Table of Contents

Introduction

This report aims to perform analysis and forecasting upon daily new confirmed COVID-19 cases per million people in the United Kingdom. The dataset is extracted from Our World in Data, OWID within the time period of 1st January 2021 to 29th March 2022 with a total of 453 observations.

Prior to generating the time series, pre-processing is performed through running code shown in Code Snippet 1 upon the raw dataset. This pre-processing aims to ensure that the time series contains of all features required for analysis.

```
covid = read.csv("/Users/sabrina/Documents/2021/Data Science/Masters_APU/Sem 3/TSF/owid-covid-data.csv")
#country: United Kingdom
uk_raw = covid[covid$location == 'United Kingdom',]
uk_raw = uk_raw[(c("date","new_cases_per_million"))]
uk_raw$date = as.Date(as.character(uk_raw$date), format = '%Y-%m-%d')

#extract rows from 1st Jan 2021 onwards
uk = subset(uk_raw, date>='2021-01-01')

```

Code Snippet 1. Pre-processing of raw dataset

a) Time Series Plot for Confirmed COVID-19 Cases per Million in the United Kingdom

Through running the codes in Code Snippet 2, the preliminary time series plot for daily new confirmed COVID-19 cases per million people in the United Kingdom is obtained as shown in Figure 1. Prior to analysing the time series components, the missing values shown in Figure 1 requires further investigation and handling.

```

#time series plot
uk_ts = xts(uk$new_cases_per_million, uk$date)
plot(uk_ts,ylab='New Confirmed Covid Cases per Million', xlab='Date',
     main = 'New Confirmed Cases per Million Time Series in United Kingdom')

```

Code Snippet 2. Plot time series

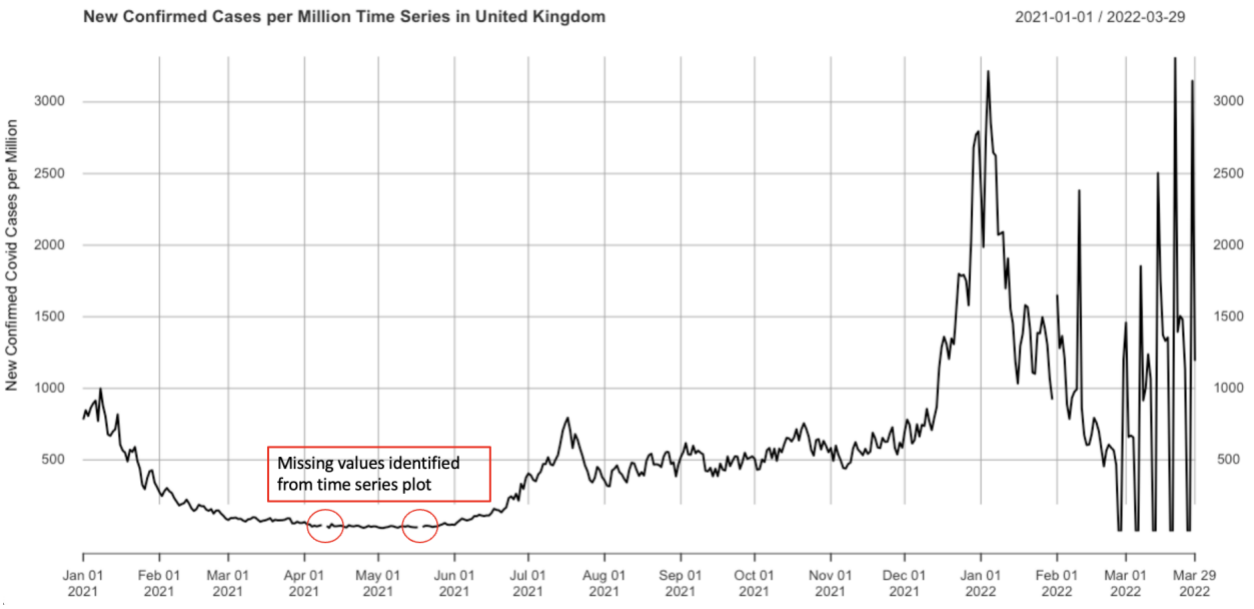


Figure 1. Preliminary Time Series Plot

As shown in Figure 2, three missing values have been identified within the time series. Spline imputation is implemented to perform imputation of missing values.

```
> colSums(sapply(uk_ts,is.na))
[1] 3
> #spline imputation
> uk_ts= na_interpolation(uk_ts, "spline")
> #perform imputation for missing values
> colSums(sapply(uk_ts,is.na))
[1] 0
```

Figure 2. Handling Missing Values

Upon completion of missing values imputation, Code Snippet 2 is executed to obtain the finalised Time Series Plot as shown in Figure 3.

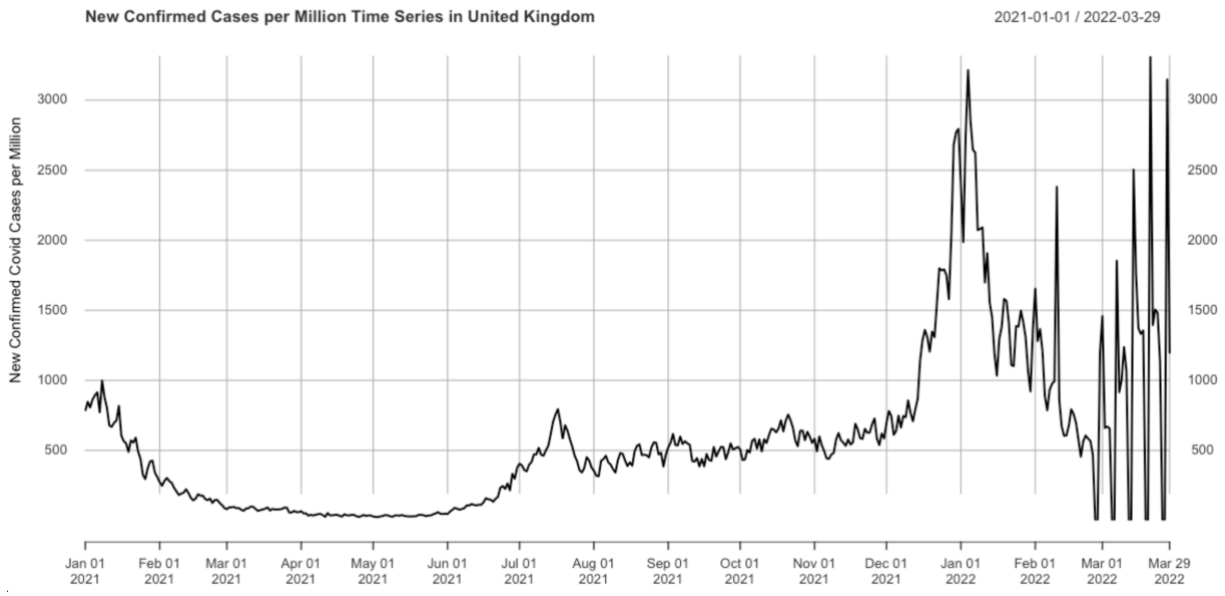


Figure 3. Time Series Plot for New Confirmed Cases per Million in United Kingdom

The presence of time series components are analysed below:

Trend

Through observation, a trend series is observed where an overall downwards trend is observed between periods Jan 01, 2021 to Mar 01, 2021 and Jan 01, 2022 to Mar 01, 2022, and an overall upwards trend is observed between periods Aug 01, 2021 to Jan 01, 2022 and Mar 01, 2022 to Mar 29, 2022.

The presence of trend is verified through the Mann-Kendall's Trend Test. The following hypotheses are used:

H_0: There is a no trend component in the time series.
H_1: There is a trend component in the time series.

```
> library(Kendall)
> MannKendall(uk_ts)
tau = 0.468, 2-sided pvalue =< 2.22e-16
```

Figure 4. Mann-Kendall Trend Test Results

The results of the Mann-Kendall's Trend Test is shown in Figure 5 where p-value is $2.22 \times 10\text{^}(-16)$. As p-value is less than 0.05, H_0 is rejected. It is concluded that a trend component is present in the time series at a 95% confidence interval.

Seasonality

Seasonality is observed within different phases of the time series as denoted in Figure 5 which each seasonal pattern in each phase is highlighted. Majority of these weekly seasonality variations are additive whereby size of fluctuations remain roughly constant. This is only an exception within the seasonal variations within Mar 01, 2022 to Mar 29, 2022.

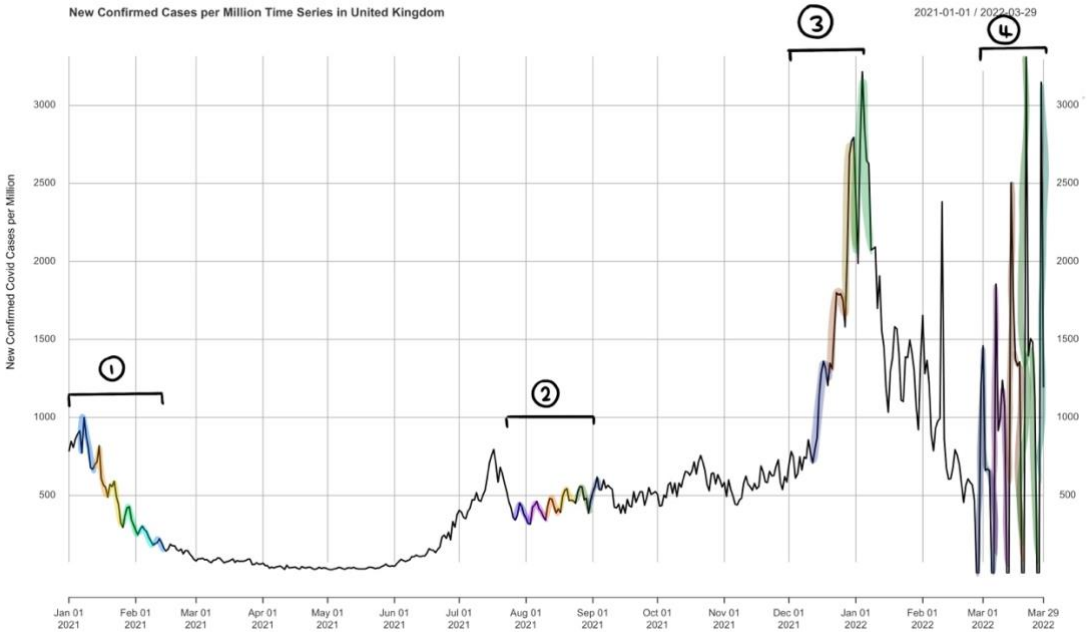


Figure 5. Observed Seasonality in Time Series

The presence of seasonality is verified through implementing isSeasonal() function and Kruskall Wallis test in R where results are shown in Figure 6. The Kruskall-Wallis test uses the following hypothesis:

H_0: There is no seasonality in the time series.
H_1: There is seasonality in the time series.

```
> isSeasonal(uk_ts)
[1] TRUE
> #Kruskall Wallis
> kw(uk_ts) #p-value = 0 < 0.05, Ho is rejected. There is seasonal variation in the series
Test used:  Kruskall Wallis

Test statistic:  118.45
P-value:  0
```

Figure 6. Seasonality Testing Results

As shown in Figure 6, the p-value is 0. As p-value is less than 0.05, H_0 is rejected. It is concluded that a seasonality component is present in the time series at a 95% confidence interval. This also aligns with the results of the isSeasonal() function where 'TRUE' is returned.

Additionally, the time series is decomposed to visualise each time series component as shown in Figure 7. The consistent size of fluctuations in the seasonal component and independence between trend and seasonal patterns shows that seasonal variation is this time series is additive.

