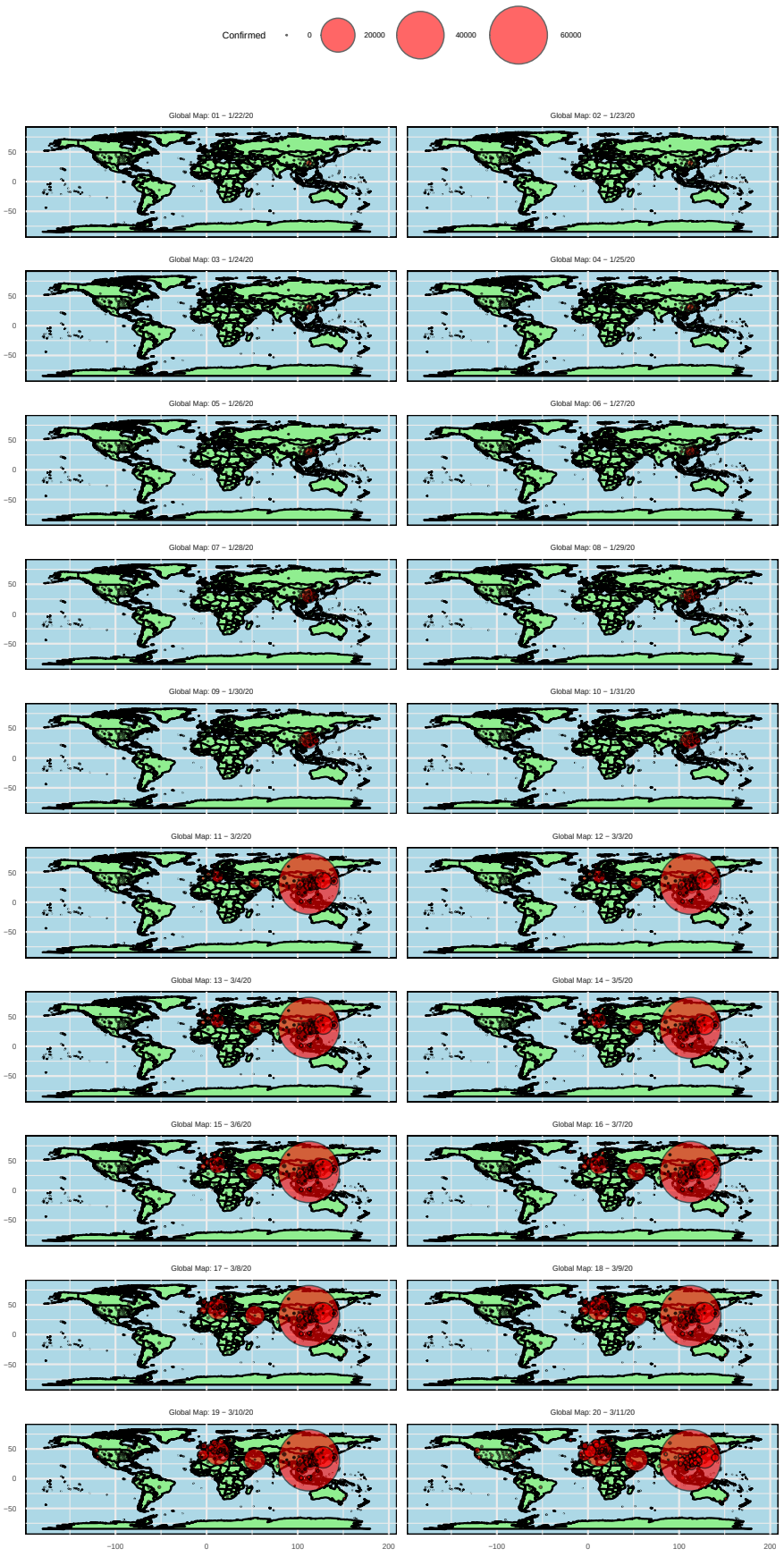
```
# Prepare timeseries data, parse latitude longtiude data
covid_data_ts_latlong <- covid_data_timeseries
colnames(covid_data_ts_latlong) <- c(colnames(covid_data_ts_latlong)[1:4],
  str_c('Global Map: ', str_pad(as.character(1:n_times), 2, 'left', '0'),
        str_c(' - ', colnames(covid_data_ts_latlong)[5:(4 + n_times)])))
# Pivot our latitude longitude data for ggplot use
covid_data_ts_latlong_pivot <- covid_data_ts_latlong %>% pivot_longer(names_to = 'Confirmed.Time',
                                                values_to = 'Confirmed',
                                                cols = colnames(covid_data_ts_latlong)[5:(4 + n_times)])
head(covid_data_ts_latlong_pivot)
```

```
## # A tibble: 6 x 6
##   `Province/State` `Country/Region`   Lat  Long Confirmed.Time         Confirmed
##   <chr>            <chr>            <dbl> <dbl> <chr>                      <dbl>
## 1 Anhui            Mainland China    31.8  117. Global Map: 01 - 1/22~         1
## 2 Anhui            Mainland China    31.8  117. Global Map: 02 - 1/23~         9
## 3 Anhui            Mainland China    31.8  117. Global Map: 03 - 1/24~        15
## 4 Anhui            Mainland China    31.8  117. Global Map: 04 - 1/25~        39
## 5 Anhui            Mainland China    31.8  117. Global Map: 05 - 1/26~        60
## 6 Anhui            Mainland China    31.8  117. Global Map: 06 - 1/27~        70
```

```
# Use ggplot map_data world, to display a global map
world <- map_data('world')
ggplot(legend = FALSE) +
  # Plot our map
  geom_polygon(data = world, aes(x = long, y = lat, group = group),
               color = 'black', fill = 'lightgreen') +  xlab('') + ylab('') +
  # Plot points of our confirmed cases using latitude/longitude data
  geom_point(data = covid_data_ts_latlong_pivot, fill = 'red', color = 'black',
             shape = 21, alpha = 0.6,
             aes(x = Long, y = Lat, fill = Confirmed, size = Confirmed)) +
  theme_minimal() +
  # Set the scale from 0 to 30 for the shapes in the map
  scale_size_continuous(range = c(0, 15)) + ggtitle('Occurrences Map - COVID19') +
  theme(text = element_text(size = 5), legend.position = 'top',
        panel.background = element_rect(fill='lightblue', colour='black')) +
  facet_wrap(~Confirmed.Time, ncol = 2)
```

```
## Warning: Removed 188 rows containing missing values (geom_point).
```

