TSModel

Yiling Yang

```
# Library Import
library(tidyverse)
library(forecast)
library(lubridate)
library(tseries)
library(kableExtra)
```

```
# Data Import
covid <- read_csv("C:/Users/Yiling Yang/Desktop/school_work/Project/COVIDforecast/data/data-yqfdF.csv")
covid$DATE_OF_INTEREST <-  as.Date(parse_date_time(covid$DATE_OF_INTEREST,
                            guess_formats(
                              as.character(covid$DATE_OF_INTEREST), c("mdy", "dmy", "dmY")
                            )))

covid <- covid %>% rename(Date = DATE_OF_INTEREST) %>% select(Date,Cases)
```

Introduction

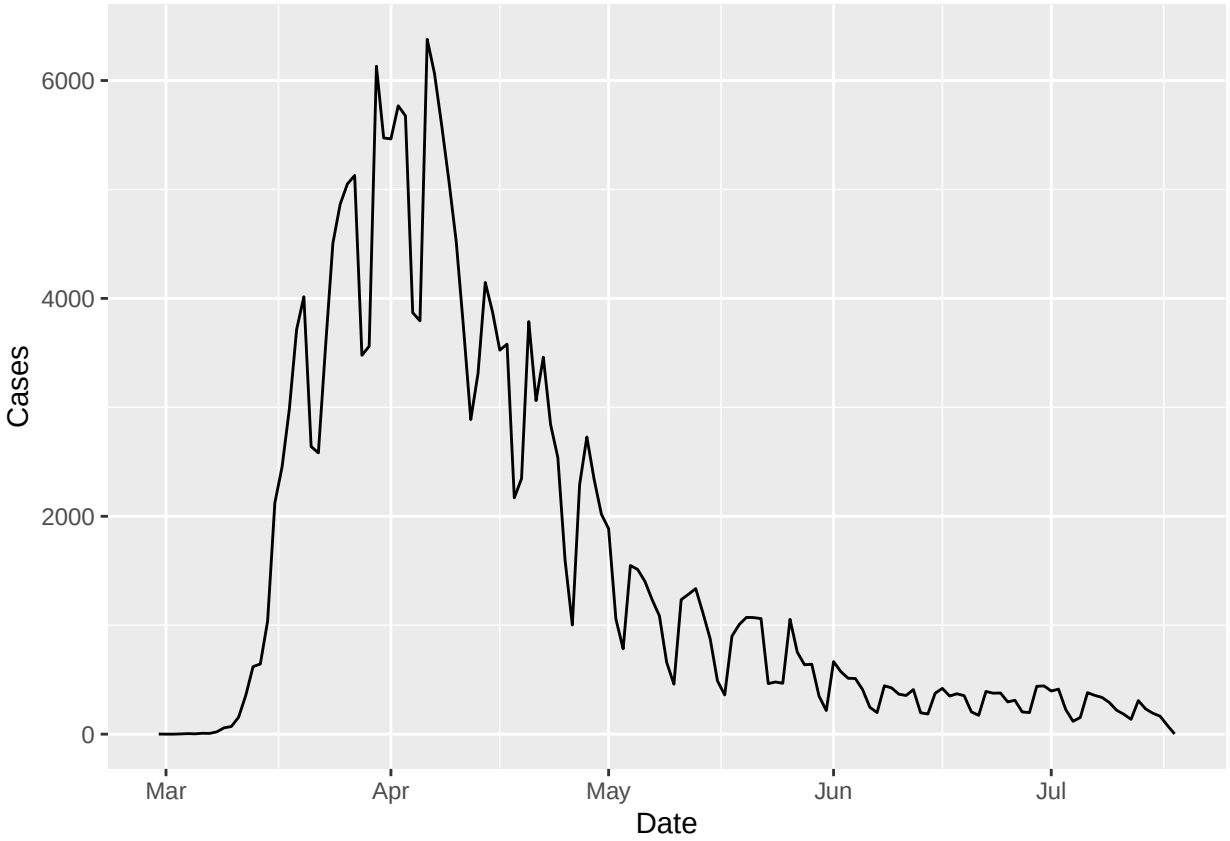
This is a personal project aimming to forecast the future daily confirmed cases of COVID-19 based on historical data from 3/15/2020 to 7/15/2020. The Project consists of two parts: 1) Forecast with Time Seris Model, ARMA and ARIMA, and 2) forecast with LSTM and 1-D CNN neural network. The LSTM neural network part will be done in Python. This markdown is particular for the first part.

Exploratory Data Analysis

To fit time series model, we usually need to make sure the stationary assumption held. I will perform 3 ways to check it: 1) Plot the time series, 2) Plot the ACF, and 3) Run ADF test.