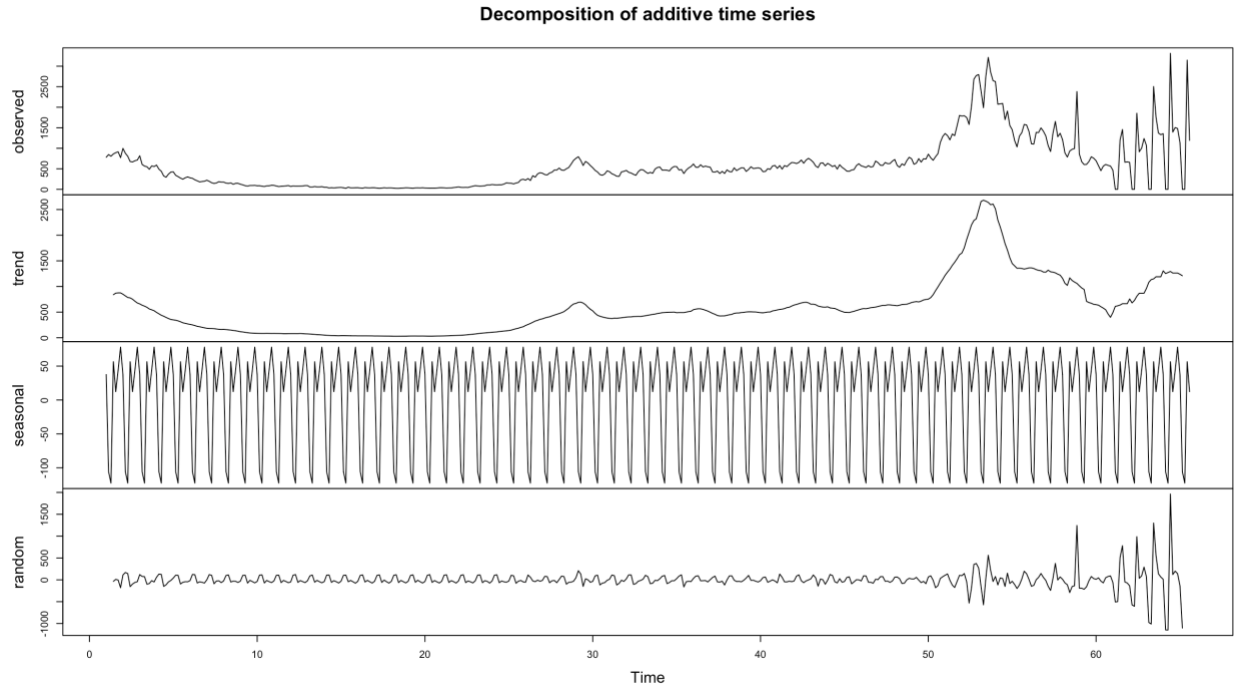


Figure 7. Decomposed Time Series

Irregular variations

Irregular variations are observed to be present throughout the time series. This is also shown in Figure 7 where random fluctuations are isolated and visualised.

b) Forecasting Methods

As trend and seasonal components have been identified within the time series, an important criteria for the selection of forecasting methods is the ability to perform smoothing for trend, seasonality, and irregular variations. Thus, the three selected forecasting methods are:

Holt-Winters' Method

The Holt-Winter's Method is an extension of the basic Exponential Smoothing technique to include an estimate for trend and seasonality. As an additive seasonality is observed in this report, the Holt-Winter's additive Method is used. The four equations required for this method are:

a. The exponentially smoothed series

$$L\_t = \alpha(y\_t - S\_(t - s) + (1 - \alpha)(L\_(t - 1) + b\_(t - 1))$$

b. The trend estimate

$$b_t = \beta(L_t - L_{t-1}) + (1 - \beta)b_{t-1}$$

c. The seasonality estimate

$$S_t = \gamma(y_t - L_t) + (1 - \gamma)S_{t-s}$$

d. The forecast for *m* period into the future

$$F_{t+m} = L_t + mb_t + S_{t+m-s}$$

Where,

- $L_t = level\ of\ series$
- $\alpha = smoothing\ constant\ for\ the\ data$
- $y_t = new\ observation\ or\ actual\ value\ in\ period\ t$
- $\beta = smoothing\ constant\ for\ trend\ estimate$
- $b_t = trend\ estimate$
- $\gamma = smoothing\ constant\ for\ seasonality\ estimate$
- $S_t = seasonal\ component\ estimate$
- $m = Number\ of\ periods\ in\ the\ forecast\ lead\ period$
- $s = length\ of\ seasonality\ (number\ of\ periods\ in\ the\ season)$
- $F_{t+m} = forecast\ for\ m\ periods\ into\ the\ future$

The smoothing parameters α, β, γ are smoothing factors for the data, trend, and seasonal variations respectively where α lies between 0 and 1 and a larger value of α provides higher weight of more recent observations to forecasts.

Decomposition Model

The decomposition process considers trend and seasonality through decomposing each component from the time series. Firstly, trend and irregular variations are estimated through the moving average technique.

Following this, the seasonal adjustment is estimated through implementing the following equations as per seasonal variation pattern:

$$Additive, SA_t = Y_t - S_t$$
$$Multiplicative, SA_t = Y_t / S_t$$

Where,

- $SA_t = Seasonal\ adjustment$
- $Y_t = Observed\ value\ at\ period\ t$
- $S_t = Trend\ value\ at\ period\ t$

Finally, time series is reseasonalised where forecasted values are calculated through implementing the following equations according to the identified seasonal variation pattern:

$$Additive, Y_{est} = T_{est} + SA_t$$
$$Multiplicative, Y_{est} = T_{est} * SA_t$$

Where,

- $Y\_est = Forecasted\ data\ value$
- $T\_est = Projected\ trend\ value$

In this report, the seasonal variation has been identified to be additive. Thus, the Additive Decomposition Model will be implemented for forecasting.

Regression Model with Dummy variables

Multiple Regression Model with dummy variables can be used to perform forecasting for time series with seasonal components with distinct seasonal patterns.

Dummy variables correspond to each season where the number of dummy variables required is equivalent to $total\ number\ of\ seasonality - 1$, while the final season is

hidden within the regression intercept. With this model, the effects of each season can be interpreted through comparison with the season hidden within the regression intercept.

The regression model is represented as:

$$Y_t = \beta_0 + \beta_1\, t + \beta_2\, S_1 + \beta_3\, S_2 + \cdots + \beta_t\, S_{t-1} + \epsilon$$

Where,

- $\beta = Variable\ coefficient$
- $S = Dummy\ variable$
- $\epsilon = Random\ error$

Each part of the model accounts for the following time series components:

$Trend$: $\beta_0 + \beta_1\, t$
$Seasonal$: $\beta_2\, S_1 + \beta_3\, S_2 + \cdots + \beta_t\, S_{t-1}$
$Irregular\ Variation$: ϵ

c) Data Partitioning and Time Series Forecasting

Code Snippet 3 is executed to perform Data Partitioning where dataset is split into a training and testing dataset using a 80:20 ratio. This avoids overfitting of the forecast models. Following the identification of the optimum forecast model, recombination of the datasets must be implemented to further increase forecasting accuracy through retaining valuable recent information.

```
45
46  #Data partitioning - 80% and 20%
47  k = round(length(uk_ts)*0.8,0) ;k
48  train = subset(uk_ts, end=k)
49  test = subset(uk_ts, start=k+1)
50
```

Code Snippet 3. Data Partitioning

Holt-Winters' Model

Approach 1: HoltWinters()

An initial Holt-Winter's Model is trained using the HoltWinters() function as shown in Code Snippet 4. The smoothing constants are shown in Figure 8 where $\alpha = 0.506, \beta = 0.115, and\ \gamma = 0.732$.

```
69  #holt winters method
70  #holt winters model A using HoltWinters function
71  hw_modelA = HoltWinters(train, seasonal = 'additive')
72
73  #hw model details
74  summary(hw_modelA)
75  hw_modelA$alpha
76  hw_modelA$beta
77  hw_modelA$gamma
78  #alpha = 0.5055, beta = 0.1151, gamma = 0.7320
```

Code Snippet 4. Training Holt-Winter's Model - HoltWinters() function

```
> hw_modelA$alpha
    alpha
0.5055098
> hw_modelA$beta
     beta
0.1150548
> hw_modelA$gamma
    gamma
0.7320395
```

Figure 8. HoltWinters() Model Details

Approach 2: hw()

The Holt-Winter's Model is further optimised through training with the hw() function as shown in Code Snippet 5. After optimisation, smoothing constants are $\alpha = 0.504, \beta = 0.058, and\ \gamma = 0.285$.

```
91  #holt winters model B using hw function
92  hw_modelB = hw(train, initial = 'optimal')
93  summary(hw_modelB)
94  #alpha = 0.5041, beta = 0.0581, gamma = 0.2851
95
```

Code Snippet 5. Training Holt-Winter's Model - hw() function

```
Call:
 hw(y = train, initial = "optimal")

  Smoothing parameters:
    alpha = 0.5041
    beta  = 0.0581
    gamma = 0.2851
```

Figure 9. hw() Model Details

Through comparing the performance between the models trained from the two approaches as shown in Figure 10, it is observed that MAE and RMSE values on Test set are lower with Approach 2. Thus, it is concluded that the Holt-Winter's Model is successfully optimised through Approach 2.

```
> hw_forecastA = forecast(hw_modelA, h=length(test))
> accuracy(hw_forecastA, test)
                     ME       RMSE        MAE      MPE     MAPE       MASE       ACF1 Theil's U
Training set    3.69547   54.73256   34.28694 1.078539 10.29711  0.4242752 0.1351476        NA
Test set    -3631.76516 4233.51584 3707.24155     -Inf      Inf 45.8743278 0.9041778         0
> hw_forecastB = forecast(fitted(hw_modelB), h=length(test))
> accuracy(hw_forecastB,test)
                     ME       RMSE        MAE        MPE     MAPE       MASE       ACF1 Theil's U
Training set   3.083573   56.18599   35.21046 -0.9500142 10.90692  0.4187487 0.6121581        NA
Test set    -916.462386 1262.47865 1103.31835       -Inf      Inf 13.1214743 0.6123811         0
```

Figure 10. Comparison between performance of Holt-Winter's model before and after optimisation

Forecasting will be performed with Holt-Winters' Model trained using Approach 2 where the model's fitted and forecasted values are shown in Figure 11. Through Figure 11, it is seen that the confidence interval of forecasts generated are able to contain the actual values.

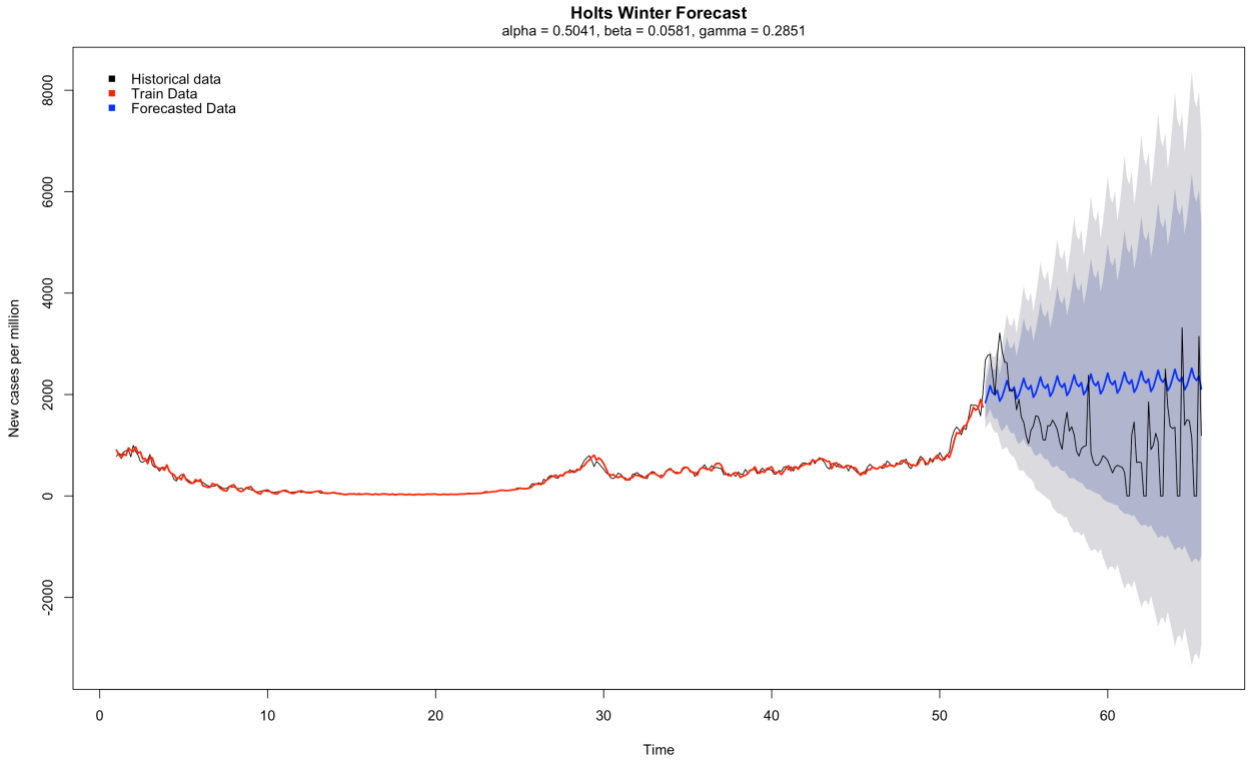


Figure 11. Observed and Forecast Values with Holt-Winters' Model

Additive Decomposition Model

Before training the Additive Decomposition Model, the seasonally adjusted dataset is plot as shown in Figure 12. The seasonally adjusted plot aims to visualise the daily new confirmed COVID-19 cases per million people in the United Kingdom without the effects of seasonality such

that when seasonally adjusted data is below the historical data, it is suggested that seasonal effects causes covid cases to be high.

However, Figure 12 shows that the seasonally adjusted data consistently fluctuates with oscillations. Thus, the effects of seasonality on the COVID-19 cases are not intuitive through observation.

