Deliverables

Loading the dataset. Note: The dataset starts from Jan22 .

```
options(scipen = 999) # turn off scientific notation
# Read the data
covid_data <- read_excel("C:/Users/meghs/Downloads/2020_Covid_Data.xlsx")

#Covid Data of Particular State
ct_data <- covid_data$CT # ---------------------------> #replace with your state

#The dataset contains values starting from Jan22 hence 345
ct_data <- ct_data[1:345]
length(ct_data)
```

```
## [1] 345
```

```
ct_data
```

```
##   [1]      0      0      0      0      0      0      0      0      0      0
##  [11]      0      0      0      0      0      0      0      0      0      0
##  [21]      0      0      0      0      0      0      0      0      0      0
##  [31]      0      0      0      0      0      0      0      0      0      0
##  [41]      0      0      0      0      0      0      1      1      2      3
##  [51]      7      7      7     26     41     68     96    159    194    223
##  [61]    327    415    618    875   1012   1291   1524   1993   2571   3128
##  [71]   3557   3824   4915   5276   5675   6906   7781   8781   9784  10131
##  [81]  11510  12035  13381  13989  14755  15884  16809  17550  17962  19815
##  [91]  20360  22469  23100  23921  24582  25269  25997  26312  26767  27700
## [101]  28764  29287  29312  29973  30621  30995  31784  32411  32984  33554
## [111]  33765  34333  34855  35464  36085  36703  37419  38116  38430  39017
## [121]  39208  39640  40022  40468  40873  41303  41288  41559  41762  42022
## [131]  42201  42740  42979  43091  43239  43460  43818  43968  44092  44179
## [141]  44347  44461  44689  44994  45088  45235  45349  45429  45440  45557
## [151]  45715  45755  45782  45899  45913  45994  46059  46206  46303  46362
## [161]  46514  46572  46646  46717  46717  46717  46976  47033  47108  47209
## [171]  47287  47287  47287  47510  47530  47636  47750  47893  47893  47893
## [181]  48055  48096  48223  48232  48776  48776  48776  48983  49077  49540
## [191]  49670  49810  49810  49810  50062  50110  50225  50245  50320  50320
## [201]  50320  50567  50684  50706  50782  50897  50897  50897  51267  51255
## [211]  51314  51432  51519  51519  51519  52011  52040  52220  52350  52495
## [221]  52495  52495  52879  53006  53108  53209  53365  53365  53365  53365
## [231]  53782  53871  54093  54326  54326  54326  54895  55031  55166  55386
## [241]  55527  55527  55527  56024  56160  56315  56472  56587  56587  56587
## [251]  57147  57329  57550  57742  58297  58297  58297  59120  59241  59364
## [261]  59748  60038  60038  60038  61377  61697  61861  62028  62830  62830
## [271]  62830  64021  64455  64871  65373  66052  66052  66052  68099  68637
## [281]  69127  70446  71207  71207  71207  73858  74843  75373  77060  78125
## [291]  78125  78125  81463  82987  84741  85899  88645  88645  88645  93284
## [301]  94986  97028  99381 101469 101469 101469 106740 107280 109152 109152
## [311] 112581 112581 112581 117295 118754 121426 126177 127715 127715 127715
## [321] 135844 138258 140548 142979 146761 146761 146761 153992 155462 157781
## [331] 160102 162782 162782 162782 167377 168960 170705 172743 172743 172743
## [341] 172743 181200 181967 183663 185708
```

1. Step-by-Step fit different ARIMA (p,d,q) x (P, D, Q) for the confirmed cases. Can you discover a better model than auto.arima?

Convert data to time series object and plot it

```
# Create a time series object using a simple numeric sequence as the time index
ct_ts <- ts(ct_data, start= c(2020,1,22), frequency=345)# Starts from the date Jan 22
ct_ts
```