Occurrences Map − COVID19



2.2 Data Modeling

For this exercise we will use various timeseries forecasting methods such as Naive, HoltWinters, ARIMA and Simple Exponential Smoothing. We will evaluate each of their performance and compare predictions made. Mainly, we will be predicted the confirmed number of cases for the next 50 days since March 11, 2020.

```
# Since the default ts() function is not a good tool to use for daily sampled data,
# - we will use 'zoo' package for irregular time series data. This takes care of-
# indexing for time-series data. Create a daily Date object - helps our work on dates
inds <- seq(as.Date("2020-01-22"), as.Date("2020-03-11"), by = "day")
# We link the time-series with our confirmed cases
covid_confirmed_ts <- zoo(covid_confirmed$total_confirmed, inds)
```

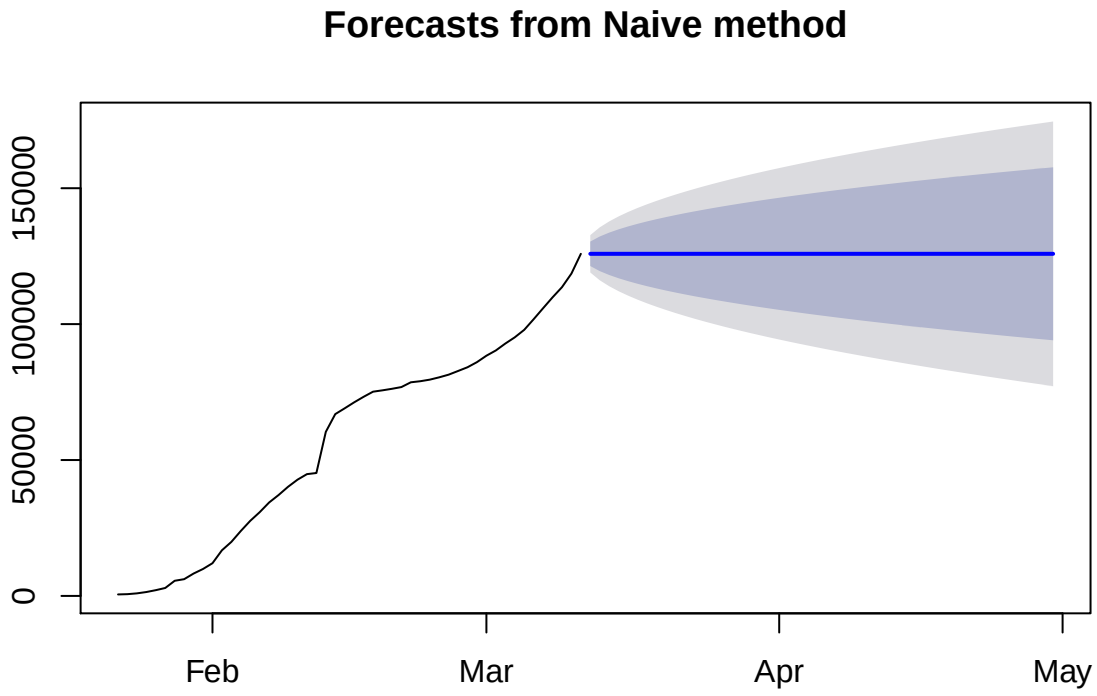
2.2.1 Naive

One of the simplest forecasting is to reuse recent observations for predict the next one. We start with this, the naive forecast which can ba generated using the naive() function.

```
# Initialize our Naive model. Forecast for the next 50 days
model_naive <- naive(covid_confirmed_ts, h = 50)
naive_forecast_max <- max(model_naive$mean)
naive_forecast_max
```

```
## [1] 125865
```

```
# The plot though will cause an issue as the x-axis is in days since the epoch (1970-01-01)
# So we need to suppress the auto plotting of this axis and then plot our own
plot(model_naive, xaxt ="n")
Axis(inds, side = 1)
```

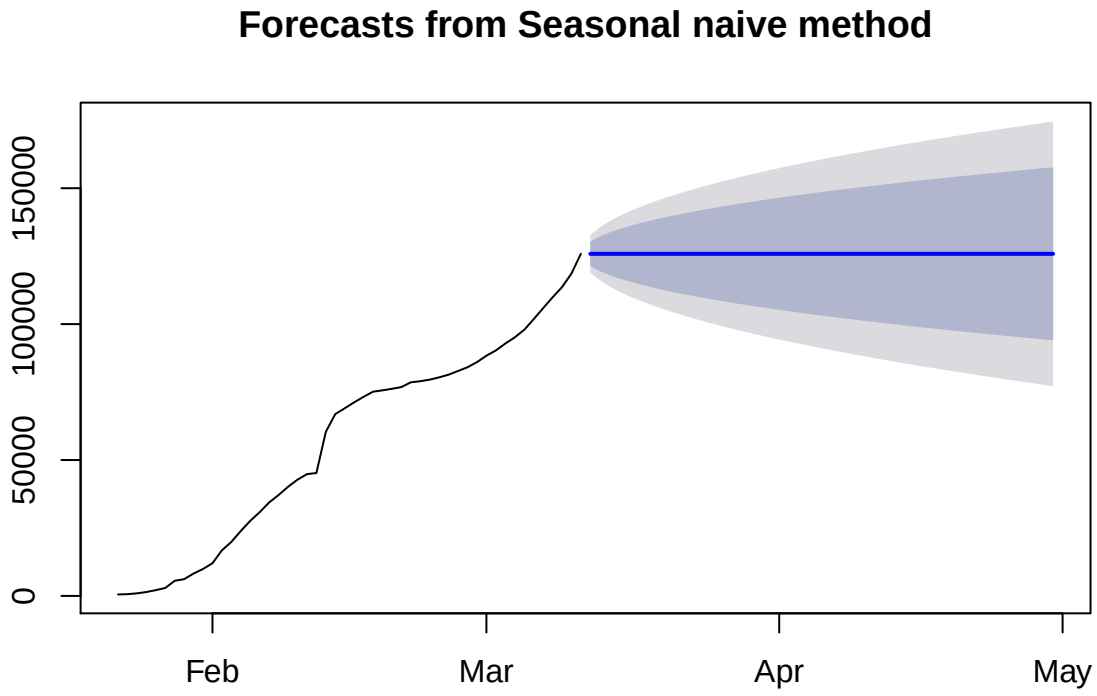


Using naive forecast, there is a more steady rate and highest number of predicted cases is 1.25865×10^5

```
# Initialize our Naive model. Forecast for the next 50 days
model_naive <- snaive(covid_confirmed_ts, h = 50)
max(model_naive$mean)
```

```
## [1] 125865
```

```
# The plot though will cause an issue as the x-axis is in days since the epoch (1970-01-01)
# So we need to suppress the auto plotting of this axis and then plot our own
plot(model_naive, xaxt ="n")
Axis(inds, side = 1)
```



Seasonal naive forecast is relatively the same.