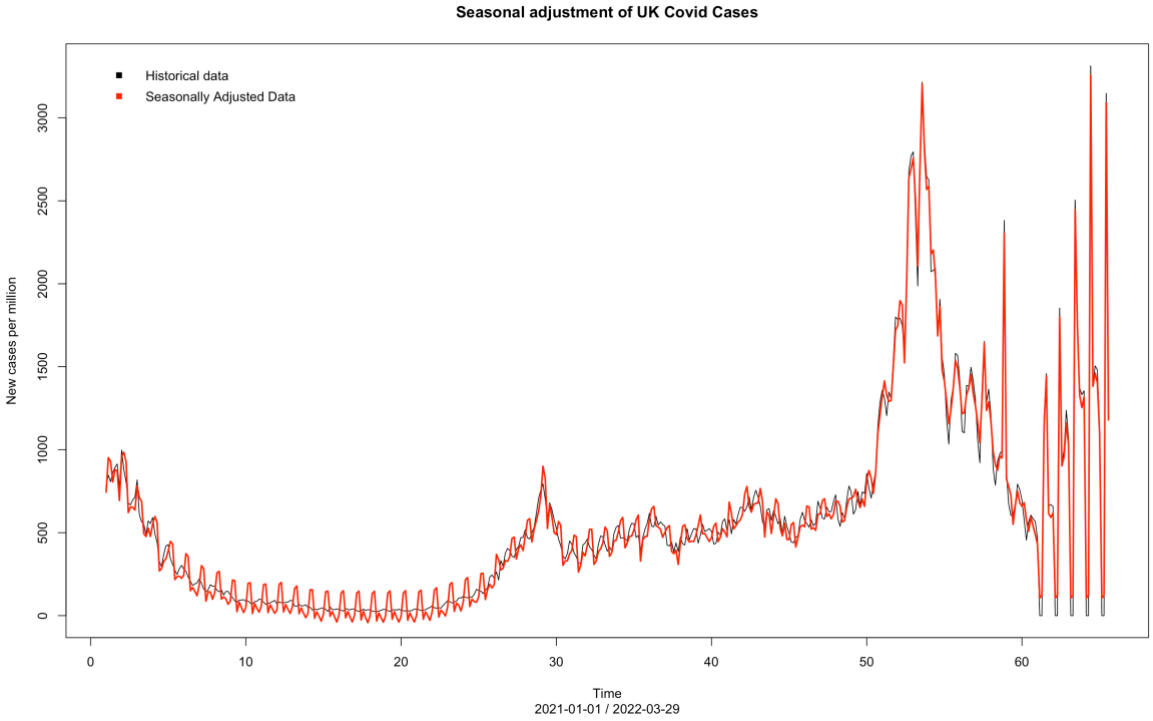


Figure 12. Seasonally Adjusted Data

Fitted and forecasted values of Additive Decomposition Model are shown in Figure 13. Figure 13 shows that the forecasted values greatly overestimates the number of COVID-19 cases as compared to the historical data.

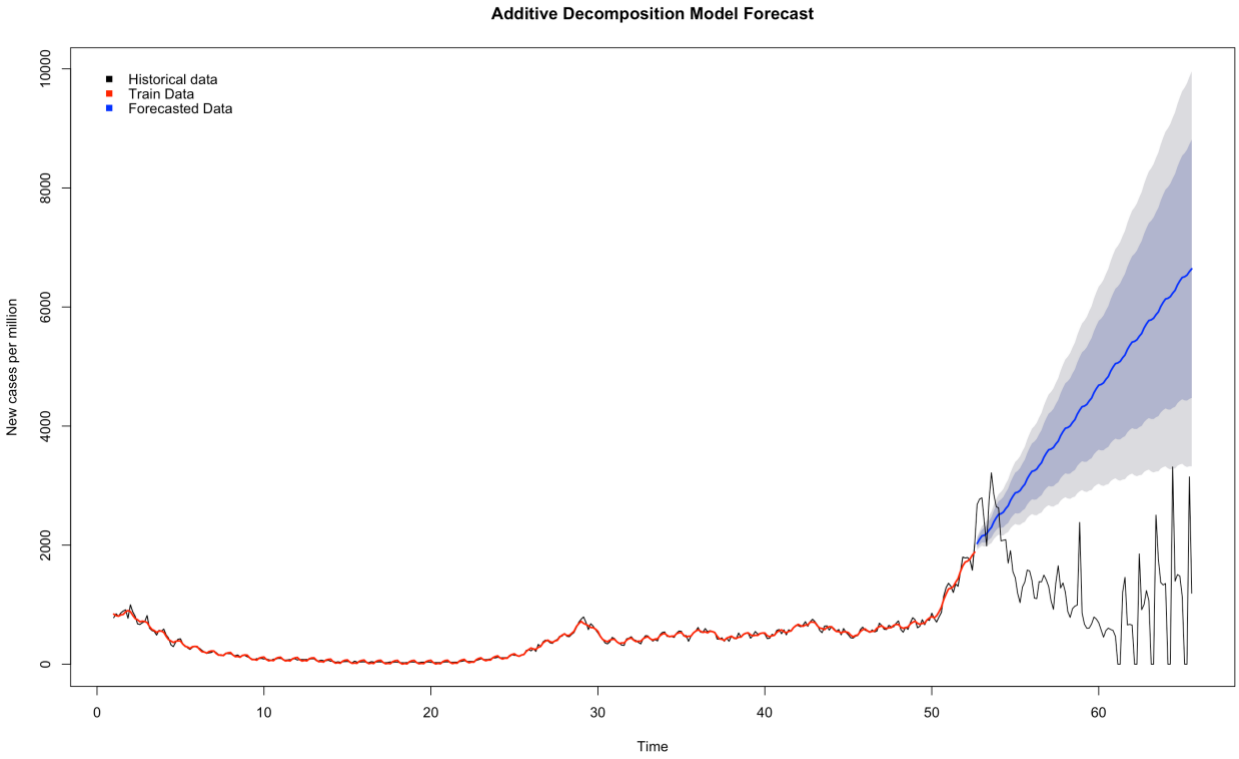


Figure 13. Observed and Forecast Values with Additive Decomposition Model

Multiple Regression Model with Dummy Variables

The trained Multiple Regression Model with Dummy Variables are investigated through model details provided in Figure 14. It is observed that time variable is statistically significant as the p-value is less than 0.05. This shows that the trend component in significant within the time series. However, the variables of season 2 to season 7 are statistically insignificant as the p-values are greater than 0.05. This implies that the seasonal component does not exist in the series.

```
> summary(reg_modelA)

Call:
tslm(formula = train ~ season + time)

Residuals:
    Min      1Q  Median      3Q     Max
-326.34 -170.36  -55.99   36.54 1306.51

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 104.5420    46.2526   2.260   0.0244 *
season2     -40.2087    54.8170  -0.734   0.4637
season3     -63.5970    54.8175  -1.160   0.2468
season4     -54.6774    54.8184  -0.997   0.3192
season5     -56.8537    54.8197  -1.037   0.3004
season6     -38.3814    55.0853  -0.697   0.4864
season7     -20.2480    55.0860  -0.368   0.7134
time          1.8541     0.1406  13.188   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 279.5 on 354 degrees of freedom
Multiple R-squared:  0.3317,    Adjusted R-squared:  0.3184
F-statistic:  25.1 on 7 and 354 DF,  p-value: < 2.2e-16
```

Figure 14. Model Details of Multiple Regression Model with Dummy Variables

Fitted and forecasted values of Multiple Regression Model with dummy variables are shown in Figure 15. Figure 15 shows that the forecasted values successfully accounts for trend from time period 60 onwards. However, there is still disparity between forecasted and exact values.

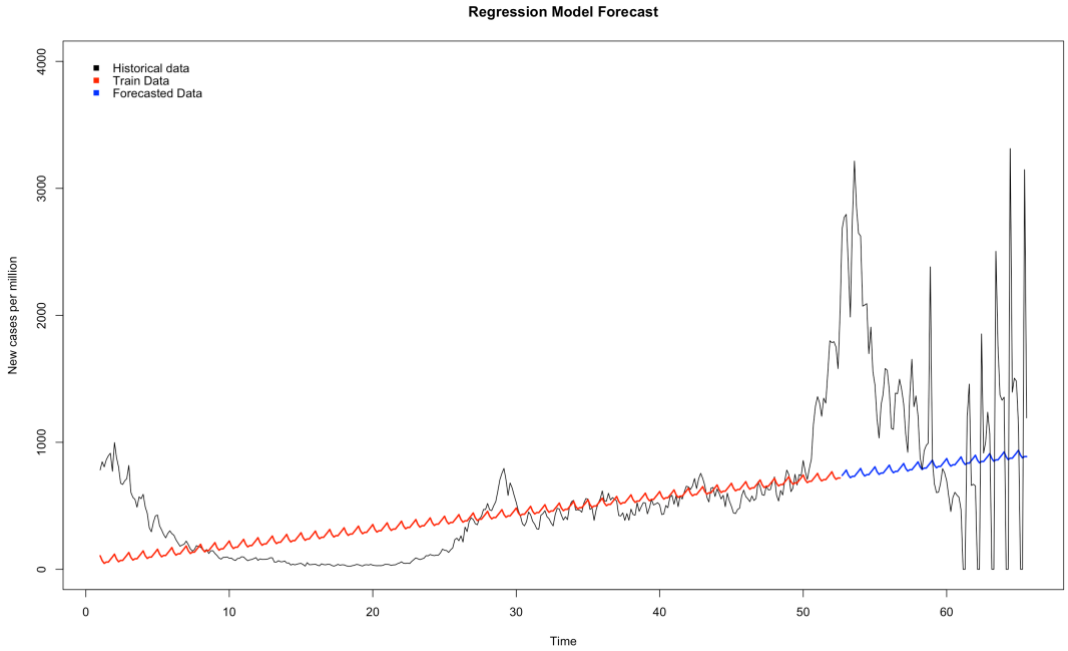


Figure 15. Observed and Forecast Values with Multiple Regression Model with Dummy Variables

d) Performance Evaluation with Forecast Errors

The forecast errors commonly used to compare between models are: Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Square Error (RMSE). Each of these metrics are described below:

Mean Absolute Error, MAE

MAE measures the average absolute magnitude of errors of the forecasted values.

$$MAE = \frac{\sum_{t=1}^{n}(|e_t|)}{n}$$

Where,

- $e_t = forecast\ errors = actual\ value - forecasted\ value\ at\ time\ period, t$
- $n = total\ number\ of\ forecast\ values$

Mean Absolute Percentage Error, MAPE

MAPE evaluates each model's performance through calculating a percentage score to represent the average deviation between forecasted values and actual values. Using a percentage score empowers comparison across data with different scales.

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{e_t}{Y_t}\right| \times 100\%$$

Where,

- $e_t = forecast\ errors = actual\ value - forecasted\ value\ at\ time\ period, t$
- $Y\_t = actual\ value\ at\ time\ period, t$
- $n = total\ number\ of\ forecast\ values$

Root Mean Square Error, RMSE

Similarly to MAE, RMSE measures the average absolute magnitude of errors between forecasted values and actual values. However, the RMSE formula contains a squaring function which gives higher weight to large errors. The difference between RMSE and MAE value provides insights to the variation of magnitude in forecast errors.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}e_t^2}$$

Where,

- $e_t = forecast\ errors = actual\ value - forecasted\ value\ at\ time\ period, t$
- $n = total\ number\ of\ forecast\ values$

In this report, **MAE** and **RMSE** are used to evaluate and compare the performance between the forecast models implemented in Section C. As the dataset in this report contains actual values of 0 to represent absence of new confirmed COVID-19 cases per million people in the United Kingdom, this will lead to undefined values of MAPE after substituting 0 as the fraction denominator. In R, it is displayed as "Inf".

Code Snippet 6 is executed to extract forecast performance of Holt-Winters Model in Figure 16, Additive Decomposition Model in Figure 17, and Multiple Regression Model in Figure 18.

```
#Holt Winters
accuracy(hw_forecastB,test)

#Additive Decomposition Model
accuracy(add_forecastA, test)

#Multiple Regression Model with Dummy Variables
rbind(accuracy(reg_modelA), accuracy(reg_forecastA$mean,test))
```

Code Snippet 6. Performance Evaluation of Forecast Models

Holt-Winters Model

```
> accuracy(hw_forecastB,test)
                    ME       RMSE        MAE        MPE     MAPE       MASE      ACF1 Theil's U
Training set  3.083573   56.18599   35.21046 -0.9500142 10.90692  0.4187487 0.6121581        NA
Test set     -916.462386 1262.47865 1103.31835       -Inf      Inf 13.1214743 0.6123811         0
```

Figure 16. Holt-Winters' Model Performance

Additive Decomposition Model

```
> accuracy(add_forecastA, test)
                     ME       RMSE        MAE    MPE     MAPE       MASE       ACF1 Theil's U
Training set     3.297683   53.99005   36.37015 2.1645 18.49922  0.4500533 0.01229775        NA
Test set     -3068.516561 3601.68514 3164.76800   -Inf      Inf 39.1616254 0.88630440         0
```

Figure 17. Additive Decomposition Model Performance

Multiple Regression Model with Dummy Variables

```
> rbind(accuracy(reg_modelA), accuracy(reg_forecastA$mean,test))
                         ME     RMSE      MAE       MPE     MAPE     MASE      ACF1
Training set -3.082372e-14 276.4061 185.4190 -144.4323 169.5282 2.294421 0.9398958
Test set      4.502771e+02 946.7488 720.8474      -Inf      Inf 0.556955 0.0000000
```

Figure 18. Multiple Regression Model with Dummy Variables Performance

Through comparing performance of each model on training set and test set, it is deduced that performance of Multiple Regression Model with dummy variables are most consistent with smallest performance gap between the training and test set.