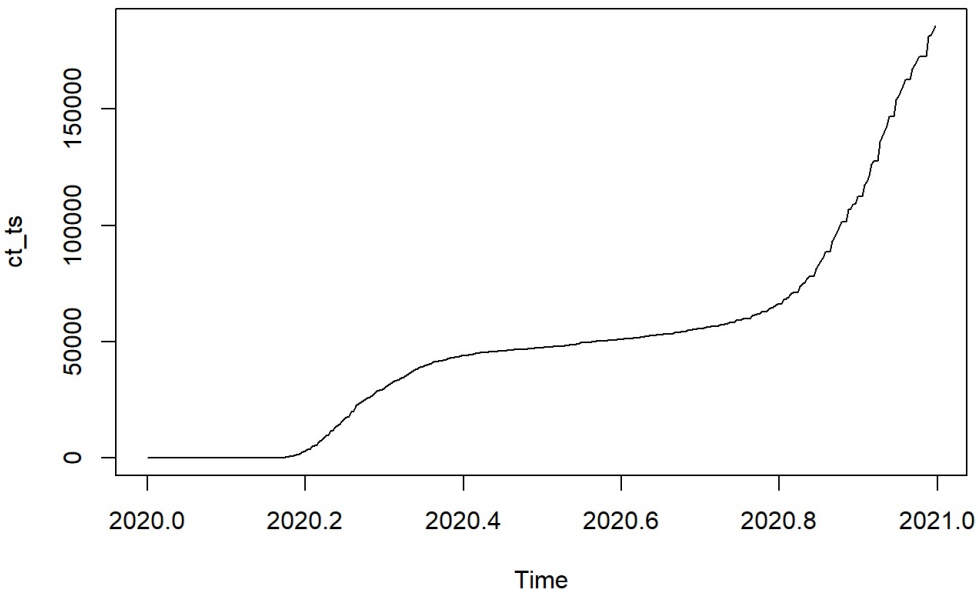
```
## Time Series:
## Start = c(2020, 1)
## End = c(2020, 345)
## Frequency = 345
##   [1]      0      0      0      0      0      0      0      0      0      0
##  [11]      0      0      0      0      0      0      0      0      0      0
##  [21]      0      0      0      0      0      0      0      0      0      0
##  [31]      0      0      0      0      0      0      0      0      0      0
##  [41]      0      0      0      0      0      0      1      1      2      3
##  [51]      7      7      7     26     41     68     96    159    194    223
##  [61]    327    415    618    875   1012   1291   1524   1993   2571   3128
##  [71]   3557   3824   4915   5276   5675   6906   7781   8781   9784  10131
##  [81]  11510  12035  13381  13989  14755  15884  16809  17550  17962  19815
##  [91]  20360  22469  23100  23921  24582  25269  25997  26312  26767  27700
## [101]  28764  29287  29312  29973  30621  30995  31784  32411  32984  33554
## [111]  33765  34333  34855  35464  36085  36703  37419  38116  38430  39017
## [121]  39208  39640  40022  40468  40873  41303  41288  41559  41762  42022
## [131]  42201  42740  42979  43091  43239  43460  43818  43968  44092  44179
## [141]  44347  44461  44689  44994  45088  45235  45349  45429  45440  45557
## [151]  45715  45755  45782  45899  45913  45994  46059  46206  46303  46362
## [161]  46514  46572  46646  46717  46717  46717  46976  47033  47108  47209
## [171]  47287  47287  47287  47510  47530  47636  47750  47893  47893  47893
## [181]  48055  48096  48223  48232  48776  48776  48776  48983  49077  49540
## [191]  49670  49810  49810  49810  50062  50110  50225  50245  50320  50320
## [201]  50320  50567  50684  50706  50782  50897  50897  50897  51267  51255
## [211]  51314  51432  51519  51519  51519  52011  52040  52220  52350  52495
## [221]  52495  52495  52879  53006  53108  53209  53365  53365  53365  53365
## [231]  53782  53871  54093  54326  54326  54326  54895  55031  55166  55386
## [241]  55527  55527  55527  56024  56160  56315  56472  56587  56587  56587
## [251]  57147  57329  57550  57742  58297  58297  58297  59120  59241  59364
## [261]  59748  60038  60038  60038  61377  61697  61861  62028  62830  62830
## [271]  62830  64021  64455  64871  65373  66052  66052  66052  68099  68637
## [281]  69127  70446  71207  71207  71207  73858  74843  75373  77060  78125
## [291]  78125  78125  81463  82987  84741  85899  88645  88645  88645  93284
## [301]  94986  97028  99381 101469 101469 101469 106740 107280 109152 109152
## [311] 112581 112581 112581 117295 118754 121426 126177 127715 127715 127715
## [321] 135844 138258 140548 142979 146761 146761 146761 153992 155462 157781
## [331] 160102 162782 162782 162782 167377 168960 170705 172743 172743 172743
## [341] 172743 181200 181967 183663 185708
```

```
## see what would happen if we don't use the whole historical data? e.g. starts from 2020.8, 2023.8,...?

ct_ts
```

```
## Time Series:
## Start = c(2020, 1)
## End = c(2020, 345)
## Frequency = 345
##   [1]      0      0      0      0      0      0      0      0      0      0
##  [11]      0      0      0      0      0      0      0      0      0      0
##  [21]      0      0      0      0      0      0      0      0      0      0
##  [31]      0      0      0      0      0      0      0      0      0      0
##  [41]      0      0      0      0      0      0      1      1      2      3
##  [51]      7      7      7     26     41     68     96    159    194    223
##  [61]    327    415    618    875   1012   1291   1524   1993   2571   3128
##  [71]   3557   3824   4915   5276   5675   6906   7781   8781   9784  10131
##  [81]  11510  12035  13381  13989  14755  15884  16809  17550  17962  19815
##  [91]  20360  22469  23100  23921  24582  25269  25997  26312  26767  27700
## [101]  28764  29287  29312  29973  30621  30995  31784  32411  32984  33554
## [111]  33765  34333  34855  35464  36085  36703  37419  38116  38430  39017
## [121]  39208  39640  40022  40468  40873  41303  41288  41559  41762  42022
## [131]  42201  42740  42979  43091  43239  43460  43818  43968  44092  44179
## [141]  44347  44461  44689  44994  45088  45235  45349  45429  45440  45557
## [151]  45715  45755  45782  45899  45913  45994  46059  46206  46303  46362
## [161]  46514  46572  46646  46717  46717  46717  46976  47033  47108  47209
## [171]  47287  47287  47287  47510  47530  47636  47750  47893  47893  47893
## [181]  48055  48096  48223  48232  48776  48776  48776  48983  49077  49540
## [191]  49670  49810  49810  49810  50062  50110  50225  50245  50320  50320
## [201]  50320  50567  50684  50706  50782  50897  50897  50897  51267  51255
## [211]  51314  51432  51519  51519  51519  52011  52040  52220  52350  52495
## [221]  52495  52495  52879  53006  53108  53209  53365  53365  53365  53365
## [231]  53782  53871  54093  54326  54326  54326  54895  55031  55166  55386
## [241]  55527  55527  55527  56024  56160  56315  56472  56587  56587  56587
## [251]  57147  57329  57550  57742  58297  58297  58297  59120  59241  59364
## [261]  59748  60038  60038  60038  61377  61697  61861  62028  62830  62830
## [271]  62830  64021  64455  64871  65373  66052  66052  66052  68099  68637
## [281]  69127  70446  71207  71207  71207  73858  74843  75373  77060  78125
## [291]  78125  78125  81463  82987  84741  85899  88645  88645  88645  93284
## [301]  94986  97028  99381 101469 101469 101469 106740 107280 109152 109152
## [311] 112581 112581 112581 117295 118754 121426 126177 127715 127715 127715
## [321] 135844 138258 140548 142979 146761 146761 146761 153992 155462 157781
## [331] 160102 162782 162782 162782 167377 168960 170705 172743 172743 172743
## [341] 172743 181200 181967 183663 185708
```

```
plot.ts(ct_ts) # plot time series first
```