Note: Seasonal forecasts take into account similarities between values of a full period as before, for instance January 2020 predictions have equal / similar values to January 2019 predictions.

2.2.2 Simple Exponential Smoothing

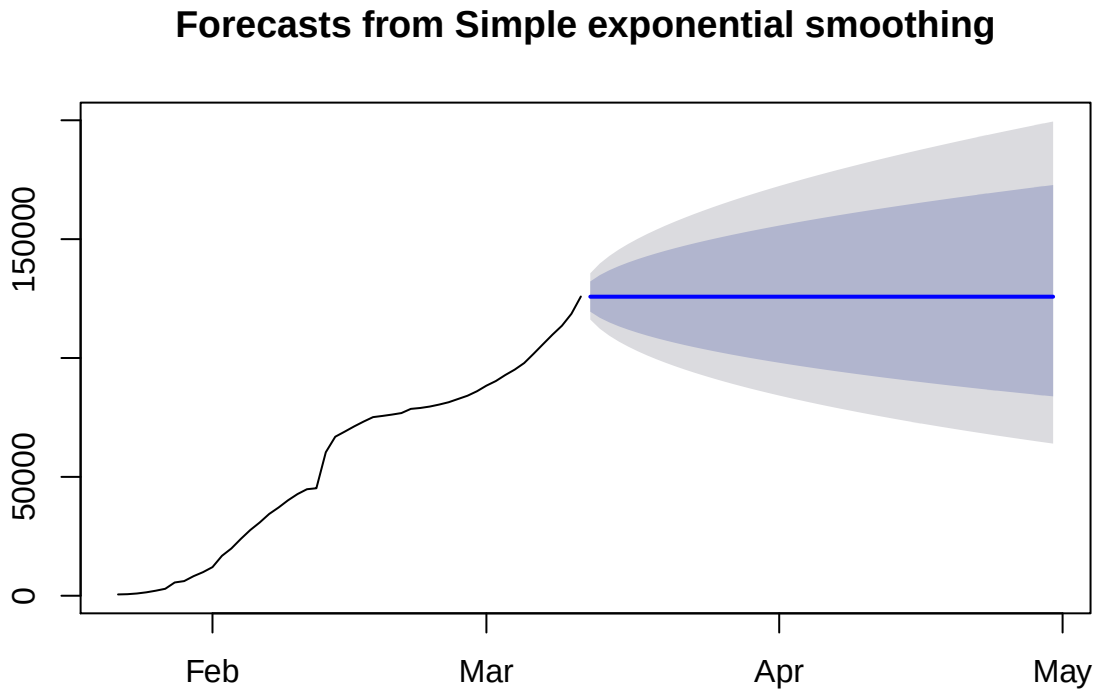
As opposed to naive which just uses the recent observations, exponential smoothing looks at older observations and has several configurable parameters that affect forecasting such as:

- alpha - Value of smoothing parameter for the level (0 - 1)
- lambda - Box-Cox transformation parameter.

```
# For simple exponential smoothing we also plug in our time series data and predict for the next 50 days
model_ses <- ses(covid_confirmed_ts, h = 50, alpha = 0.99, lambda="auto")
ses_forecast_max <- max(model_ses$mean)
ses_forecast_max
```

```
## [1] 125791
```

```
plot(model_ses, xaxt ="n")
Axis(inds, side = 1)
```



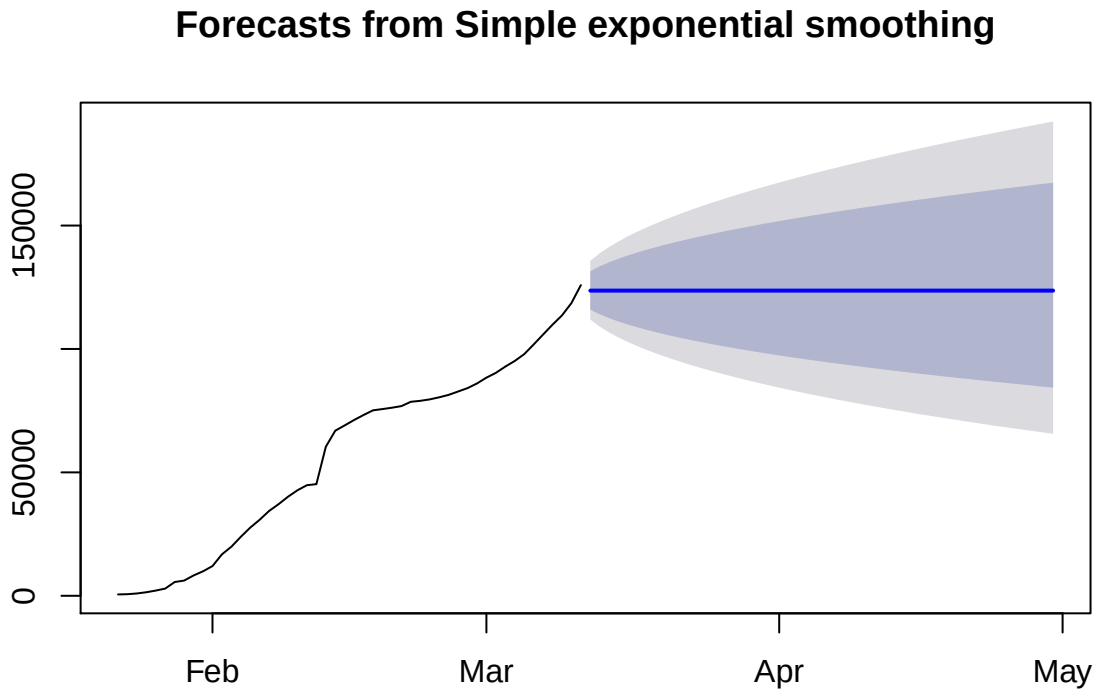
Using Simple Exponential Smooth the maximum predicted number of cases is 1.25791×10^5