Additionally, Multiple Regression Model forecasts on Test set derives the lowest MAE and RMSE values. Thus, it is concluded that the **Multiple Regression Model** is optimum.

Finally, the forecasting ability of the model is compared to a Naïve Forecasting Model through the MASE value. As MASE value is less than 1, it shows that Multiple Regression Model has better forecasting abilities than a Naïve Forecasting Model.

e) Autocorrelation Analysis and Transforming to Stationary

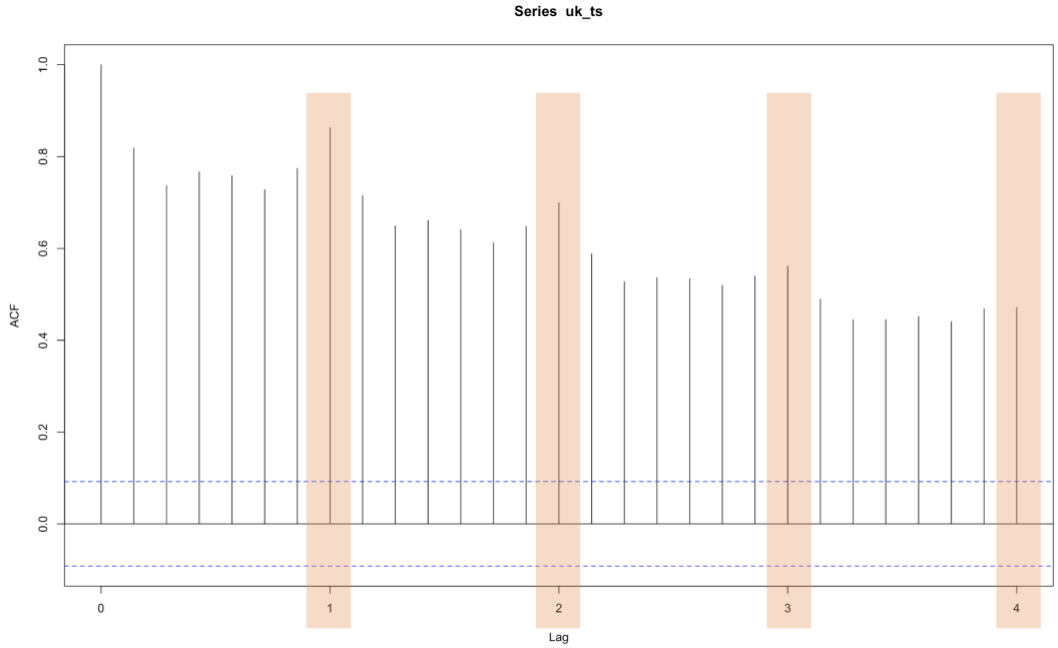


Figure 19. ACF Plot for Original Time Series with Seasonal Lags Highlighted

The seasonal lags within the ACF plot in Figure 19 is analysed to deduce if time series is seasonally stationary. The orange highlights in Figure 19 shows that autocorrelation is significant and slowly decreasing at seasonal lags (lag 7, 14, 21,28). This shows that the time series has a weekly seasonal component where it is not stationary in seasonal.

Additionally, the autocorrelation in non-seasonal lags are shown to be significant and slowly decreasing. This shows that that the time series is also not stationary in non-seasonal.

To verify these observations, further analysis is pursued through statistical testing.

Firstly, the Unit Root Test is implemented through the following hypotheses:

H_0 : The series contains a unit root, an unpredictable systematic pattern. This series is not stationary.
H_1 : The series does not contain a unit root. The series is stationary.

```
> adf.test(uk_ts)

        Augmented Dickey-Fuller Test

data:  uk_ts
Dickey-Fuller = -3.142, Lag order = 7, p-value = 0.09795
alternative hypothesis: stationary
```

Figure 20. Results of Unit Root Test

Through Figure 20, it is observed that p-value is 0.09795. As p-value is greater than 0.05, H_0 is not rejected. It is concluded that the series contains a unit root and is not stationary with a 95% confidence interval.

Statistical testing is also expanded through the Canova-Hansen test to check for seasonal stability. The Canova-Hansen test implements the following hypotheses:

H_0 : The series is stationary in seasonal.
H_1 : The series is not stationary in seasonal.

```
> ch.test(uk_ts)

        Canova and Hansen test for seasonal stability

data:  uk_ts

        statistic pvalue
Season1    1.8504      0 ***
Season2     0.978  0.002 **
Season3    0.9967 0.0018 **
Season4    2.1851      0 ***
Season5    2.2108      0 ***
Season6     1.995      0 ***
Season7    2.1693      0 ***
joint      2.9907   0.01 **
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Test type: seasonal dummies
NW covariance matrix lag order: 10
First order lag: no
Other regressors: no
P-values: based on response surface regressions
```

Figure 21. Results of Canova-Hansen Test

Through Figure 21, it is observed that p-values for all seasons are less than 0.05. Thus, H_0 is rejected. It is concluded that the series is not stationary in seasonal with a 95% confidence interval.

Finally, stationarity in the trend component is investigation through the KPSS Test. The KPSS Test implements the following hypotheses:

H_0 : The series is stationary in trend.
H_1 : The series is not stationary in trend.

```
> #check for stationary in trend
> kpss.test(uk_ts, null=c("Trend")) #p-value < 0.05

        KPSS Test for Trend Stationarity

data:  uk_ts
KPSS Trend = 0.47439, Truncation lag parameter = 5, p-value = 0.01

Warning message:
In kpss.test(uk_ts, null = c("Trend")) :
  p-value smaller than printed p-value
```

Figure 22. Results of KPSS Test

Through Figure 22, it is observed that p-value is 0.01. As p-value is less than 0.05, H_0 is rejected. It is concluded that the series is not stationary in trend with a 95% confidence interval.

The results of all statistical tests aligns with observations whereby the time series is not stationary. Thus, differencing is implemented to transform the series to stationary. As results of nsdiffs() is equivalent to 1, first degree of seasonal differencing is executed.

```
> nsdiffs(uk_ts)
[1] 1
```

Figure 23. Results of nsdiffs() function

After first degree of seasonal differencing, Unit Root test, KPSS test, and Canova and Hansen test is repeated with results shown in Appendix III. Results of all statistical test shows that the first degree seasonal differenced time series is stationary.

f) ARIMA or SARIMA Modelling

i) Proposing Models

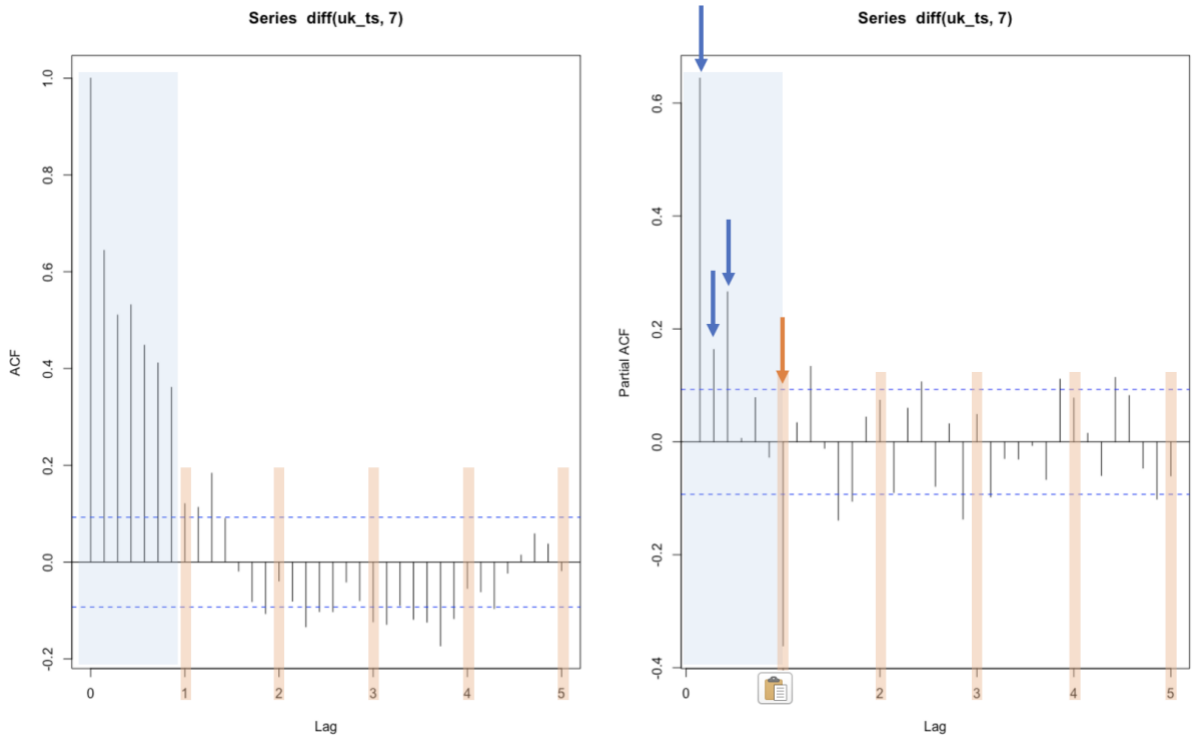


Figure 24. ACF and PACF for first degree seasonally differenced stationary time series

As the time series is verified to contain seasonal components, two SARIMA models will be proposed to perform forecasting. The SARIMA models are proposed through analysing the seasonal shaded in orange and non-seasonal component shaded in blue within the ACF and PACF plots in Figure 24. SARIMA models are denoted by:

$$ARIMA(p,d,q)(P,D,Q)_S$$

Where,

- p = order of regular AR component
- d = degree of regular differencing
- q = order of regular MA component
- $P = order\ of\ seasonal\ AR\ component\ at\ period\ S$
- $D = degree\ of\ seasonal\ differencing\ at\ period\ S$
- $Q = order\ of\ seasonal\ MA\ component\ at\ period\ S$
- $S = length\ of\ seasonality$

Model 1

As only first degree seasonal differencing has been applied, the general representation of SARIMA model to be proposed is ARIMA(p,0,q)(P,1,Q)[7],

Proposing P and Q values

In the seasonal lags, one significant spike as pointed with the orange arrow is identified in the PACF plot. The seasonal lags in PACF plot follows a cut off after first seasonal lag pattern while the ACF follows a die down exponentially with sine wave pattern. Thus, a seasonal AR model is proposed where P = 1 and Q = 0. The representation of SARIMA model to be proposed is developed into ARIMA(p,0,q)(1,1,0)[7].

Proposing p and q values

In the regular lags, three significant spikes as point with the blue arrow is identified in the PACF plot. The PACF plot follows a cut off after three lags while the ACF follows a die down exponentially with sine wave pattern. Thus, a regular AR model is proposed where p = 3 and q = 0. The final proposed SARIMA model is represented by **ARIMA(3,0,0)(1,1,0)[7].**

Model 2

The second SARIMA model proposed will leverage upon the auto.arima() function in R to obtain a SARIMA model with optimum Akaike's Information Criterion, AIC value. The results of auto.arima() function selected the SARIMA model resulting in an AIC value of 5916.1234 represented by **ARIMA(5,0,2)(1,1,0)[7].**

ii) Summary Output of Proposed SARIMA Models

Proposed Model 1: ARIMA(3,0,0)(1,1,0)[7]

```
> summary(arima_modelA)
Series: uk_ts
ARIMA(3,0,0)(1,1,0)[7]

Coefficients:
         ar1     ar2     ar3     sar1
      0.5098  0.0691  0.2633  -0.2999
s.e.  0.0458  0.0522  0.0458   0.0463

sigma^2 estimated as 35652:  log likelihood=-2968.94
AIC=5947.88   AICc=5948.01   BIC=5968.38

Training set error measures:
                    ME     RMSE      MAE MPE MAPE      MASE        ACF1
Training set 0.4444999 186.5096 86.98201 NaN  Inf 0.6164004 0.006392149
```

Figure 25. Summary Output for Proposed Model 1 - ARIMA(3,0,1)(1,1,0)[7]

Proposed Model 2: ARIMA(5,0,2)(1,1,0)[7]

```
> summary(arima_modelB)
Series: uk_ts
ARIMA(5,0,2)(1,1,0)[7]

Coefficients:
         ar1      ar2     ar3      ar4     ar5     ma1     ma2     sar1
      0.3907  -0.7583  0.7531  -0.0406  0.3156  0.1512  0.9588  -0.2313
s.e.  0.0480   0.0528  0.0510   0.0515  0.0506  0.0203  0.0180   0.0608

sigma^2 estimated as 32734:  log likelihood=-2948.85
AIC=5915.71   AICc=5916.12   BIC=5952.61

Training set error measures:
                    ME     RMSE      MAE MPE MAPE      MASE         ACF1
Training set 0.5569243 177.9051 87.38254 NaN  Inf 0.6192388 -0.02494337
```

Figure 26. Summary Output for Proposed Model 2 - ARIMA(5,0,2)(1,1,0)[7]

iii) Model Equation for Proposed SARIMA Models

Proposed Model 1: ARIMA(3,0,0)(1,1,0)[7]