ARIMA models Auto ARIMA model

```
#Fit the auto.arima model

auto_fit <- auto.arima(ct_ts)
summary(auto_fit)
```

```
## Series: ct_ts
## ARIMA(2,2,2)
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       0.3824  -0.2181  -1.6356  0.7939
## s.e.  0.0693   0.0621   0.0506  0.0472
##
## sigma^2 = 673855:  log likelihood = -2787.9
## AIC=5585.8   AICc=5585.98   BIC=5604.99
##
## Training set error measures:
##                     ME     RMSE     MAE      MPE     MAPE MASE        ACF1
## Training set 40.15631 813.7172 388.816 2.684823 3.321913  NaN -0.01221406
```

```
# Series: ct_ts
# ARIMA(2,2,2)
#
# Coefficients:
#          ar1      ar2      ma1     ma2
#       0.3824  -0.2181  -1.6356  0.7939
# s.e.  0.0693   0.0621   0.0506  0.0472
#
# sigma^2 = 673855:  log likelihood = -2787.9
# AIC=5585.8   AICc=5585.98   BIC=5604.99
#
# Training set error measures:
#                     ME     RMSE     MAE      MPE     MAPE MASE        ACF1
# Training set 40.15631 813.7172 388.816 2.684823 3.321913  NaN -0.01221406
```

Step-by-Step

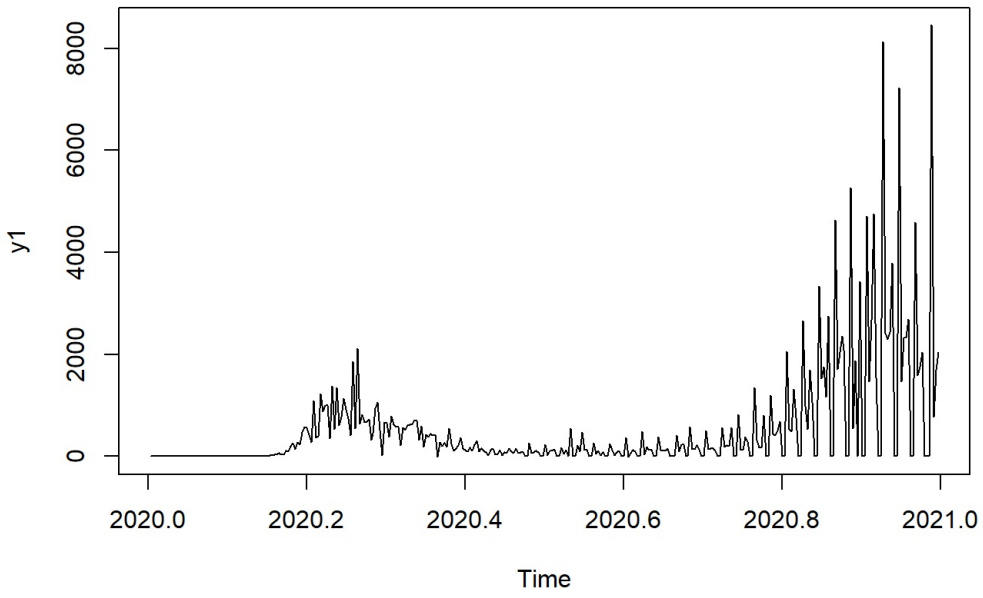
2. Select the best model using information criteria (AIC_c, BIC), and present out-of-sample forecasts with prediction intervals

```
## Step 1. Is the time series stationary?

# Use Augmented Dickey-Fuller Test to test stationary
adf.test(ct_ts)             # if p-value is large (> 0.10), then non stationary
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ct_ts
## Dickey-Fuller = -1.8445, Lag order = 7, p-value = 0.6424
## alternative hypothesis: stationary
```

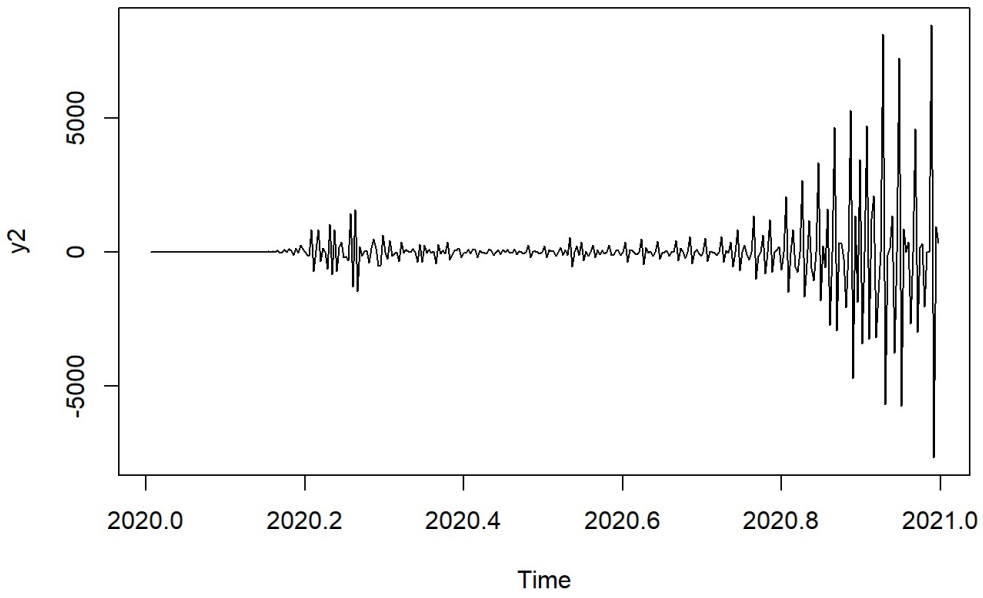
```
y1 <- diff(ct_ts, differences = 1)
plot.ts(y1)     # looks stationary visually
```




```
adf.test(y1)    # p-value is large
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  y1
## Dickey-Fuller = -1.0266, Lag order = 6, p-value = 0.9334
## alternative hypothesis: stationary
```

```
y2 <- diff(ct_ts, differences = 2)
plot.ts(y2)     # looks stationary visually
```