Let us try with a different value for alpha, smoothing parameters

```
# For simple exponential smoothing we also plug in our time series data and predict for the next 50 days
# lambda="auto" Box Cox transmaton allows non-normal dependent variables into normal distribution
model_ses <- ses(covid_confirmed_ts, h = 50, alpha = 0.75, lambda="auto",
                 damped=FALSE,  exponential=TRUE, beta = 0.75)
max(model_ses$mean)
```

```
## [1] 123630
```

```
plot(model_ses, xaxt ="n")
Axis(inds, side = 1)
```



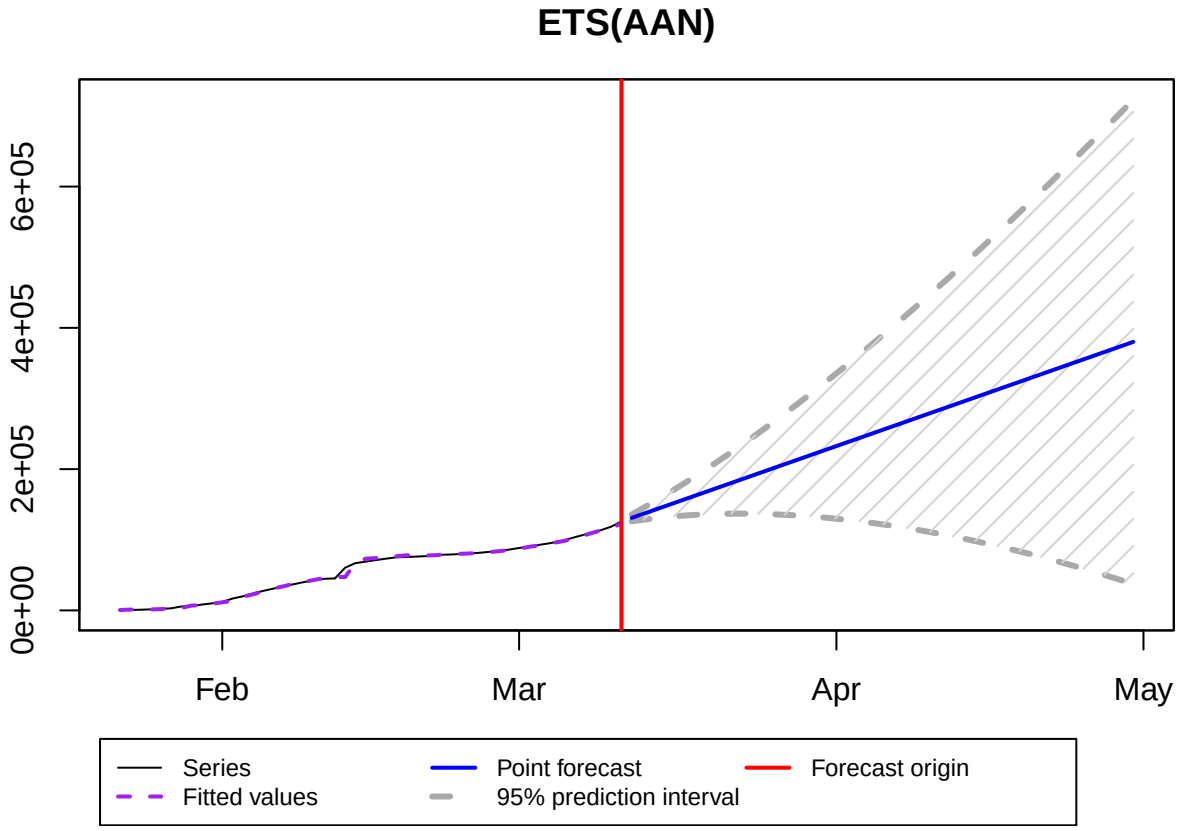
For simple exponential smoothing, we have tested varius parameters, and we got relatively similar forecast.

2.2.3 HoltWinters

Using HoltWinters method we can factor in the Trends, Seasonability and Level. These effects factor in questions such as annual sales trends (Black Friday Sales), weekends inactivity, and other recurring effects. It can be used to model complex seasonal and trend patterns.

```
# Initialize our HoltWinters model, the es() function constructs a model and returns forecast and fitted v
# "AAM" denotes HoltWinters model, h is our forecast length, interval allows for prediction intervals
model_AAM <- es(covid_confirmed_ts, "AAM", h=50, interval=TRUE, silent = "none")
```

```
## The provided data is not ts object. Only non-seasonal models are available.
```




```
holtwinters_forecast_max <- max(model_AAM$forecast)
holtwinters_forecast_max
```