1. Data at glance

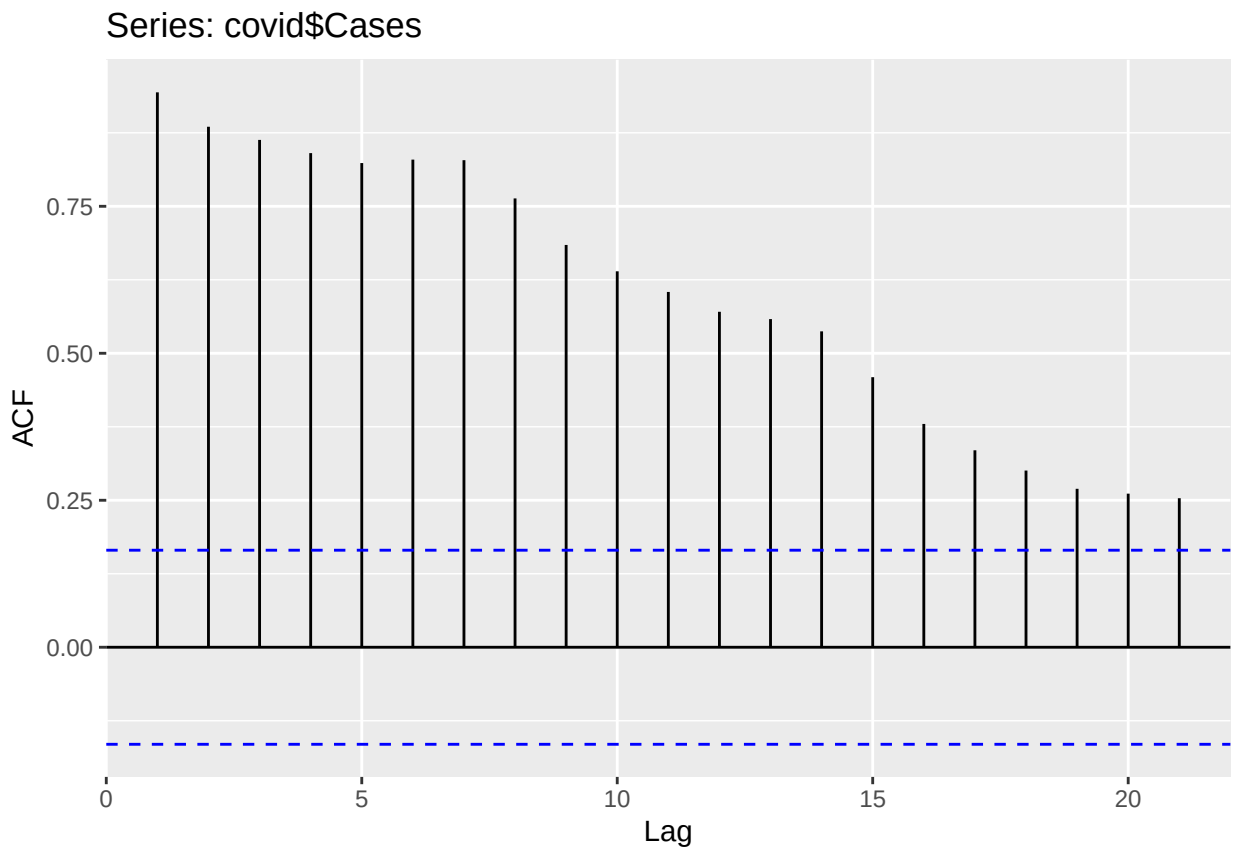
```
ggplot(covid,aes(x=Date,y=Cases)) + geom_line()
```



The data are not bounded very well, which might imply non-stationary.

2. ACF

```
ggAcf(covid$Cases,type="correlation")
```



The autocorrelation graph also shows a slow decreasing of the autocorrelation coefficient. This can also be a sign of non-stationary.

3. Augmented Dickey-Fuller Test

```
adf.test(covid$Cases)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  covid$Cases
## Dickey-Fuller = -2.5205, Lag order = 5, p-value = 0.3598
## alternative hypothesis: stationary
```

The ADF test output confirms with previous two chunks' output that the data is highly likely not stationary. We then might need transform it through differencing or detrending. Fortunately, the auto.arima function will help us achieve that.

Model Fitting

In this section, I will fit various time series model. For all of them, I will train them with the first 120 day as the training dataset, about 85 percent of the dataset with the use of AIC as the metric. And then, to compare with neural networks from the second part, I will record the mean square error of the validation set, which are the latest 20 observations.