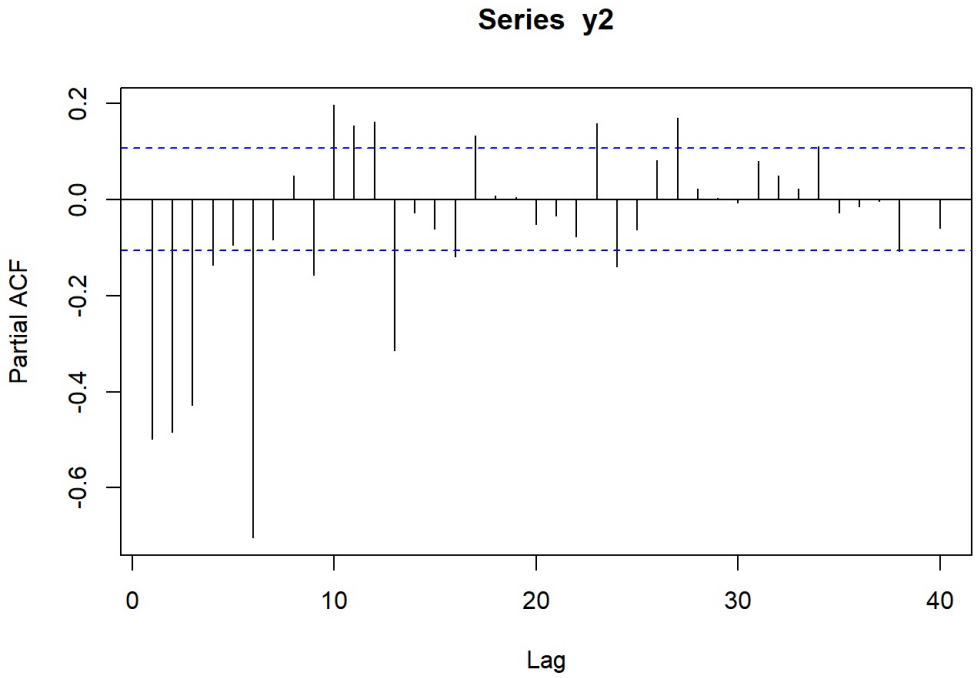



```
adf.test(y2)    # estimated p = 0.01 => small p-value (< 0.10) => so yd is stationary ==> fix d = 2 in ARIMA mode
ls to be fitted  -----> here my p value is 0.01
```

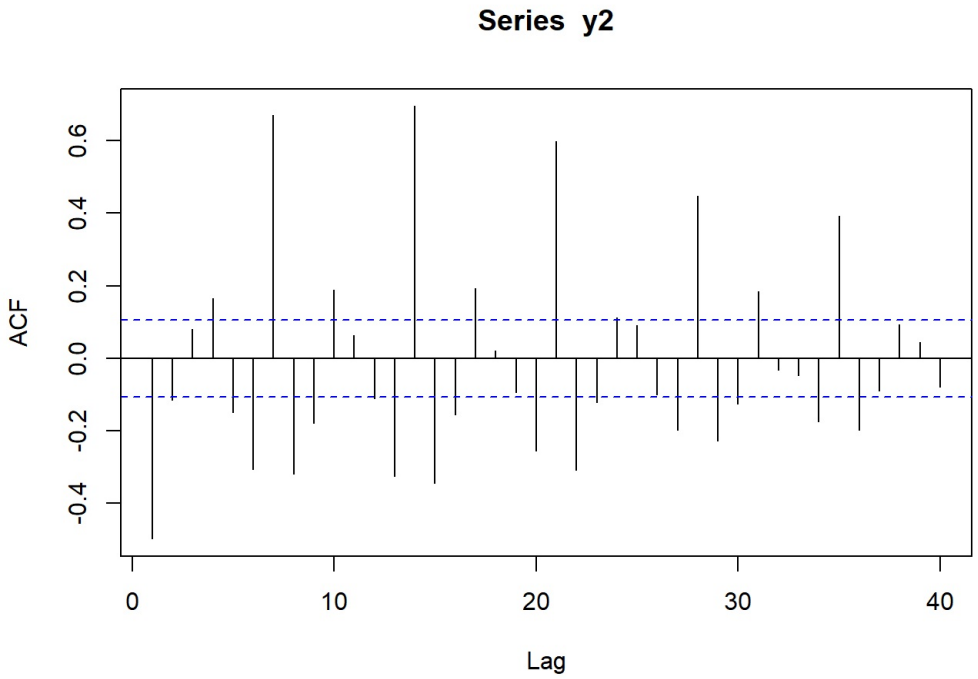
```
## Warning in adf.test(y2): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  y2
## Dickey-Fuller = -15.936, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

```
## Step 2. Decide AR(p) or MA(q) or both ARMA(p,q).
Pacf(y2, lag.max = 40) # PACF suggest p=4
```




```
Acf(y2, lag.max = 40)   #ACF suggest q=6
```



```r
# # nearby models Arima, d = 2;p->1,2,3 d-->2 q-->1,2,3
# Create a list to store models and their BIC values
models <- list()
bic_values <- numeric()

# Define the range of p, d, and q
p_values <- 1:3
d_value <- 2
q_values <- 1:3

# Iterate over all combinations of p, d, q
for (p in p_values) {
  for (q in q_values) {
    model <- Arima(ct_ts, order = c(p, d_value, q))
    model_name <- paste("ARIMA(", p, ",", d_value, ",", q, ")", sep = "")
    models[[model_name]] <- model
    bic_values[model_name] <- BIC(model)
  }
}

# Print the summaries and BIC of all models
for (model_name in names(models)) {
  cat("Model:", model_name, "\n")
  print(summary(models[[model_name]]))
  cat("BIC:", bic_values[model_name], "\n\n")
}
```

```
## Model: ARIMA(1,2,1)
## Series: ct_ts
## ARIMA(1,2,1)
##
## Coefficients:
##           ar1      ma1
##       -0.2108  -0.8942
## s.e.   0.0547   0.0187
##
## sigma^2 = 789555:  log likelihood = -2815.53
## AIC=5637.06   AICc=5637.13   BIC=5648.57
##
## Training set error measures:
##                     ME     RMSE      MAE      MPE     MAPE MASE        ACF1
## Training set 67.83507 883.4029 386.5834 2.521119 3.099501  NaN -0.06660272
## BIC: 5648.572
##
## Model: ARIMA(1,2,2)
## Series: ct_ts
## ARIMA(1,2,2)
##
## Coefficients:
##          ar1      ma1     ma2
##       0.3966  -1.7108  0.8433
## s.e.  0.0692   0.0423  0.0427
##
## sigma^2 = 695126:  log likelihood = -2793.71
## AIC=5595.42   AICc=5595.53   BIC=5610.77
##
## Training set error measures:
##                     ME     RMSE      MAE      MPE     MAPE MASE       ACF1
## Training set 35.73504 827.6787 386.1961 2.591942 3.279466  NaN 0.05447678
## BIC: 5610.766
##
## Model: ARIMA(1,2,3)
## Series: ct_ts
## ARIMA(1,2,3)
##
## Coefficients:
##          ar1      ma1     ma2     ma3
##       0.0258  -1.2796  0.1492  0.3297
## s.e.  0.1547   0.1458  0.2186  0.0993
##
## sigma^2 = 678429:  log likelihood = -2789.05
## AIC=5588.1   AICc=5588.27   BIC=5607.28
##
## Training set error measures:
##                     ME     RMSE     MAE      MPE     MAPE MASE         ACF1
## Training set 37.46222 816.4742 387.481 2.626935 3.271208  NaN -0.005104951
## BIC: 5607.284
```