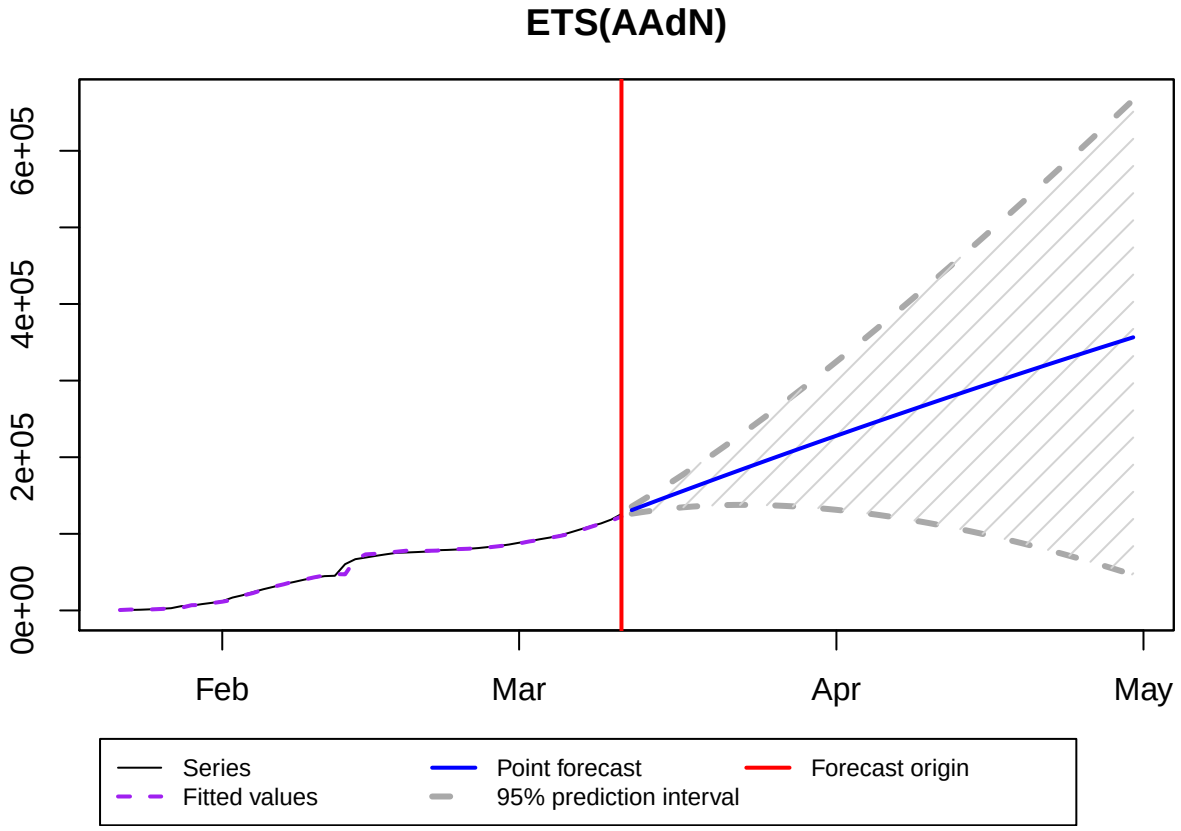
```
## [1] 380108.8
```

Based on above forecast, with our current trend the highest number of confirmed cases with me 3.80109×10^5 there is also an evident steady increase.

Using the same function, let us try different types of interval, loss function, and ETS model:

- The first letter stands for the type of the error term ("A" or "M"),
- The second (and sometimes the third as well) is for the trend ("N", "A", "Ad", "M" or "Md"),
- The last one is for the type of seasonality ("N", "A" or "M")

```
model_AAM <- es(covid_confirmed_ts, "AAdN", h=50, interval="l",
                silent = "none")
```




```
max(model_AAM$forecast)
```

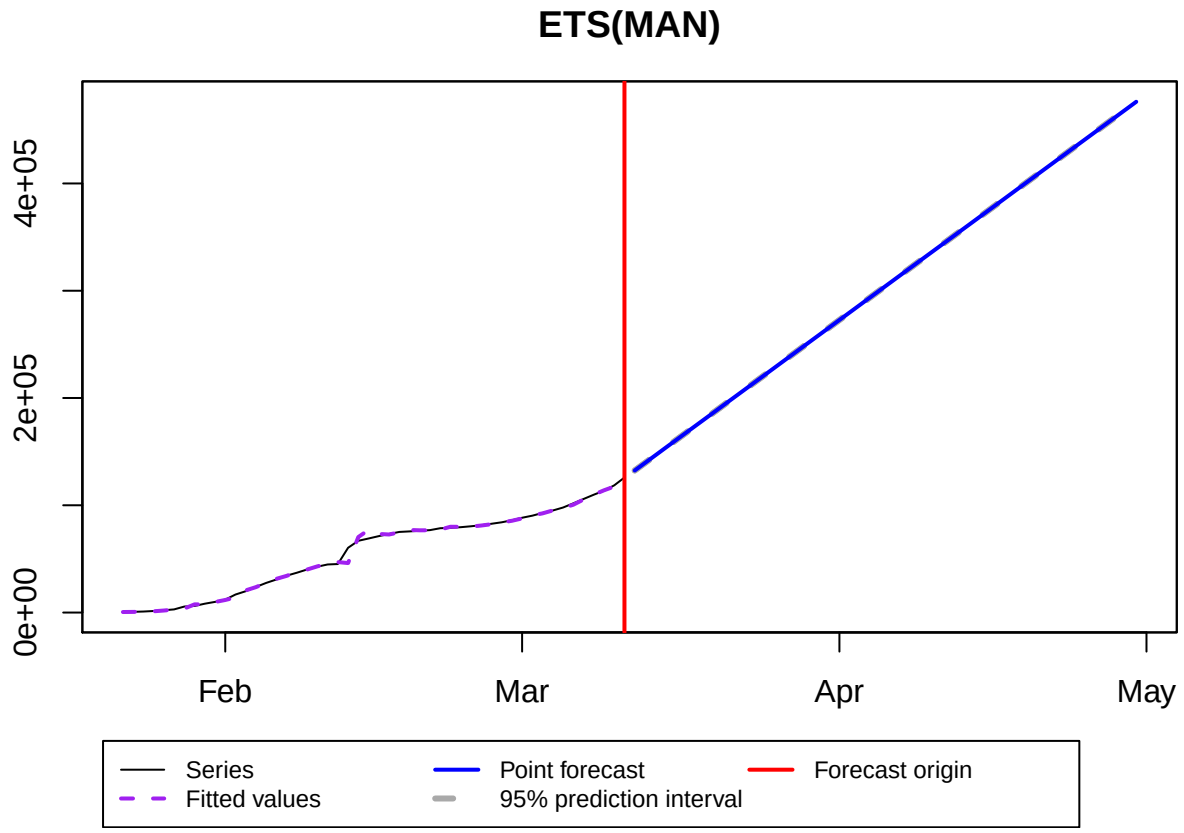
```
## [1] 356503.2
```

The "AAdN" model has a decreased forecast

```
model_AAM <- es(covid_confirmed_ts, "MAdM", h=50, interval="np",
                silent = "none")
```

```
## The provided data is not ts object. Only non-seasonal models are available.

## Warning: The parameter phi is equal to one, so we reverted to the non-damped
## version of the model
```




```
max(model_AAM$forecast)
```

```
## [1] 476138.8
```

The "MAdM" which takes into account Multiplicative error term and seasonality yields the highest forecast of cases.