$ARIMA(3,0,0)(1,1,0)[7]$

$$\phi_p(B)\,\Phi_P(B^s)\,\nabla_s^D\,\nabla^d\,y_t = \delta + \theta_q(B)\,\Theta_Q(B^s)\,\varepsilon_t$$

$$p=3,\ d=0,\ q=0,\ P=1,\ D=1,\ Q=0,\ s=7$$

$$\hookrightarrow \phi_3(B)\,\Phi_1(B^7)\,\nabla_1^7\,\nabla^0\,y_t = \delta + \cancel{\theta_0(B)\,\Theta_0\,B^7}\,\varepsilon_t$$

$$\hookrightarrow (1-\phi_1B-\phi_2B^2-\phi_3B^3)(1-\Phi_1B^7)(1-B^7)\,y_t = \delta + \varepsilon_t$$

$$\hookrightarrow (1-\Phi_1B^7-\phi_1B+\phi_1\Phi_1B^8-\phi_2B^2+\phi_2\Phi_1B^9-\phi_3B^3+\phi_3\Phi_1B^{10})(1-B^7)\,y_t = \delta+\varepsilon_t$$

$$\hookrightarrow (1-B^7-\Phi_1B^7+\Phi_1B^{14}-\phi_1B+\phi_1B^8+\phi_1\Phi_1B^8-\phi_1\Phi_1B^{15} -\phi_2B^2+\phi_2B^9+\phi_2\Phi_1B^9-\phi_2\Phi_1B^{16}-\phi_3B^3 +\phi_3B^{10}+\phi_3\Phi_1B^{10}-\phi_3\Phi_1B^{17})\,y_t = \delta+\varepsilon_t$$

$$\hookrightarrow y_t - y_{t-7} - \Phi_1y_{t-7} + \Phi_1y_{t-14} - \phi_1y_{t-1} + \phi_1y_{t-8} + \phi_1\Phi_1y_{t-8} - \phi_1\Phi_1y_{t-15} - \phi_2y_{t-2} + \phi_2y_{t-9} + \phi_2\Phi_1y_{t-9} - \phi_2\Phi_1y_{t-16} - \phi_3y_{t-3} + \phi_3y_{t-10} + \phi_3\Phi_1y_{t-10} - \phi_3\Phi_1y_{t-17} = \delta+\varepsilon_t$$

$$\hookrightarrow y_t - \phi_1y_{t-1} - \phi_2y_{t-2} - \phi_3y_{t-3} - (1+\Phi_1)y_{t-7} + \phi_1(1+\Phi_1)y_{t-8} + \phi_2(1+\Phi_1)y_{t-9} + \phi_3(1+\Phi_1)y_{t-10} + \Phi_1y_{t-14} - \phi_1\Phi_1y_{t-15} + \phi_2\Phi_1y_{t-16} - \phi_3\Phi_1y_{t-17} = \delta+\varepsilon_t$$

Sub in coefficient values :

$$\hookrightarrow y_t - 0.5098 y_{t-1} - 0.0691 y_{t-2} - 0.2633 y_{t-3}$$
$$- 0.7001 y_{t-7} + 0.3569 y_{t-8} + 0.04838 y_{t-9}$$
$$+ 0.1843 y_{t-10} - 0.2999 y_{t-14} - 0.1529 y_{t-15}$$
$$- 0.0207 y_{t-16} - 0.0790 y_{t-17} = \delta + \varepsilon_t$$

After seasonal differencing, $\delta = 0$.

$$\hookrightarrow y_t = 0.5098 y_{t-1} + 0.0691 y_{t-2} + 0.2633 y_{t-3}$$
$$+ 0.7001 y_{t-7} - 0.3569 y_{t-8} - 0.04838 y_{t-9}$$
$$- 0.1843 y_{t-10} + 0.2999 y_{t-14} + 0.1529 y_{t-15}$$
$$+ 0.0207 y_{t-16} + 0.0790 y_{t-17} + e_t$$

Proposed Model 2: ARIMA(5,0,2)(1,1,0)[7]

$$ARIMA(5,0,2)(1,1,0)$$

$$\phi_p(B)\,\Phi_P(B^s)\,\nabla_s^D\,\nabla^d\,y_t = \delta + \theta_q(B)\,\Theta_Q(B^s)\,\varepsilon_t$$

$$p=5,\ d=0,\ q=2,\ P=1,\ D=1,\ Q=0,\ s=7$$

$$\phi_5(B)\,\Phi_1(B^7)\,\nabla_7^1\,\nabla^0\,y_t = \delta + \theta_2(B)\,\Theta_0(B^7)\,\varepsilon_t$$

$$\hookrightarrow (1-\phi_1B-\phi_2B^2-\phi_3B^3-\phi_4B^4-\phi_5B^5)(1-\Phi_1B^7)(1-B^7)\,y_t = \delta + (1+\theta_1B+\theta_2B^2)\varepsilon_t$$

$$\hookrightarrow (1-\Phi_1B^7-\phi_1B+\phi_1\Phi_1B^8-\phi_2B^2+\phi_2\Phi_1B^9-\phi_3B^3+\phi_3\Phi_1B^{10}-\phi_4B^4+\phi_4\Phi_1B^{11}-\phi_5B^5+\phi_5\Phi_1B^{12})(1-B^7)\,y_t = \delta+\varepsilon_t+\theta_1\varepsilon_{t-1}+\theta_2\varepsilon_{t-2}$$

$$\hookrightarrow (1-B^7-\Phi_1B^7+\Phi_1B^{14}-\phi_1B+\phi_1B^8+\phi_1\Phi_1B^8-\phi_1\Phi_1B^{15}-\phi_2B^2+\phi_2B^9+\phi_2\Phi_1B^9-\phi_2\Phi_1B^{16}-\phi_3B^3+\phi_3B^{10}+\phi_3\Phi_1B^{10}-\phi_3\Phi_1B^{17}-\phi_4B^4+\phi_4B^{11}+\phi_4\Phi_1B^{11}-\phi_4\Phi_1B^{18}-\phi_5B^5+\phi_5B^{12}+\phi_5\Phi_1B^{12}-\phi_5\Phi_1B^{19})\,y_t = \delta+\varepsilon_t+\theta_1\varepsilon_{t-1}+\theta_2\varepsilon_{t-2}$$

$$\hookrightarrow (1-\phi_1B-\phi_2B^2-\phi_3B^3-\phi_4B^4-\phi_5B^5-(1+\Phi_1)B^7+\phi_1(1+\Phi_1)B^8+\phi_2(1+\Phi_1)B^9+\phi_3(1+\Phi_1)B^{10}+\phi_4(1+\Phi_1)B^{11}+\phi_5(1+\Phi_1)B^{12}+\Phi_1B^{14}-\phi_1\Phi_1B^{15}-\phi_2\Phi_1B^{16}-\phi_3\Phi_1B^{17}-\phi_4\Phi_1B^{18}-\phi_5\Phi_1B^{19})\,y_t = \delta+\varepsilon_t+\theta_1\varepsilon_{t-1}+\theta_2\varepsilon_{t-2}$$

$$\hookrightarrow y_t - \phi_1 y_{t-1} - \phi_2 y_{t-2} - \phi_3 y_{t-3} - \phi_4 y_{t-4} - \phi_5 y_{t-5} - (1+\Phi_1) y_{t-7}$$

$$+ \phi_1 (1+\Phi_1) y_{t-8} + \phi_2 (1+\Phi_1) y_{t-9} + \phi_3 (1+\Phi_1) y_{t-10} + \phi_4 (1+\Phi_1) y_{t-11}$$

$$+ \phi_5 (1+\Phi_1) y_{t-12} + \Phi_1 y_{t-14} - \phi_1 \Phi_1 y_{t-15} - \phi_2 \Phi_1 y_{t-16}$$

$$- \phi_3 \Phi_1 y_{t-17} - \phi_4 \Phi_1 y_{t-18} - \phi_5 \Phi_1 y_{t-19} = \delta + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2}$$

$$\hookrightarrow y_t - 0.3907 y_{t-1} + 0.7583 y_{t-2} - 0.7531 y_{t-3} + 0.0406 y_{t-4}$$

$$- 0.3156 y_{t-5} - 0.7687 y_{t-7} + 0.3003 y_{t-8} - 0.5829 y_{t-9}$$

$$+ 0.5789 y_{t-10} - 0.0312 y_{t-11} + 0.2426 y_{t-12}$$

$$- 0.2313 y_{t-14} - 0.0904 y_{t-15} - 0.1254 y_{t-16}$$

$$+ 0.1742 y_{t-17} - 0.0094 y_{t-18} - 0.07300 y_{t-19}$$

$$= \delta + \varepsilon_t + 0.1512 \varepsilon_{t-1} + 0.9588 \varepsilon_{t-2}$$

After seasonal differencing, $\delta = 0$.

$$\hookrightarrow y_t = 0.3907 y_{t-1} - 0.7583 y_{t-2} + 0.7531 y_{t-3} - 0.0406 y_{t-4}$$

$$+ 0.3156 y_{t-5} + 0.7687 y_{t-7} - 0.3003 y_{t-8} + 0.5829 y_{t-9}$$

$$- 0.5789 y_{t-10} + 0.0312 y_{t-11} - 0.2426 y_{t-12} + 0.2313 y_{t-14}$$

$$+ 0.0904 y_{t-15} + 0.1754 y_{t-16} - 0.1742 y_{t-17} - 0.0094 y_{t-18}$$

$$+ 0.07300 y_{t-19} + e_t + 0.1512 e_{t-1} + 0.9588 e_{t-2}$$

iv) Adequacy of Proposed SARIMA Models

The adequacy of each proposed model is verified through the checkresiduals() function. This function implements an overall model adequacy check through the Ljung-Box test. Residual analysis is also performed to check if residuals are according to the White Noise Process. According to the White Noise Process, residuals are expected to be fulfil these three requirements:

- Error terms to be independent and uncorrelated with anything
- Error terms to have a mean of 0
- Error terms to have constant variance

To perform the Ljung-Box test, the following hypotheses are used:

H_0 : Errors are independent, model is adequate.
H_1 : Errors are not independent, model is inadequate.

```
> checkresiduals(arima_modelA) #arima(3,0,0)(1,1,0)[7]

        Ljung-Box test

data:  Residuals from ARIMA(3,0,0)(1,1,0)[7]
Q* = 44.119, df = 10, p-value = 3.133e-06

Model df: 4.   Total lags used: 14
```

Figure 27. Ljung-Box Test Results for Proposed Model 1 - ARIMA(3,0,0)(1,1,0)[7]

Through Figure 27, p-value is $3.133\ x\ 10^{-06}$. As p-value is less than 0.05, H_0 is rejected. It is concluded that the proposed model 1, ARIMA(3,0,0)(1,1,0)[7] is inadequate with a 95% confidence interval.

```
> checkresiduals(arima_modelB) #arima(5,0,2)(1,1,0)[7]

        Ljung-Box test

data:  Residuals from ARIMA(5,0,2)(1,1,0)[7]
Q* = 26.519, df = 6, p-value = 0.0001781

Model df: 8.   Total lags used: 14
```

Figure 28. Ljung-Box Test Results for Proposed Model 2 - ARIMA(5,0,2)(1,1,0)[7]

Through Figure 28, p-value is 0.0001781. As p-value is less than 0.05, H_0 is rejected. It is concluded that the proposed model 2, ARIMA(5,0,2)(1,1,0)[7] is inadequate with a 95% confidence interval.

The results of Ljung-Box test can be visualised through residual plots in Figure 29 and 30. The significant autocorrelations in the ACF plots visualises the results of Ljung-Box Test such that error terms are not independent. In Figure 29, 6 significant autocorrelations are identified while in Figure 30, 3 significant autocorrelations are identified. This is reflected by a greater p-value in the Ljung-Box Test for proposed model 2, ARIMA(5,0,2)(1,1,0)[7].