```
##
## Model: ARIMA(2,2,1)
## Series: ct_ts
## ARIMA(2,2,1)
##
## Coefficients:
##           ar1      ar2      ma1
##       -0.3126  -0.3105  -0.8479
## s.e.   0.0544   0.0538   0.0255
##
## sigma^2 = 723382:  log likelihood = -2800.18
## AIC=5608.36   AICc=5608.47   BIC=5623.71
##
## Training set error measures:
##                     ME     RMSE      MAE      MPE     MAPE MASE        ACF1
## Training set 63.78378 844.3331 399.0209 2.765527 3.310667  NaN -0.04685579
## BIC: 5623.706
##
## Model: ARIMA(2,2,2)
## Series: ct_ts
## ARIMA(2,2,2)
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       0.3824  -0.2181  -1.6356  0.7939
## s.e.  0.0693   0.0621   0.0506  0.0472
##
## sigma^2 = 673855:  log likelihood = -2787.9
## AIC=5585.8   AICc=5585.98   BIC=5604.99
##
## Training set error measures:
##                     ME     RMSE     MAE      MPE     MAPE MASE        ACF1
## Training set 40.15631 813.7172 388.816 2.684823 3.321913  NaN -0.01221406
## BIC: 5604.989
##
## Model: ARIMA(2,2,3)
## Series: ct_ts
## ARIMA(2,2,3)
##
## Coefficients:
##           ar1     ar2      ma1      ma2     ma3
##       -0.4075  0.1993  -0.8494  -0.5973  0.7005
## s.e.   0.1119  0.0904   0.0934   0.1498  0.0847
##
## sigma^2 = 677708:  log likelihood = -2788.4
## AIC=5588.81   AICc=5589.06   BIC=5611.83
##
## Training set error measures:
##                     ME     RMSE      MAE      MPE    MAPE MASE       ACF1
## Training set 36.02659 814.8358 386.9043 2.611586 3.25787  NaN 0.01067431
## BIC: 5611.832
##
## Model: ARIMA(3,2,1)
## Series: ct_ts
## ARIMA(3,2,1)
##
## Coefficients:
##           ar1      ar2      ar3      ma1
##       -0.3871  -0.3830  -0.1531  -0.8115
## s.e.   0.0613   0.0602   0.0595   0.0341
##
## sigma^2 = 711788:  log likelihood = -2796.96
## AIC=5603.92   AICc=5604.1   BIC=5623.11
##
## Training set error measures:
##                    ME     RMSE      MAE      MPE     MAPE MASE         ACF1
## Training set 62.5553 836.3067 397.3354 2.859249 3.405724  NaN -0.004775868
## BIC: 5623.109
##
## Model: ARIMA(3,2,2)
## Series: ct_ts
## ARIMA(3,2,2)
##
## Coefficients:
##           ar1      ar2      ar3      ma1      ma2
##       -0.6446  -0.4667  -0.2402  -0.5503  -0.2194
## s.e.   0.1727   0.0814   0.0744   0.1703   0.1435
##
## sigma^2 = 711829:  log likelihood = -2796.46
```

```
## AIC=5604.93   AICc=5605.18   BIC=5627.95
##
## Training set error measures:
##                     ME     RMSE      MAE      MPE     MAPE MASE        ACF1
## Training set 62.86999 835.0964 395.6497 2.840162 3.384065  NaN -0.00453057
## BIC: 5627.952
##
## Model: ARIMA(3,2,3)
## Series: ct_ts
## ARIMA(3,2,3)
##
## Coefficients:
##           ar1     ar2      ar3      ma1      ma2     ma3
##       -0.5967  0.1589  -0.1986  -0.6387  -0.8431  0.7956
## s.e.   0.0700  0.0940   0.0627   0.0533   0.0327  0.0490
##
## sigma^2 = 667485:  log likelihood = -2786.15
## AIC=5586.3   AICc=5586.63   BIC=5613.16
##
## Training set error measures:
##                     ME     RMSE      MAE      MPE     MAPE MASE        ACF1
## Training set 39.72626 807.4693 390.4017 2.661136 3.300773  NaN -0.01373434
## BIC: 5613.16
```

```
# Identify the model with the lowest BIC
best_model_name <- names(which.min(bic_values))
best_model <- models[[best_model_name]]
cat("Model with the lowest BIC:\n")
```