2.2.4 ARIMA

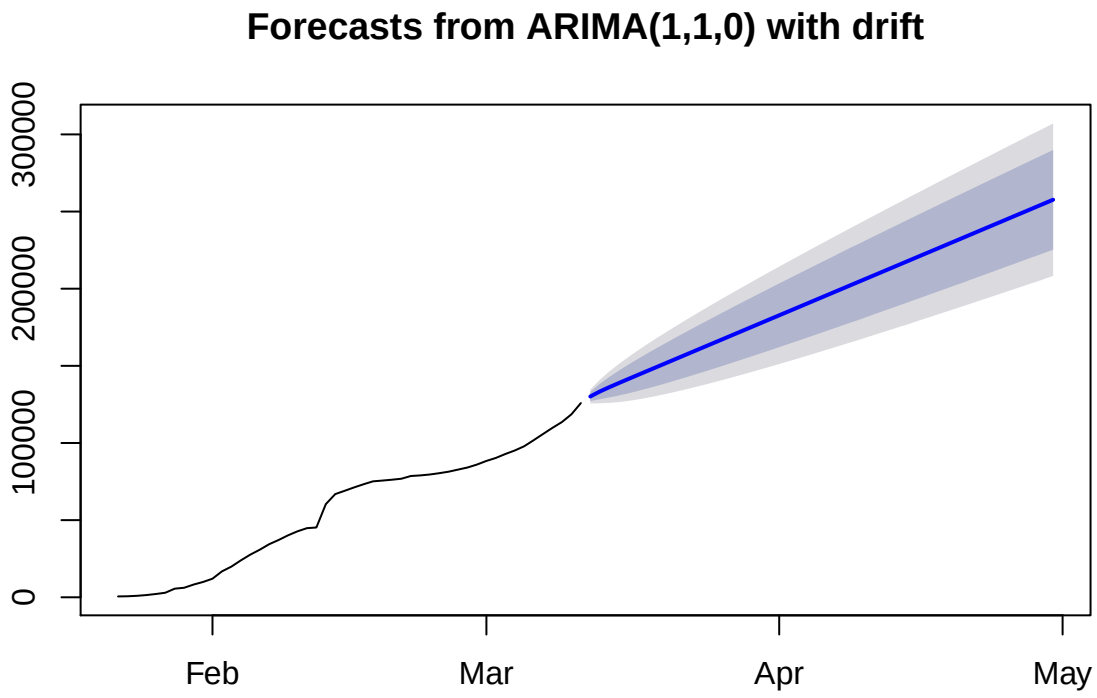
Auto Regressive Integrated Moving Average models factor in lags and lagged errors. It has 3 terms: p (autoregression), d(moving average), q(difference) The model also describes autocorrelations of the observations.

```
# Initialize our ARIMA model. It is straightforward as we only have to plug in our time series
# The auto.arima function return the best model based on AIC/BIC value, we can enter various parameters in
model_arima <- auto.arima(covid_confirmed_ts)

# Forecast next 50 days for the ARIMA model
forecast_arima <- forecast::forecast(model_arima, h=50)
arima_forecast_max <- max(forecast_arima$mean)
arima_forecast_max
```

```
## [1] 257657.5
```

```
plot(forecast_arima, xaxt ="n")
Axis(inds, side = 1)
```



Using the ARIMA model this is the forecast 2.57657×10^5 of cases for the next 50 days

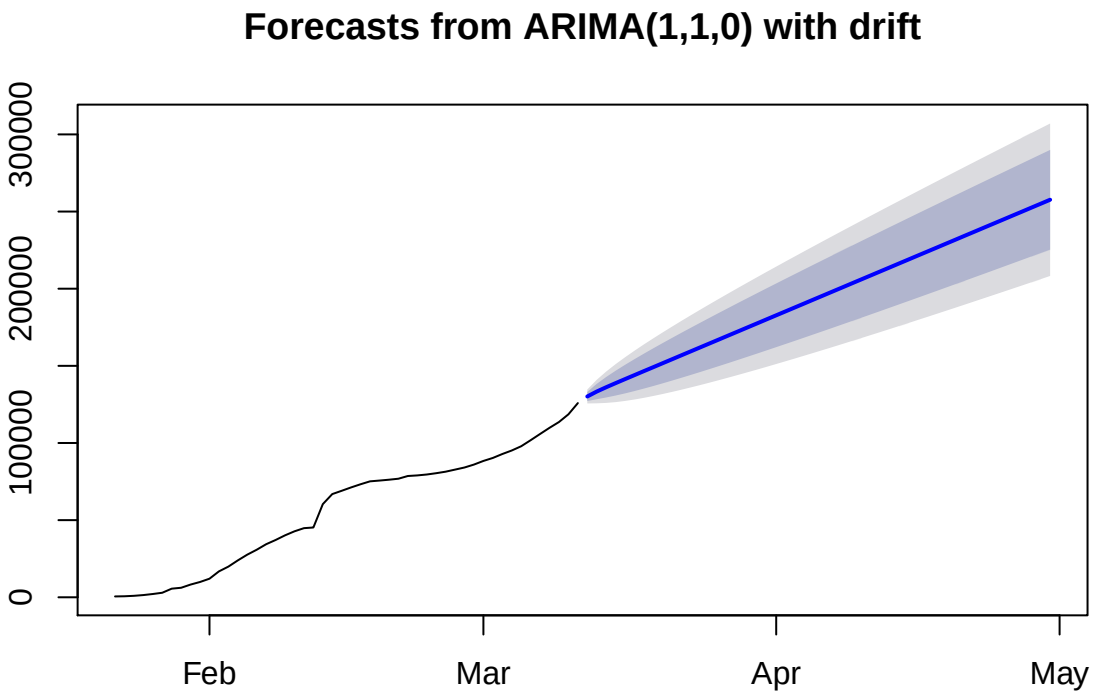
Let us try another model with different values

```
model_arima <- auto.arima(covid_confirmed_ts, max.p = 5,
  max.q = 5, max.P = 2, max.Q = 2,
  max.order = 5, max.d = 2, max.D = 1,
  start.p = 2, start.q = 2, start.P = 1, start.Q = 1,
  stationary = FALSE,
  seasonal = FALSE)

forecast_arima <- forecast::forecast(model_arima, h=50)
max(forecast_arima$mean)
```

```
## [1] 257657.5
```

```
plot(forecast_arima, xaxt ="n")
Axis(inds, side = 1)
```



Based on the forecast, it seems like we got the same model even though we had differing values for parameters, the initial auto arima model did its job to return the best model.