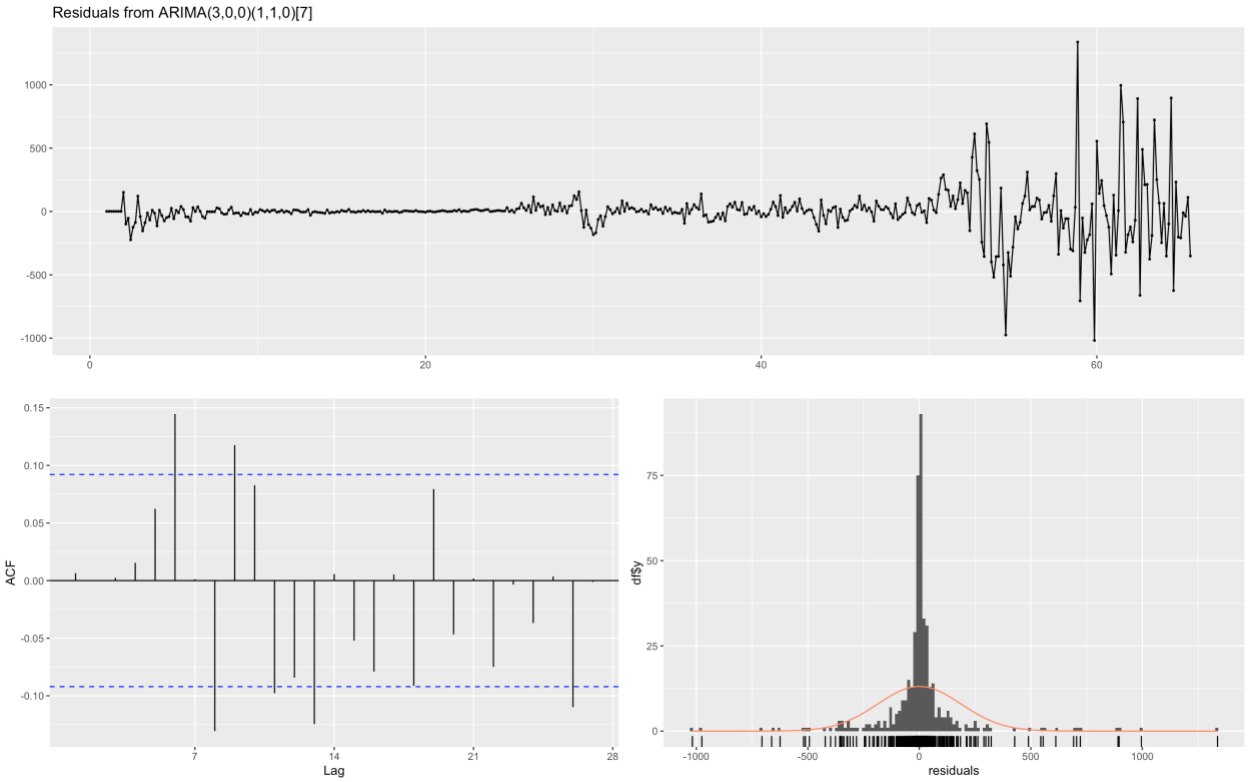


Figure 29. Residuals Check for Proposed Model 1 - ARIMA(3,0,0)(1,1,0)[7]

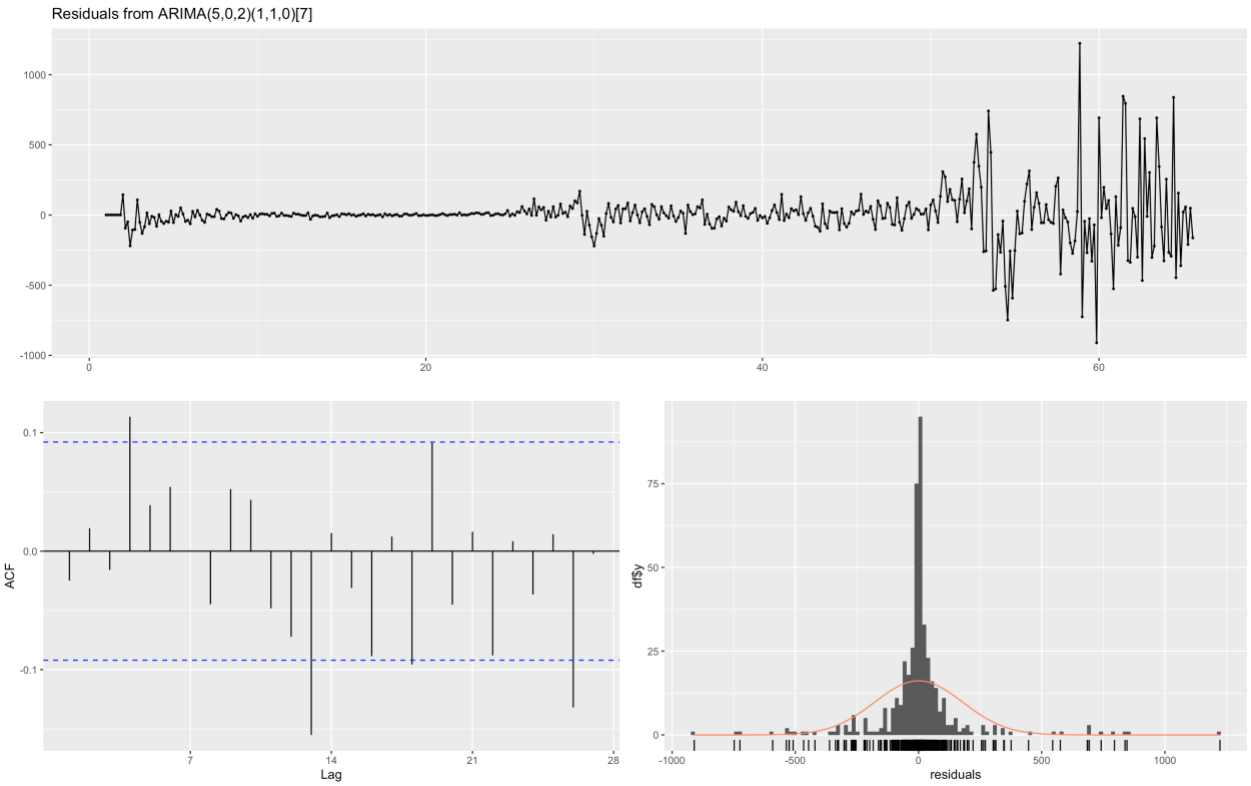


Figure 30. Residuals Check for Proposed Model 2 - ARIMA(5,0,2)(1,1,0)[7]

v) Significance of Parameter Coefficients of Proposed SARIMA Model

The significance of parameter coefficients of proposed SARIMA models are tested at $\alpha = 0.05$ through executing the coeftest() function in R.

The results of coeftest() for proposed model 1, ARIMA(3,0,0)(1,1,0)[7] is shown in Figure 31. Figure 31 shows that coefficient ar2 are statistically insignificant as p-value is greater than 0.05 while the remaining coefficients in the model are statistically significant as p-values are less than 0.05.

```
> coeftest(arima_modelA) #arima(3,0,0)(1,1,0)[7]

z test of coefficients:

      Estimate Std. Error z value  Pr(>|z|)
ar1   0.509785   0.045820 11.1258 < 2.2e-16 ***
ar2   0.069124   0.052244  1.3231    0.1858
ar3   0.263298   0.045775  5.7520 8.817e-09 ***
sar1 -0.299880   0.046274 -6.4805 9.141e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 31. Results of coefficient testing for Proposed Model 1 - ARIMA(3,0,0)(1,1,0)[7]

The results of coeftest() for proposed model 2, ARIMA(5,0,2)(1,1,0)[7] is shown in Figure 32. Figure 32 shows that coefficient ar4 are statistically insignificant as p-value is greater than 0.05 while he remaining coefficients in the model are statistically significant as p-values are less than 0.05.

```
> coeftest(arima_modelB) #arima(5,0,2)(1,1,0)[7]

z test of coefficients:

      Estimate Std. Error  z value  Pr(>|z|)
ar1   0.390730   0.047994   8.1412 3.913e-16 ***
ar2  -0.758252   0.052807 -14.3590 < 2.2e-16 ***
ar3   0.753145   0.051037  14.7569 < 2.2e-16 ***
ar4  -0.040579   0.051469  -0.7884 0.4304510
ar5   0.315634   0.050577   6.2407 4.356e-10 ***
ma1   0.151157   0.020295   7.4482 9.466e-14 ***
ma2   0.958764   0.018035  53.1621 < 2.2e-16 ***
sar1 -0.231293   0.060755  -3.8070 0.0001407 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 32. Results of coefficient testing for Proposed Model 2 - ARIMA(5,0,2)(1,1,0)[7]

In conclusion, both proposed models have one not significant coefficient while remaining coefficients are significant at $\alpha = 0.05$.

vi) Performance Evaluation using Forecast Errors for Proposed SARIMA Models

The performance of the proposed SARIMA models are shown in Figure 33 and 34.

Proposed Model 1: ARIMA(3,0,0)(1,1,0)[7]

```
> accuracy(arima_modelA)
                     ME     RMSE      MAE MPE MAPE      MASE         ACF1
Training set 0.4444999 186.5096 86.98201 NaN  Inf 0.6164004 0.006392149
```

Figure 33. Performance Evaluation of Proposed Model 1 - ARIMA(3,0,0)(1,1,0)[7]

Proposed Model 2: ARIMA(5,0,2)(1,1,0)[7]

```
> accuracy(arima_modelB)
                     ME     RMSE      MAE MPE MAPE      MASE        ACF1
Training set 0.5569243 177.9051 87.38254 NaN  Inf 0.6192388 -0.02494337
```

Figure 34. Performance Evaluation of Proposed Model 2 - ARIMA(5,0,2)(1,1,0)[7]

Through comparing the performance of proposed SARIMA models, it is concluded that proposed model 2 is better as RMSE value is lower. While MAE value of proposed model 1 is smaller, the lower RMSE value in proposed model 2 suggests that there are fewer large errors.

These models are also compared to the Multiple Regression Model whereby both proposed SARIMA models outperform the Multiple Regression Model with a lower RMSE and MAE.

g) Forecasting with optimum ARIMA or SARIMA model

As proposed model 2 - ARIMA(5,0,2)(1,1,0)[7] achieved less violation in white noise process and better performance evaluated through RMSE and MAE values, it is selected as the optimum model to perform forecasting for the next 5 periods.

Code snippet 7 is executed to perform forecasting. The fitted and forecasted values are visualised in Figure 35.

```
arima_forecastB = forecast(arima_modelB, h = 5)
```

Code Snippet 7. Forecast with optimum SARIMA Model - ARIMA(5,0,2)(1,1,0)[7]

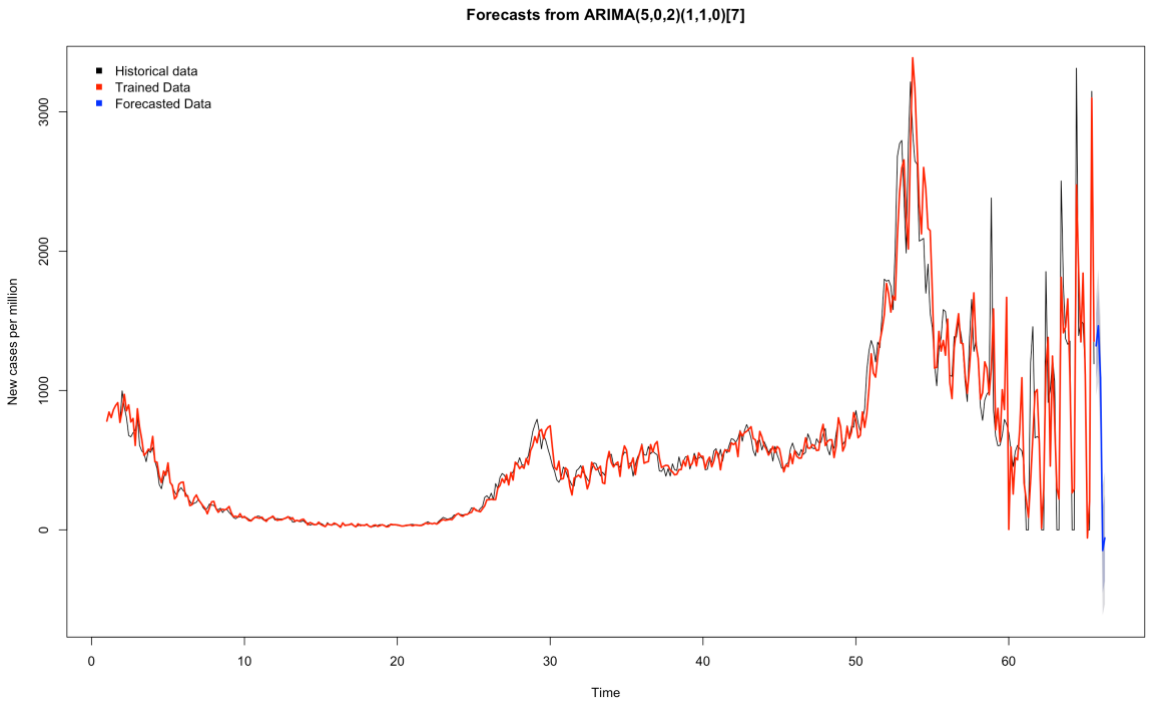
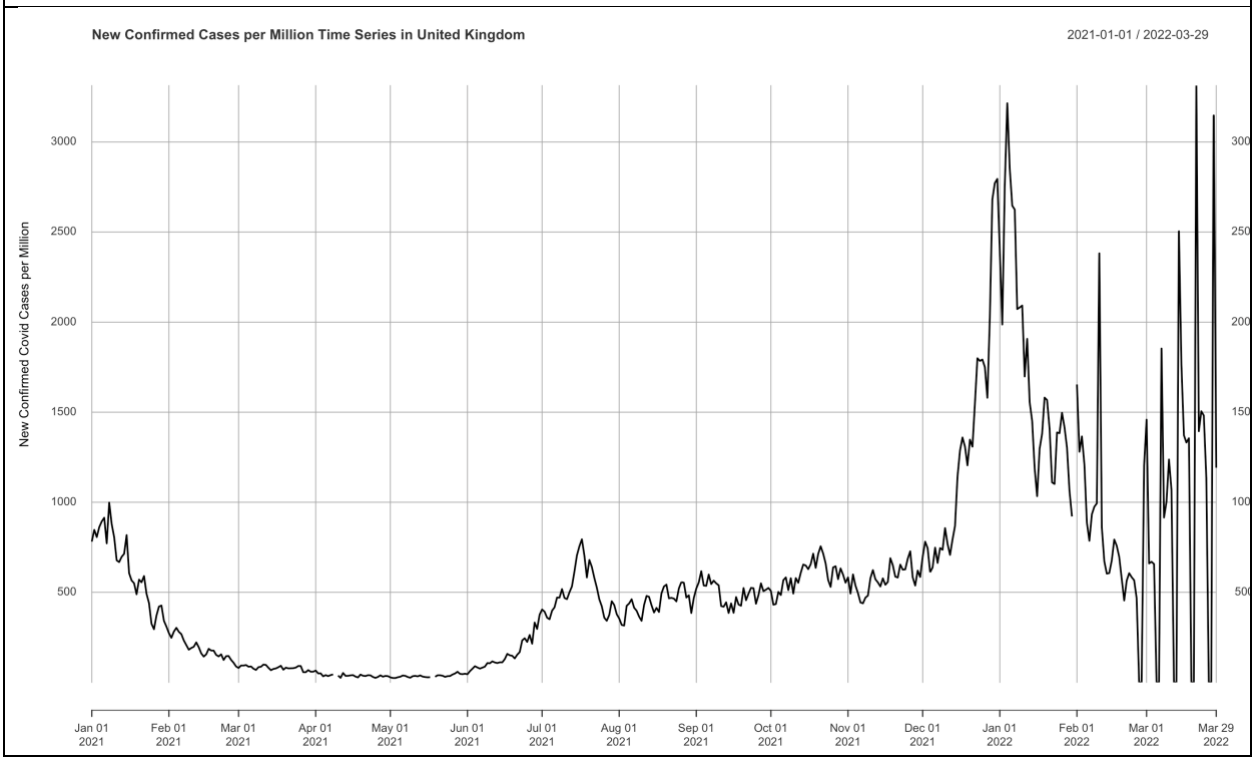