```
## Model with the lowest BIC:
```

```
cat(best_model_name, "\n")
```

```
## ARIMA(2,2,2)
```

```
print(summary(best_model))
```

```
## Series: ct_ts
## ARIMA(2,2,2)
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       0.3824  -0.2181  -1.6356  0.7939
## s.e.  0.0693   0.0621   0.0506  0.0472
##
## sigma^2 = 673855:  log likelihood = -2787.9
## AIC=5585.8   AICc=5585.98   BIC=5604.99
##
## Training set error measures:
##                     ME     RMSE     MAE      MPE     MAPE MASE        ACF1
## Training set 40.15631 813.7172 388.816 2.684823 3.321913  NaN -0.01221406
```

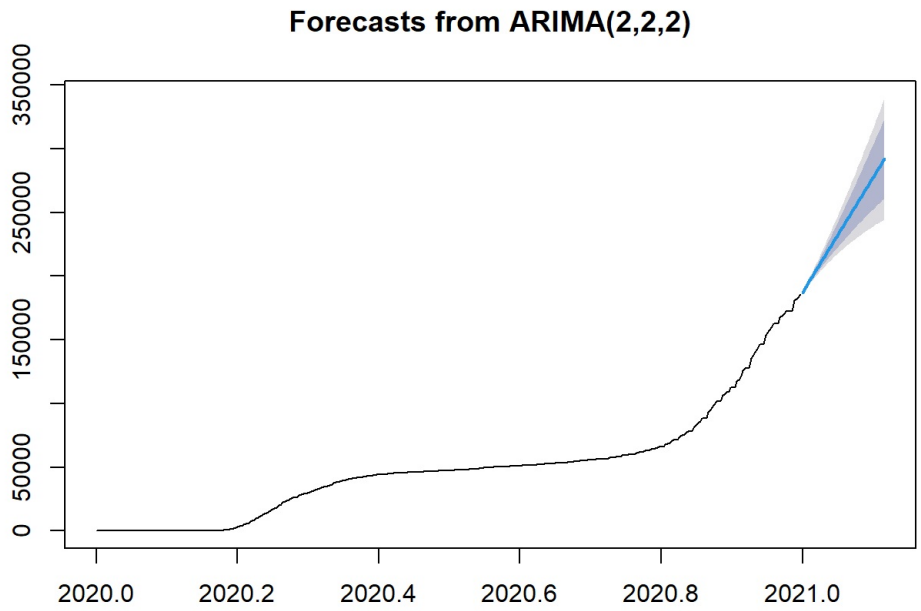
```
# predict best_model
y_hat <- predict(best_model)
y_hat
```

```
## $pred
## Time Series:
## Start = c(2021, 1)
## End = c(2021, 1)
## Frequency = 345
## [1] 187207.9
##
## $se
## Time Series:
## Start = c(2021, 1)
## End = c(2021, 1)
## Frequency = 345
## [1] 820.8867
```

3. Discover a set of good models in the neighborhood of the

best model. Then present the “Consensus Forecast” for COVID cases over the next week (from Sunday to Saturday after the submission date).

```
# Forecasting for a specified period (e.g., 41 days from the end of the series)
forecasted_values <- forecast(best_model, h=41)
# Plot the forecast
plot(forecasted_values)
```




```
print(forecasted_values)
```

```
##           Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2021.0000       187207.9 186155.9 188259.9 185599.0 188816.8
## 2021.0029       189528.0 188215.0 190840.9 187520.0 191536.0
## 2021.0058       192280.5 190828.2 193732.8 190059.4 194501.6
## 2021.0087       195019.5 193369.9 196669.1 192496.7 197542.3
## 2021.0116       197659.0 195705.4 199612.6 194671.3 200646.7
## 2021.0145       200263.4 197928.5 202598.3 196692.4 203834.4
## 2021.0174       202876.1 200113.5 205638.7 198651.1 207101.1
## 2021.0203       205499.6 202272.5 208726.7 200564.2 210435.0
## 2021.0232       208125.4 204398.5 211852.4 202425.5 213825.3
## 2021.0261       210749.8 206489.3 215010.3 204233.9 217265.7
## 2021.0290       213373.1 208548.2 218198.0 205994.0 220752.2
## 2021.0319       215996.3 210578.7 221413.9 207710.8 224281.8
## 2021.0348       218619.7 212583.0 224656.4 209387.4 227852.1
## 2021.0377       221243.2 214562.4 227924.0 211025.8 231460.6
## 2021.0406       223866.7 216518.0 231215.5 212627.8 235105.7
## 2021.0435       226490.2 218450.6 234529.8 214194.7 238785.8
## 2021.0464       229113.7 220361.2 237866.1 215728.0 242499.4
## 2021.0493       231737.2 222250.7 241223.6 217228.8 246245.5
## 2021.0522       234360.6 224119.6 244601.7 218698.3 250022.9
## 2021.0551       236984.1 225968.6 247999.6 220137.4 253830.8
## 2021.0580       239607.6 227798.3 251416.9 221546.8 257668.4
## 2021.0609       242231.1 229609.1 254853.0 222927.4 261534.7
## 2021.0638       244854.5 231401.5 258307.5 224279.9 265429.1
## 2021.0667       247478.0 233176.0 261780.0 225605.0 269351.0
## 2021.0696       250101.5 234932.9 265270.0 226903.2 273299.8
## 2021.0725       252725.0 236672.7 268777.2 228175.1 277274.8
## 2021.0754       255348.4 238395.6 272301.2 229421.3 281275.5
## 2021.0783       257971.9 240102.0 275841.8 230642.3 285301.5
## 2021.0812       260595.4 241792.2 279398.6 231838.4 289352.4
## 2021.0841       263218.8 243466.5 282971.2 233010.2 293427.5
## 2021.0870       265842.3 245125.1 286559.5 234158.1 297526.6
## 2021.0899       268465.8 246768.3 290163.3 235282.4 301649.2
## 2021.0928       271089.3 248396.4 293782.1 236383.6 305795.0
## 2021.0957       273712.7 250009.6 297415.9 237462.0 309963.5
## 2021.0986       276336.2 251608.1 301064.3 238517.9 314154.6
## 2021.1014       278959.7 253192.2 304727.2 239551.6 318367.7
## 2021.1043       281583.2 254761.9 308404.4 240563.6 322602.7
## 2021.1072       284206.6 256317.6 312095.7 241554.0 326859.3
## 2021.1101       286830.1 257859.4 315800.9 242523.2 331137.0
## 2021.1130       289453.6 259387.5 319519.7 243471.4 335435.8
## 2021.1159       292077.1 260902.0 323252.1 244398.9 339755.2
```

4. Your HW score will be based on MAPE (Mean Absolute Percentage Errors) based on actual cases and your forecasts. The smaller the MAPE, the better the model.