3 Results

Now let us tabulate our models' performance and maximum forecast for Confirmed cases.

```
# Look at the Naive model performance and maximum forecast
summary(model_naive)
```

```
##
## Forecast method: Seasonal naive method
##
## Model Information:
## Call: snaive(y = covid_confirmed_ts, h = 50)
##
## Residual sd: 2431.0251
##
## Error measures:
##                     ME    RMSE      MAE      MPE     MAPE MASE      ACF1
## Training set 2557.347 3511.31 2557.347 9.697016 9.697016    1 0.3318278
##
## Forecasts:
##       Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## 18333         125865 121365.08 130364.9 118982.96 132747.0
## 18334         125865 119501.15 132228.9 116132.32 135597.7
## 18335         125865 118070.90 133659.1 113944.96 137785.0
```

```
## 18336         125865 116865.15 134864.8 112100.92 139629.1
## 18337         125865 115802.86 135927.1 110476.29 141253.7
## 18338         125865 114842.48 136887.5 109007.51 142722.5
## 18339         125865 113959.32 137770.7 107656.83 144073.2
## 18340         125865 113137.29 138592.7 106399.65 145330.4
## 18341         125865 112365.23 139364.8 105218.88 146511.1
## 18342         125865 111634.99 140095.0 104102.08 147627.9
## 18343         125865 110940.44 140789.6 103039.85 148690.1
## 18344         125865 110276.80 141453.2 102024.91 149705.1
## 18345         125865 109640.29 142089.7 101051.45 150678.6
## 18346         125865 109027.82 142702.2 100114.76 151615.2
## 18347         125865 108436.87 143293.1  99210.97 152519.0
## 18348         125865 107865.30 143864.7  98336.84 153393.2
## 18349         125865 107311.34 144418.7  97489.62 154240.4
## 18350         125865 106773.44 144956.6  96666.97 155063.0
## 18351         125865 106250.28 145479.7  95866.88 155863.1
## 18352         125865 105740.73 145989.3  95087.58 156642.4
## 18353         125865 105243.76 146486.2  94327.53 157402.5
## 18354         125865 104758.48 146971.5  93585.37 158144.6
## 18355         125865 104284.12 147445.9  92859.89 158870.1
## 18356         125865 103819.96 147910.0  92150.02 159580.0
## 18357         125865 103365.38 148364.6  91454.80 160275.2
## 18358         125865 102919.80 148810.2  90773.34 160956.7
## 18359         125865 102482.71 149247.3  90104.87 161625.1
## 18360         125865 102053.64 149676.4  89448.66 162281.3
## 18361         125865 101632.16 150097.8  88804.08 162925.9
## 18362         125865 101217.90 150512.1  88170.51 163559.5
## 18363         125865 100810.48 150919.5  87547.42 164182.6
## 18364         125865 100409.58 151320.4  86934.30 164795.7
## 18365         125865 100014.90 151715.1  86330.69 165399.3
## 18366         125865  99626.16 152103.8  85736.15 165993.8
## 18367         125865  99243.09 152486.9  85150.30 166579.7
## 18368         125865  98865.45 152864.5  84572.76 167157.2
## 18369         125865  98493.03 153237.0  84003.18 167726.8
## 18370         125865  98125.60 153604.4  83441.25 168288.7
## 18371         125865  97762.98 153967.0  82886.67 168843.3
## 18372         125865  97404.98 154325.0  82339.15 169390.8
## 18373         125865  97051.42 154678.6  81798.44 169931.6
## 18374         125865  96702.16 155027.8  81264.28 170465.7
## 18375         125865  96357.02 155373.0  80736.44 170993.6
## 18376         125865  96015.88 155714.1  80214.71 171515.3
## 18377         125865  95678.59 156051.4  79698.87 172031.1
## 18378         125865  95345.03 156385.0  79188.73 172541.3
## 18379         125865  95015.07 156714.9  78684.11 173045.9
## 18380         125865  94688.61 157041.4  78184.82 173545.2
## 18381         125865  94365.53 157364.5  77690.72 174039.3
## 18382         125865  94045.73 157684.3  77201.62 174528.4
```

```
# Residual sd: 2431.0251
# Error measures:
#                     ME    RMSE      MAE      MPE     MAPE MASE      ACF1
# Training set 2557.347 3511.31 2557.347 9.697016 9.697016    1 0.3318278

# Look at the SES model performance and maximum forecast
summary(model_ses)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = covid_confirmed_ts, h = 50, alpha = 0.75, lambda = "auto",
##
##  Call:
##      damped = FALSE, exponential = TRUE, beta = 0.75)
##
##   Box-Cox transformation: lambda= 0.6811
##
##   Smoothing parameters:
##     alpha = 0.75
##
##   Initial states:
##     l = 113.5379
##
##   sigma:  144.3666
##
##      AIC     AICc      BIC
## 694.7957 695.0510 698.6197
##
## Error measures:
##                     ME     RMSE      MAE      MPE     MAPE     MASE      ACF1
## Training set 3299.832 4296.649 3301.767 11.80902 12.15766 1.291091 0.5481646
##
## Forecasts:
##       Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## 18333         123630 115929.97 131486.2 111918.23 135707.0
## 18334         123630 114029.76 133474.2 109048.87 138781.8
## 18335         123630 112458.43 135133.3 106682.73 141353.5
## 18336         123630 111090.34 136589.3 104627.66 143614.2
## 18337         123630 109863.97 137903.5 102789.52 145658.0
## 18338         123630 108743.69 139111.6 101113.84 147539.4
## 18339         123630 107706.73 140236.5  99565.73 149293.3
## 18340         123630 106737.42 141293.7  98121.28 150943.7
## 18341         123630 105824.50 142294.5  96763.21 152507.7
## 18342         123630 104959.50 143247.4  95478.60 153998.4
## 18343         123630 104135.90 144158.9  94257.43 155425.6
## 18344         123630 103348.53 145034.2  93091.82 156797.4
## 18345         123630 102593.21 145877.5  91975.36 158120.1
## 18346         123630 101866.52 146692.2  90902.82 159398.9
## 18347         123630 101165.59 147481.1  89869.81 160638.3
## 18348         123630 100488.01 148246.6  88872.64 161841.9
## 18349         123630  99831.74 148990.9  87908.18 163012.9
## 18350         123630  99194.99 149715.6  86973.70 164154.0
## 18351         123630  98576.24 150422.4  86066.86 165267.4
## 18352         123630  97974.13 151112.6  85185.60 166355.4
## 18353         123630  97387.48 151787.3  84328.09 167419.7
## 18354         123630  96815.23 152447.6  83492.73 168462.0
## 18355         123630  96256.46 153094.4  82678.09 169483.5
## 18356         123630  95710.32 153728.7  81882.87 170485.8
## 18357         123630  95176.06 154351.0  81105.93 171469.8
```

```
## 18358         123630  94652.99 154962.2  80346.22 172436.6
## 18359         123630  94140.49 155562.8  79602.79 173387.3
## 18360         123630  93638.01 156153.4  78874.77 174322.6
## 18361         123630  93145.01 156734.6  78161.37 175243.4
## 18362         123630  92661.04 157306.7  77461.88 176150.3
## 18363         123630  92185.65 157870.3  76775.61 177044.1
## 18364         123630  91718.45 158425.7  76101.97 177925.3
## 18365         123630  91259.06 158973.3  75440.39 178794.5
## 18366         123630  90807.15 159513.4  74790.33 179652.3
## 18367         123630  90362.38 160046.4  74151.31 180499.1
## 18368         123630  89924.47 160572.5  73522.87 181335.4
## 18369         123630  89493.15 161092.1  72904.59 182161.7
## 18370         123630  89068.14 161605.4  72296.08 182978.2
## 18371         123630  88649.21 162112.6  71696.97 183785.4
## 18372         123630  88236.13 162614.0  71106.90 184583.6
## 18373         123630  87828.70 163109.7  70525.55 185373.2
## 18374         123630  87426.70 163600.0  69952.61 186154.5
## 18375         123630  87029.95 164085.1  69387.80 186927.7
## 18376         123630  86638.27 164565.1  68830.85 187693.1
## 18377         123630  86251.50 165040.2  68281.48 188451.0
## 18378         123630  85869.47 165510.6  67739.47 189201.6
## 18379         123630  85492.04 165976.4  67204.58 189945.2
## 18380         123630  85119.06 166437.8  66676.59 190681.9
## 18381         123630  84750.40 166894.9  66155.30 191412.1
## 18382         123630  84385.92 167347.8  65640.50 192135.8
```

```
#     AIC     AICc      BIC
# 674.1495 674.4048 677.9736
#
# Error measures:
#                     ME     RMSE      MAE     MPE     MAPE      MASE      ACF1
# Training set 2530.505 3499.967 2530.536 9.58068 9.586328 0.9895163 0.3540446

# Look at the HoltWinters model performance and maximum forecast
summary(model_AAM)
```

```
## Time elapsed: 0.96 seconds
## Model estimated: ETS(MAN)
## Persistence vector g:
## alpha  beta
## 0.831 0.831
## Initial values were optimised.
##
## Loss function type: MSE; Loss function value: 0.0096
## Error standard deviation: 0.1034
## Sample size: 50
## Number of estimated parameters: 4
## Number of degrees of freedom: 46
## Information criteria:
##      AIC     AICc      BIC     BICc
## 958.6877 960.0513 968.2478 970.9151
##
## 95% nonparametric prediction interval was constructed
```

```
# Loss function type: MSE; Loss function value: 5235844.5326
# Error standard deviation: 2385.61
# Information criteria:
#      AIC     AICc      BIC     BICc
# 923.4458 924.3347 931.0939 932.8326

# Look at the ARIMA model performance and maximum forecast
summary(model_arima)
```

```
## Series: covid_confirmed_ts
## ARIMA(1,1,0) with drift
##
## Coefficients:
##          ar1      drift
##       0.3606  2582.8334
## s.e.  0.1390   498.1569
##
## sigma^2 estimated as 5297239:  log likelihood=-447.9
## AIC=901.81   AICc=902.34   BIC=907.48
##
## Training set error measures:
##                    ME     RMSE      MAE       MPE     MAPE    MASE        ACF1
## Training set 21.2316 2231.458 1176.584 -13.61523 16.47069 0.46008 -0.01570852
```

```
# AIC=901.81   AICc=902.34   BIC=907.48
#
# Training set error measures:
#                    ME     RMSE      MAE       MPE     MAPE    MASE        ACF1
# Training set 21.2316 2231.458 1176.584 -13.61523 16.47069 0.46008 -0.01570852
```