ARMA

The Autoregressive–moving-average (ARMA) model combines the use of Moving Average (MA) and Autoregressive (AR) model. The equation can be written as:

$$X_t = c + \epsilon_t + \sum_{i=1}^{p} \beta_i X_t - i + \sum_{i=1}^{p} \theta_i \epsilon_t - i$$

where c is a constant.

```
arma.model <- auto.arima(covid$Cases[1:130],max.d = 0,allowdrift = T)
arma.model
```

```
## Series: covid$Cases[1:130]
## ARIMA(2,0,3) with non-zero mean
##
## Coefficients:
##          ar1      ar2      ma1      ma2     ma3       mean
##       1.9360  -0.9444  -1.0984  -0.4636  0.6716  1660.3410
## s.e.  0.0323   0.0325   0.0780   0.1108  0.0779   547.2428
##
## sigma^2 estimated as 247230:  log likelihood=-991
## AIC=1996   AICc=1996.92   BIC=2016.08
```

As the output shown, the chosen ARMA model is ARMA(2,3) with AIC 1852.83.

ARIMA

The difference between ARMA and ARIMA is that ARIMA allows for measure of how many nonseasonaldifferences are needed to achieve stationarity.

```
arima.model <- auto.arima(covid$Cases[1:120],allowdrift = T)
arima.model
```

```
## Series: covid$Cases[1:120]
## ARIMA(2,1,3)
##
## Coefficients:
##          ar1      ar2      ma1     ma2     ma3
##       1.1243  -0.4704  -1.1860  0.0201  0.4861
## s.e.  0.1584   0.1489   0.1726  0.2612  0.1233
##
## sigma^2 estimated as 275273:  log likelihood=-912.64
## AIC=1837.27   AICc=1838.02   BIC=1853.95
```

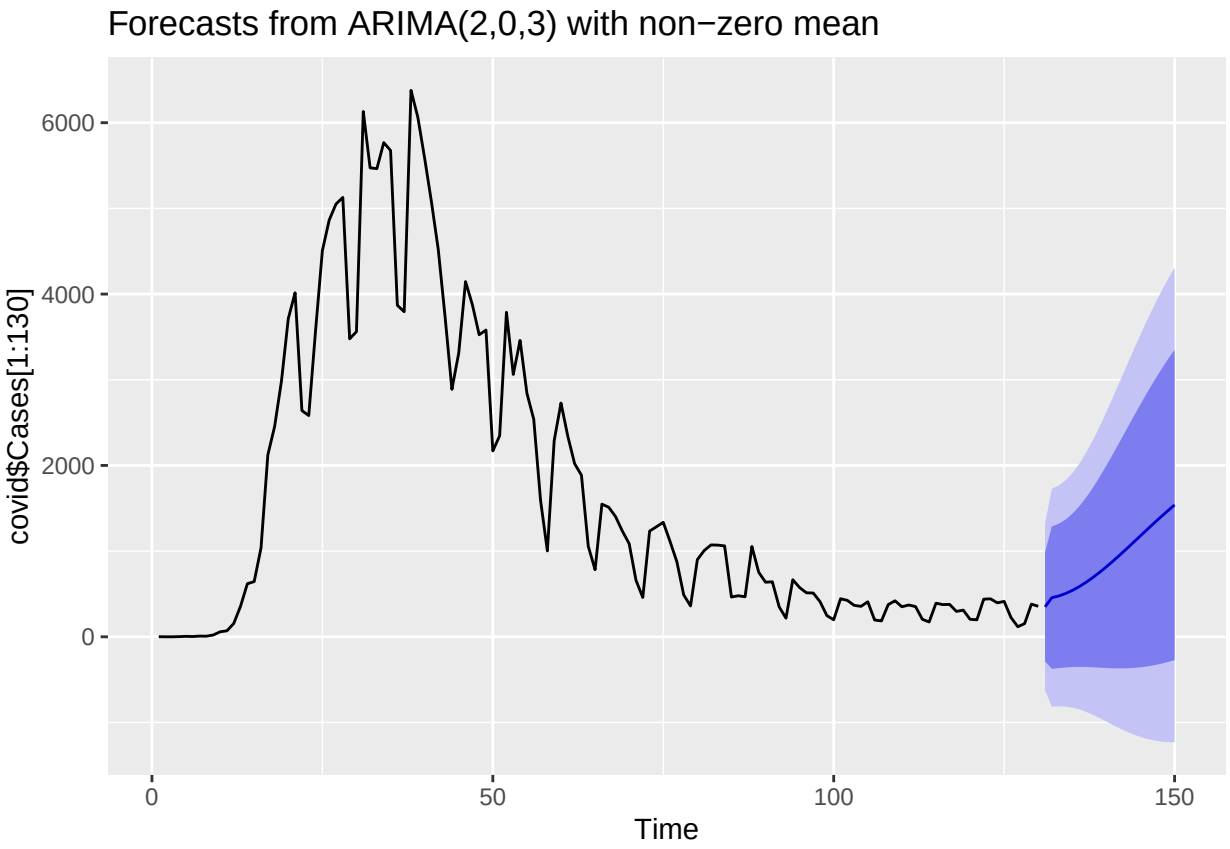
As the output shown, the ARIMA(2,1,3) was chosen with AIC of 1837.27.