```
# Define the range of p, d, and q
p_values <- 1:3
d_value <- 2
q_values <- 1:3

# Store models
models <- list()

# Iterate over all combinations of p, d, q
for (p in p_values) {
  for (q in q_values) {
    model <- Arima(ct_ts, order = c(p, d_value, q))
    model_name <- paste("ARIMA(", p, ",", d_value, ",", q, ")", sep = "")
    models[[model_name]] <- model
  }
}

# Print the summary of all models
for (model_name in names(models)) {
  cat("Model:", model_name, "\n")
  print(summary(models[[model_name]]))
  cat("\n")
}
```

```
## Model: ARIMA(1,2,1)
## Series: ct_ts
```

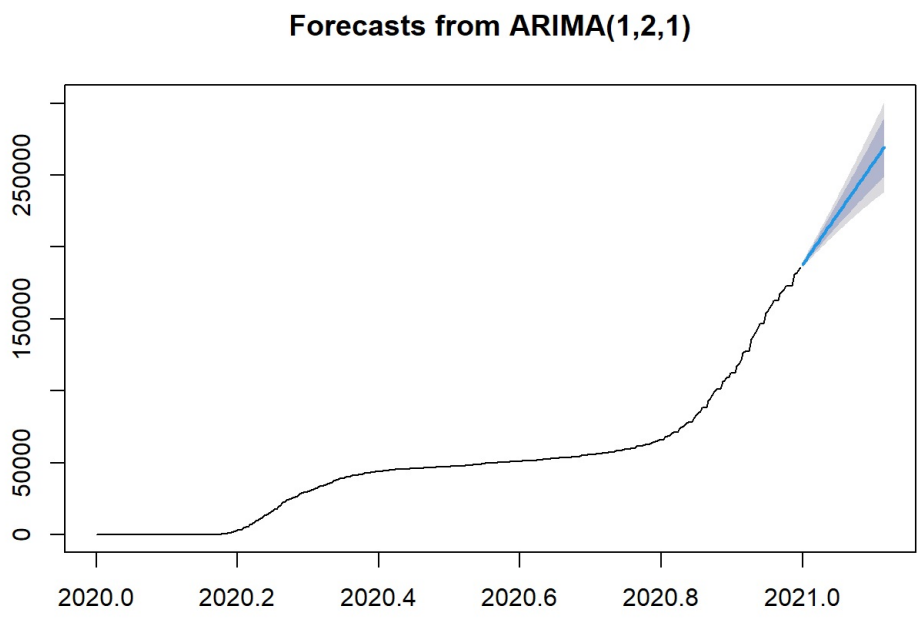
```
## ARIMA(1,2,1)
##
## Coefficients:
##          ar1      ma1
##       -0.2108  -0.8942
## s.e.   0.0547   0.0187
##
## sigma^2 = 789555:  log likelihood = -2815.53
## AIC=5637.06   AICc=5637.13   BIC=5648.57
##
## Training set error measures:
##                    ME     RMSE      MAE      MPE     MAPE MASE        ACF1
## Training set 67.83507 883.4029 386.5834 2.521119 3.099501  NaN -0.06660272
##
## Model: ARIMA(1,2,2)
## Series: ct_ts
## ARIMA(1,2,2)
##
## Coefficients:
##          ar1      ma1     ma2
##       0.3966  -1.7108  0.8433
## s.e.  0.0692   0.0423  0.0427
##
## sigma^2 = 695126:  log likelihood = -2793.71
## AIC=5595.42   AICc=5595.53   BIC=5610.77
##
## Training set error measures:
##                    ME     RMSE      MAE      MPE     MAPE MASE       ACF1
## Training set 35.73504 827.6787 386.1961 2.591942 3.279466  NaN 0.05447678
##
## Model: ARIMA(1,2,3)
## Series: ct_ts
## ARIMA(1,2,3)
##
## Coefficients:
##          ar1      ma1     ma2     ma3
##       0.0258  -1.2796  0.1492  0.3297
## s.e.  0.1547   0.1458  0.2186  0.0993
##
## sigma^2 = 678429:  log likelihood = -2789.05
## AIC=5588.1   AICc=5588.27   BIC=5607.28
##
## Training set error measures:
##                    ME     RMSE     MAE      MPE     MAPE MASE         ACF1
## Training set 37.46222 816.4742 387.481 2.626935 3.271208  NaN -0.005104951
##
## Model: ARIMA(2,2,1)
## Series: ct_ts
## ARIMA(2,2,1)
##
## Coefficients:
##          ar1      ar2      ma1
##       -0.3126  -0.3105  -0.8479
## s.e.   0.0544   0.0538   0.0255
##
## sigma^2 = 723382:  log likelihood = -2800.18
## AIC=5608.36   AICc=5608.47   BIC=5623.71
##
## Training set error measures:
##                    ME     RMSE      MAE      MPE     MAPE MASE        ACF1
## Training set 63.78378 844.3331 399.0209 2.765527 3.310667  NaN -0.04685579
##
## Model: ARIMA(2,2,2)
## Series: ct_ts
## ARIMA(2,2,2)
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       0.3824  -0.2181  -1.6356  0.7939
## s.e.  0.0693   0.0621   0.0506  0.0472
##
## sigma^2 = 673855:  log likelihood = -2787.9
## AIC=5585.8   AICc=5585.98   BIC=5604.99
##
## Training set error measures:
##                    ME     RMSE     MAE      MPE     MAPE MASE        ACF1
## Training set 40.15631 813.7172 388.816 2.684823 3.321913  NaN -0.01221406
##
## Model: ARIMA(2,2,3)
```