Figure 35. Fitted and Forecasted Values with ARIMA(5,0,2)(1,1,0)[7]

Appendix I: Time Series Plot for Confirmed COVID-19 Cases per Million in the United Kingdom

Plotting Time Series

```
covid = read.csv("/Users/sabrina/Documents/2021/Data Science/Masters_APU/Sem 3/TSF/owid-covid-data.csv")
#country: United Kingdom
uk_raw = covid[covid$location == 'United Kingdom',]
uk_raw = uk_raw[(c("date","new_cases_per_million"))]
uk_raw$date = as.Date(as.character(uk_raw$date), format = '%Y-%m-%d')

#extract rows from 1st Jan 2021 onwards
uk = subset(uk_raw, date>='2021-01-01')

#time series plot
uk_ts = xts(uk$new_cases_per_million, uk$date)
plot(uk_ts,ylab='New Confirmed Covid Cases per Million', xlab='Date',
     main = 'New Confirmed Cases per Million Time Series in United Kingdom')

#perform imputation for missing values
colSums(sapply(uk_ts,is.na))
#spline imputation
uk_ts= na_interpolation(uk_ts, "spline")

uk_ts = ts(uk$new_cases_per_million, frequency=7)
plot(uk_ts)
```



Verifying Trend

```r
#verifying trend
library(pastecs)
trend.test(uk_ts)
#p-value = 2.2x10^-16, Ho is rejected, trend component is present in the series
library(Kendall)
MannKendall(uk_ts)
```

```
> library(pastecs)

Attaching package: 'pastecs'

The following objects are masked from 'package:xts':

    first, last

> trend.test(uk_ts)

        Spearman's rank correlation rho

data:  uk_ts and time(uk_ts)
S = 6014600, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.6117904

Warning message:
In cor.test.default(x, time(x), alternative = "two.sided", method = "spearman") :
  Cannot compute exact p-value with ties
> #p-value = 2.2x10^-16, Ho is rejected, trend component is present in the series
> library(Kendall)
> MannKendall(uk_ts)
tau = 0.468, 2-sided pvalue =< 2.22e-16
```

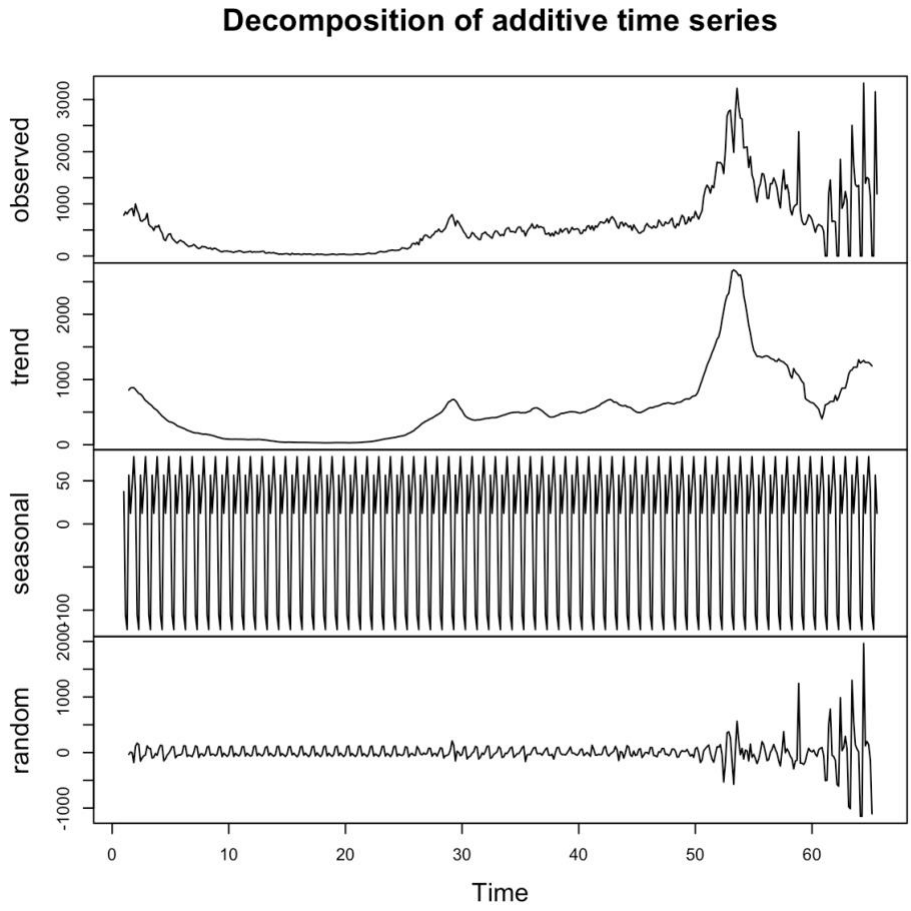
Verifying Seasonality

```r
#verifying seasonality
library(seastests)
isSeasonal(uk_ts)
#Kruskall Wallis
kw(uk_ts) #p-value = 0 < 0.05, Ho is rejected. There is seasonal variation in the series
```

```
> library(seastests)
> isSeasonal(uk_ts)
[1] TRUE
> #Kruskall Wallis
> kw(uk_ts) #p-value = 0 < 0.05, Ho is rejected. There is seasonal variation in the series
Test used:  Kruskall Wallis

Test statistic:  118.45
P-value:  0
```

Decompose Time Series

```
#decomposing time series
uk_ts_components = decompose(uk_ts)
plot(uk_ts_components)
```



Appendix II: Data Partitioning and Time Series Forecasting

Data Partitioning

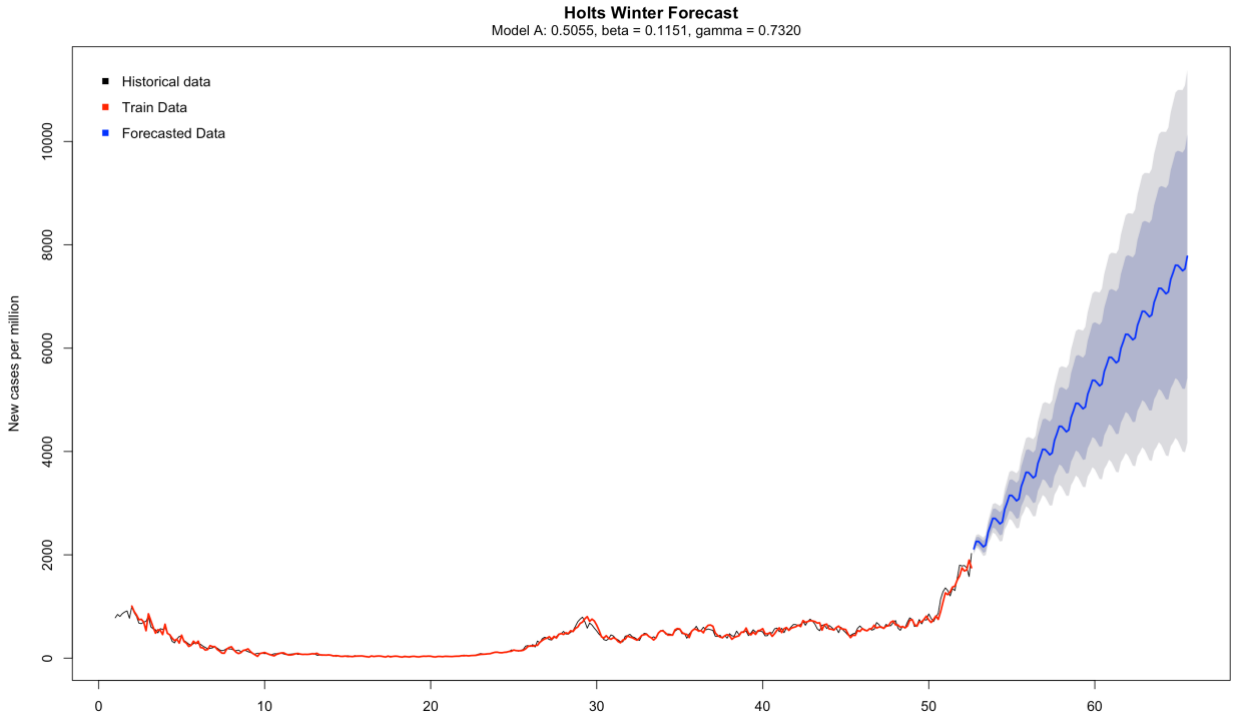
```
#Data partitioning - 80% and 20%
k = round(length(uk_ts)*0.8,0) ;k
train = subset(uk_ts, end=k)
test = subset(uk_ts, start=k+1)
```

Holt Winters Method A

```
#holt winters method
#holt winters model A using HoltWinters function
hw_modelA = HoltWinters(train, seasonal = 'additive')

#hw model details
summary(hw_modelA)
hw_modelA$alpha
hw_modelA$beta
hw_modelA$gamma
#alpha = 0.5055, beta = 0.1151, gamma = 0.7320

hw_forecastA = forecast(hw_modelA, h=length(test))
accuracy(hw_forecastA, test)
```

```
#plotting holts winter trained model A and forecasts against historical data
par(mfrow = c(1,1))
plot(hw_forecastA, main = 'Holts Winter Forecast',ylab = 'New cases per million')
mtext(side=3, line=0.5, at=33, 'Model A: 0.5055, beta = 0.1151, gamma = 0.7320')
lines(fitted(hw_modelA)[,1], col='red',lwd=2)
legend('topleft', c("Historical data", "Train Data", "Forecasted Data"), col=c("black",'red','blue'),
       pch=15, y.intersp=0.6,x.intersp=0.5, bty='n')
```