Validation

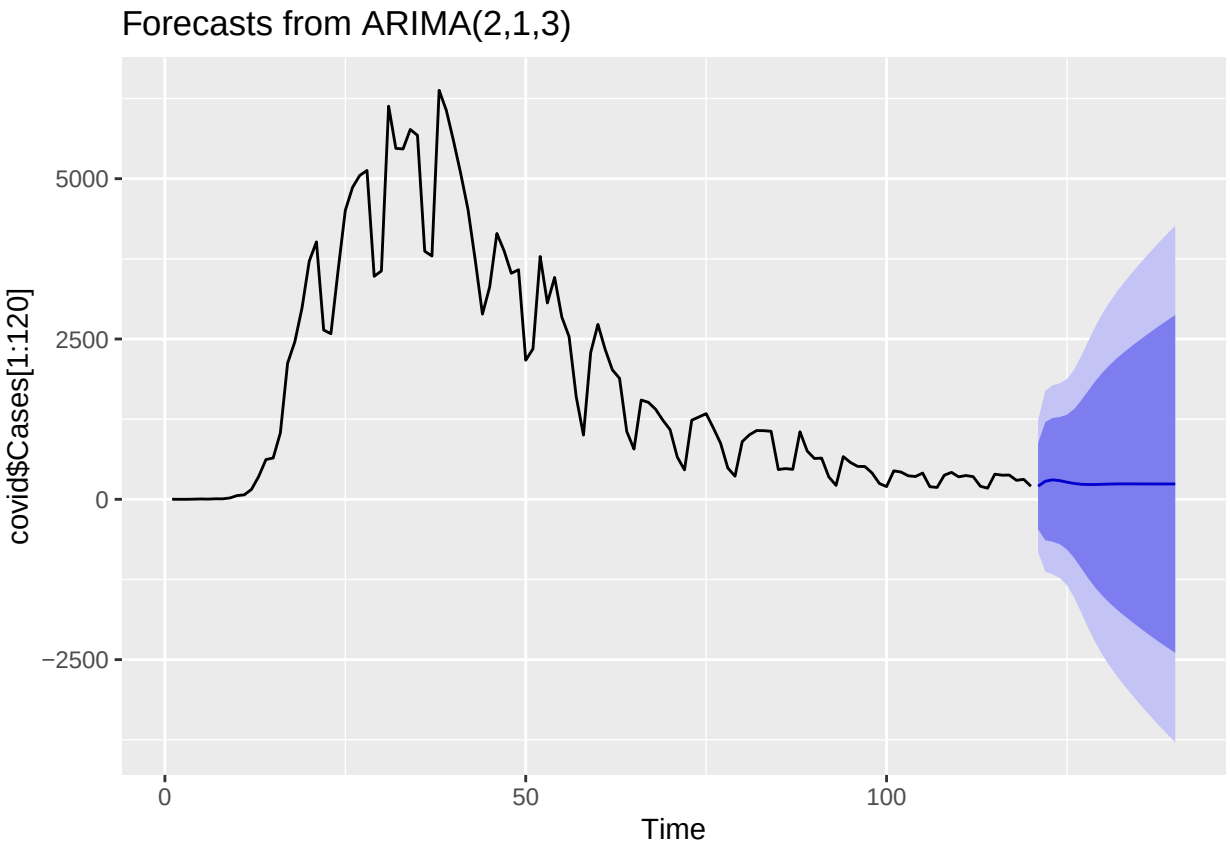
To compare with the neural networks, I will then do a forecast of next 41 days' data, and record their MSE. For demostration, I include the corresponding graphs for both two models.

```
arma.forecast <- forecast(arma.model,h=20)
arima.forecast <- forecast(arima.model,h=20)

autoplot(arma.forecast)
```




```
autoplot(arima.forecast)
```




```
kable(data.frame(
  model = c("ARMA","ARIMA"),
  MSE = c(mean((covid$Cases[121:140] - arma.forecast$mean)**2),
mean((covid$Cases[121:140] - arima.forecast$mean)**2))
))
```

model	MSE
ARMA	590503.075
ARIMA	9882.828

As the table shown above, the ARIMA(2,1,3) outperforms the ARMA(2,3) model. I will then step into the second part of this project, fitting LSTM CNN neural network.