3.1 Model performance

For evaluation, there is a combination of error metrics available: ME, RMSE, MAE, MAPE. Additionally we are introducted to AIC and BIC which are penalized-likelihood criteria to predict best subsets. In general the lower the AIC and BIC, the better the model performance.

If we compare our Naive and SES models, they have similar error metrics, and SES has better AIC/BIC values compared to our HoltWinters and ARIMA model.

On the other hand if we look at HoltWInter and ARIMA, ARIMA has better AIC/BIC values and it performed relatively better compared to our Naive and SES models.

3.2 Model forecast

Forecast model	Maximum forecasted cases
Naive	1.25865×10^5
Simple Exponential Smoothing (SES)	1.2579097×10^5
Holt-Winters	3.8010878×10^5
ARIMA	2.5765749×10^5

We can see that the highest forecast came from HoltWinters model, while the lowest came from SES.

4 Conclusion

In this short exercise we were able to explore the Coronavirus dataset and do a forecast on the number of cases for the next 50 days.

4.1 Summary

- The dplyr functions, including group_by, and summarize worked splendidly for the data set. We have explored various ways of data exploration with dplyr.
- We were able to practice a lot with timeseries data, i.e., 'zoo' time series, ts() function, as well as plotting these series.
- We were also able to display a global map using ggplot of Confirmed cases.
- We also observed different forecast behaviors using our four models. Some have linear predictions whereby others were more non-linear.
- We still have not explored trends, seasonality, and recurring patters since we only have 50-days worth of datapoints.

4.2 Recommendations

- A deeper look on the Province.State and Country.Region to customize the forecast models and parameters.
- Particular attention to recent events, lock-downs, and bans, and how they relate with the forecasting.
- Finally, humanity as a whole is ever-hopeful and optimistic in finding a vaccine. A model forecasting the decline in Confirmed cases as a product of developing a vaccine will be useful for future consideration.