```
## Series: ct_ts
## ARIMA(2,2,3)
##
## Coefficients:
##           ar1     ar2      ma1      ma2     ma3
##       -0.4075  0.1993  -0.8494  -0.5973  0.7005
## s.e.   0.1119  0.0904   0.0934   0.1498  0.0847
##
## sigma^2 = 677708:  log likelihood = -2788.4
## AIC=5588.81   AICc=5589.06   BIC=5611.83
##
## Training set error measures:
##                    ME     RMSE      MAE      MPE    MAPE MASE       ACF1
## Training set 36.02659 814.8358 386.9043 2.611586 3.25787  NaN 0.01067431
##
## Model: ARIMA(3,2,1)
## Series: ct_ts
## ARIMA(3,2,1)
##
## Coefficients:
##           ar1      ar2      ar3      ma1
##       -0.3871  -0.3830  -0.1531  -0.8115
## s.e.   0.0613   0.0602   0.0595   0.0341
##
## sigma^2 = 711788:  log likelihood = -2796.96
## AIC=5603.92   AICc=5604.1   BIC=5623.11
##
## Training set error measures:
##                   ME     RMSE      MAE      MPE     MAPE MASE         ACF1
## Training set 62.5553 836.3067 397.3354 2.859249 3.405724  NaN -0.004775868
##
## Model: ARIMA(3,2,2)
## Series: ct_ts
## ARIMA(3,2,2)
##
## Coefficients:
##           ar1      ar2      ar3      ma1      ma2
##       -0.6446  -0.4667  -0.2402  -0.5503  -0.2194
## s.e.   0.1727   0.0814   0.0744   0.1703   0.1435
##
## sigma^2 = 711829:  log likelihood = -2796.46
## AIC=5604.93   AICc=5605.18   BIC=5627.95
##
## Training set error measures:
##                    ME     RMSE      MAE      MPE     MAPE MASE        ACF1
## Training set 62.86999 835.0964 395.6497 2.840162 3.384065  NaN -0.00453057
##
## Model: ARIMA(3,2,3)
## Series: ct_ts
## ARIMA(3,2,3)
##
## Coefficients:
##           ar1     ar2      ar3      ma1      ma2     ma3
##       -0.5967  0.1589  -0.1986  -0.6387  -0.8431  0.7956
## s.e.   0.0700  0.0940   0.0627   0.0533   0.0327  0.0490
##
## sigma^2 = 667485:  log likelihood = -2786.15
## AIC=5586.3   AICc=5586.63   BIC=5613.16
##
## Training set error measures:
##                    ME     RMSE      MAE      MPE     MAPE MASE        ACF1
## Training set 39.72626 807.4693 390.4017 2.661136 3.300773  NaN -0.01373434
```

```
# ARIMA(1,2,1): MAPE = 3.099501
# ARIMA(1,2,2): MAPE = 3.279466
# ARIMA(1,2,3): MAPE = 3.271208
# ARIMA(2,2,1): MAPE = 3.310667
# ARIMA(2,2,2): MAPE = 3.321913
# ARIMA(2,2,3): MAPE = 3.25787
# ARIMA(3,2,1): MAPE = 3.405724
# ARIMA(3,2,2): MAPE = 3.384065
# ARIMA(3,2,3): MAPE = 3.300773
# Based on these values, the model with the lowest MAPE is:
#
# ARIMA(1,2,1) with a MAPE of 3.099501
# Therefore, the ARIMA(1,2,1) model is the best in terms of forecasting accuracy according to the MAPE criterion
among the models listed. Remember, a lower MAPE value indicates a better fit of the model to the data
```

```
# Fit the best ARIMA model based on lowest MAPE -substitute order which the combination that gave the lowest MAPE

best_model_MAPE <- Arima(ct_ts, order = c(1,2,1))
# Forecasting for a specified period (e.g., 41 days from the end of the series)
forecasted_values <- forecast(best_model_MAPE, h=41)
# Plot the forecast
plot(forecasted_values)
```




```
print(forecasted_values)
```

```
##           Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2021.0000       187752.7 186614.0 188891.4 186011.1 189494.3
## 2021.0029       189797.5 188269.3 191325.6 187460.3 192134.6
## 2021.0058       191842.2 189920.7 193763.7 188903.5 194780.9
## 2021.0087       193887.0 191592.2 196181.8 190377.4 197396.5
## 2021.0116       195931.7 193266.2 198597.3 191855.1 200008.3
## 2021.0145       197976.5 194939.6 201013.3 193332.0 202621.0
## 2021.0174       200021.2 196609.5 203432.9 194803.4 205239.0
## 2021.0203       202066.0 198274.3 205857.6 196267.2 207864.7
## 2021.0232       204110.7 199933.2 208288.3 197721.7 210499.7
## 2021.0261       206155.5 201585.3 210725.6 199166.0 213144.9
## 2021.0290       208200.2 203230.4 213170.1 200599.5 215801.0
## 2021.0319       210245.0 204868.1 215621.9 202021.7 218468.2
## 2021.0348       212289.7 206498.3 218081.2 203432.5 221147.0
## 2021.0377       214334.5 208120.9 220548.1 204831.6 223837.3
## 2021.0406       216379.2 209735.8 223022.6 206219.1 226539.4
## 2021.0435       218424.0 211343.2 225504.8 207594.8 229253.1
## 2021.0464       220468.7 212942.9 227994.6 208958.9 231978.5
## 2021.0493       222513.5 214535.0 230492.0 210311.4 234715.6
## 2021.0522       224558.2 216119.5 232996.9 211652.3 237464.1
## 2021.0551       226603.0 217696.6 235509.4 212981.8 240224.1
## 2021.0580       228647.7 219266.2 238029.2 214300.0 242995.5
## 2021.0609       230692.5 220828.5 240556.4 215606.9 245778.1
## 2021.0638       232737.2 222383.5 243090.9 216902.6 248571.8
## 2021.0667       234782.0 223931.3 245632.6 218187.4 251376.6
## 2021.0696       236826.7 225472.0 248181.4 219461.2 254192.3
## 2021.0725       238871.5 227005.6 250737.3 220724.2 257018.7
## 2021.0754       240916.2 228532.3 253300.2 221976.6 259855.9
## 2021.0783       242961.0 230052.0 255870.0 223218.4 262703.6
## 2021.0812       245005.7 231564.9 258446.6 224449.7 265561.7
## 2021.0841       247050.5 233071.0 261030.0 225670.7 268430.2
## 2021.0870       249095.2 234570.4 263620.0 226881.5 271309.0
## 2021.0899       251140.0 236063.2 266216.7 228082.1 274197.9
## 2021.0928       253184.7 237549.5 268820.0 229272.7 277096.8
## 2021.0957       255229.5 239029.3 271429.7 230453.4 280005.6
## 2021.0986       257274.2 240502.6 274045.9 231624.2 282924.2
## 2021.1014       259319.0 241969.6 276668.4 232785.4 285852.6
## 2021.1043       261363.7 243430.3 279297.2 233936.9 288790.6
## 2021.1072       263408.5 244884.7 281932.3 235078.8 291738.1
## 2021.1101       265453.2 246333.0 284573.5 236211.4 294695.1
## 2021.1130       267498.0 247775.2 287220.8 237334.5 297661.5
## 2021.1159       269542.7 249211.3 289874.2 238448.4 300637.1
```