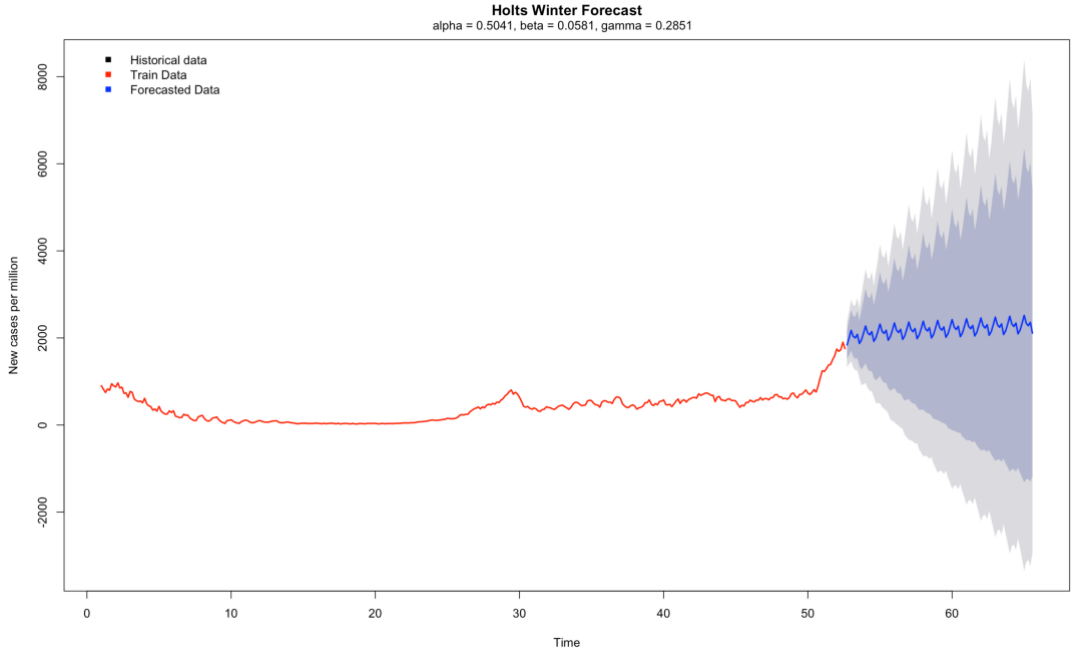
```
> #hw model details
> summary(hw_modelA)
             Length Class  Mode
fitted       1420   mts    numeric
x             362   ts     numeric
alpha           1   -none- numeric
beta            1   -none- numeric
gamma           1   -none- numeric
coefficients    9   -none- numeric
seasonal        1   -none- character
SSE             1   -none- numeric
call            3   -none- call
> hw_modelA$alpha
    alpha
0.5055098
> hw_modelA$beta
     beta
0.1150548
> hw_modelA$gamma
    gamma
0.7320395
> hw_forecastA = forecast(hw_modelA, h=length(test))
> accuracy(hw_forecastA, test)
                     ME       RMSE        MAE      MPE     MAPE       MASE      ACF1 Theil's U
Training set     3.69547   54.73256   34.28694 1.078539 10.29711  0.4242752 0.1351476        NA
Test set     -3631.76516 4233.51584 3707.24155     -Inf      Inf 45.8743278 0.9041778         0
>
> holt_forecastA
         Point Forecast       Lo 80    Hi 80       Lo 95     Hi 95
52.71429       1712.279  1468.69615 1955.861  1339.75129  2084.806
52.85714       1910.916  1579.93288 2241.900  1404.72089  2417.111
53.00000       2059.412  1639.57193 2479.252  1417.32205  2701.502
53.14286       2086.921  1598.11150 2575.731  1339.35113  2834.492
53.28571       1892.248  1392.17838 2392.318  1127.45747  2657.039
53.42857       1728.576  1220.39180 2236.760   951.37540  2505.776
53.57143       1854.054  1254.51495 2453.592   937.13835  2770.969
53.71429       1821.847  1179.82325 2463.872   839.95618  2803.739
53.85714       2018.277  1249.13631 2787.418   841.97770  3194.577
54.00000       2161.155  1276.30944 3046.000   807.90054  3514.409
54.14286       2177.703  1225.08496 3130.322   720.79919  3634.608
54.28571       1964.810  1050.92297 2878.696   567.14059  3362.479
54.42857       1787.067   906.93767 2667.197   441.02526  3133.109
54.57143       1909.463   917.36331 2901.563   392.17740  3426.749
54.71429       1869.973   848.30115 2891.644   307.46092  3432.484
54.85714       2065.433   882.22337 3248.643   255.86999  3874.996
55.00000       2205.843   884.30699 3527.380   184.72789  4226.959
55.14286       2217.578   831.36375 3603.792    97.54637  4337.609
55.28571       1996.681   697.11485 3296.247     9.16600  3984.196
55.42857       1812.759   586.60586 3038.912   -62.48027  3687.998
55.57143       1933.801   576.79322 3290.809  -141.56342  4009.166
55.71429       1891.111   516.52396 3265.699  -211.13869  3993.362
55.85714       2086.146   517.74740 3654.545  -312.51280  4484.805
56.00000       2225.473   497.18564 3953.760  -417.71441  4868.660
56.14286       2235.093   444.31407 4025.872  -503.66701  4973.853
```



Holt Winters Method B

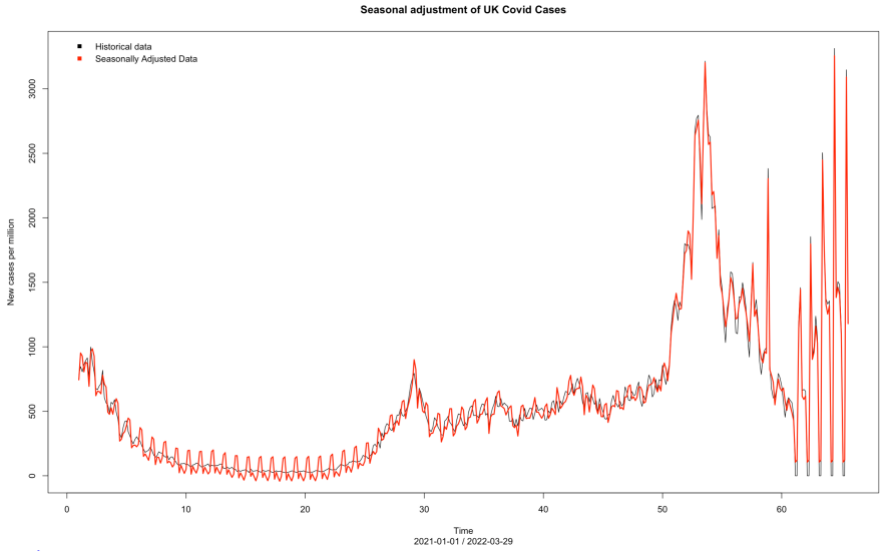
```r
#holt winters model B using hw function
hw_modelB = hw(train, initial = 'optimal')
summary(hw_modelB)
#alpha = 0.5041, beta = 0.0581, gamma = 0.2851

hw_forecastB = forecast(fitted(hw_modelB), h=length(test))
accuracy(hw_forecastB,test)

#plotting holts winter trained model A and forecasts against historical data
plot(hw_forecastB, main = 'Holts Winter Forecast',ylab = 'New cases per million', xlab = 'Time')
lines(fitted(hw_modelB), col='red',lwd=2)
legend('topleft', c("Historical data", "Train Data", "Forecasted Data"),
       col=c("black",'red','blue'), pch=15, y.intersp=0.6,x.intersp=0.5, bty='n')
mtext(side=3, line=0.5, at=33, 'alpha = 0.5041, beta = 0.0581, gamma = 0.2851')
```

```
> summary(hw_modelB)

Forecast method: Holt-Winters' additive method

Model Information:
Holt-Winters' additive method

Call:
 hw(y = train, initial = "optimal")

  Smoothing parameters:
    alpha = 0.5041
    beta  = 0.0581
    gamma = 0.2851

  Initial states:
    l = 838.0724
    b = 7.0064
    s = -9.4072 29.8537 -59.0839 -5.3614 -50.9493 36.7174
           58.2305

  sigma:  56.6104

     AIC     AICc      BIC
5067.808 5068.702 5114.508

Error measures:
                    ME     RMSE      MAE       MPE     MAPE      MASE      ACF1
Training set 2.661118 55.74366 35.38705 0.8090621 10.63772 0.4378881 0.1577115

Forecasts:
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
52.71429       2100.043 2027.493 2172.592 1989.088 2210.997
52.85714       2230.115 2146.887 2313.342 2102.829 2357.400
53.00000       2236.585 2141.971 2331.198 2091.886 2381.284
53.14286       2185.356 2078.709 2292.003 2022.253 2348.458
53.28571       2147.633 2028.352 2266.913 1965.209 2330.056
53.42857       2209.589 2077.113 2342.065 2006.984 2412.194
53.57143       2408.904 2262.701 2555.108 2185.305 2632.503
53.71429       2540.683 2370.680 2710.685 2280.686 2800.679
53.85714       2670.755 2486.799 2854.711 2389.418 2952.091
54.00000       2677.225 2478.759 2875.690 2373.698 2980.752
54.14286       2625.996 2412.494 2839.497 2299.474 2952.518
54.28571       2588.273 2359.235 2817.310 2237.991 2938.554
54.42857       2650.229 2405.179 2895.279 2275.457 3025.001
54.57143       2849.544 2588.024 3111.065 2449.583 3249.505
> accuracy(hw_forecastB,test)
                     ME       RMSE        MAE         MPE     MAPE       MASE      ACF1 Theil's U
Training set    3.083573   56.18599   35.21046 -0.9500142 10.90692  0.4187487 0.6121581        NA
Test set     -916.462386 1262.47865 1103.31835       -Inf      Inf 13.1214743 0.6123811         0
```



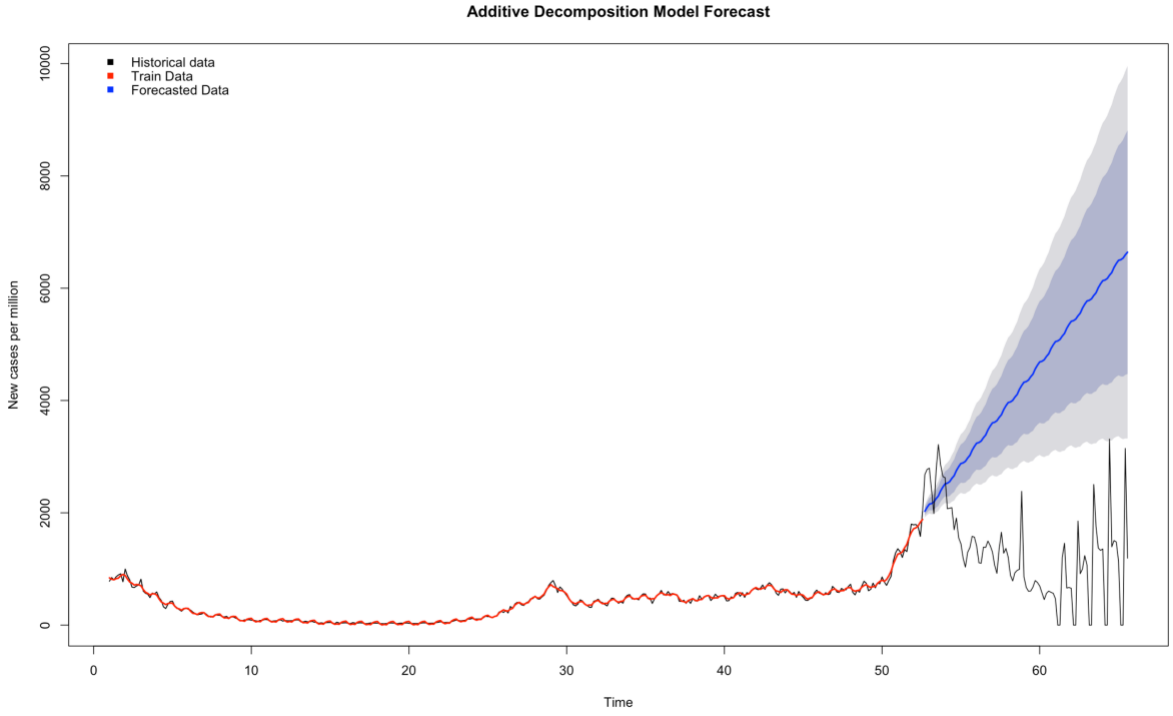
Decomposition Model

```r
#decomposition model
#additive model
add_fit = decompose(uk_ts)
add_fit
plot(add_fit)
seadj = uk_ts - add_fit$seasonal
plot(uk_ts, main = 'Seasonal adjustment of UK Covid Cases', sub = '2021-01-01 / 2022-03-29', ylab = 'New cases per million')
legend('topleft', c("Historical data", "Seasonally Adjusted Data"), col=c("black",'red'), pch=15, y.intersp=0.5,x.intersp=0.5, bty='n')
lines(seadj, col='red', lwd=2)

add_modelA = stl(train, s.window='periodic')
summary(add_modelA)

add_forecastA = stlf(train, s.window='period', h=length(test))
accuracy(add_forecastA, test)

plot(add_forecastA, main = 'Additive Decomposition Model Forecast', ylab= 'New cases per million', xlab = 'Time')
lines(uk_ts, col='black', lwd=1)
fitted = trendcycle(add_modelA) + seasonal(add_modelA)
lines(fitted, col='red', lwd = 2)
legend('topleft', c("Historical data", "Train Data", "Forecasted Data"), col=c("black",'red','blue'),
       pch=15, y.intersp=0.3, x.intersp=0.5, bty='n')
```




```
> summary(add_modelA)
 Call:
 stl(x = train, s.window = "periodic")

 Time.series components:
    seasonal             trend             remainder
 Min.   :-28.59578   Min.   :  28.3292   Min.   :-252.36351
 1st Qu.:-25.78836   1st Qu.:  85.2354   1st Qu.: -18.60580
 Median : -4.06430   Median : 424.2488   Median :   1.24436
 Mean   : -0.12089   Mean   : 402.2206   Mean   :  -0.22673
 3rd Qu.: 30.36315   3rd Qu.: 573.4202   3rd Qu.:  18.69443
 Max.   : 36.04246   Max.   :1907.2698   Max.   : 147.34509
 IQR:
     STL.seasonal STL.trend STL.remainder data
      56.15       488.18     37.30         487.85
   %  11.5        100.1       7.6          100.0

 Weights: all == 1

 Other components: List of 5
 $ win  : Named num [1:3] 3621 11 7
 $ deg  : Named int [1:3] 0 1 1
 $ jump : Named num [1:3] 363 2 1
 $ inner: int 2
 $ outer: int 0
> accuracy(add_forecastA, test)
                     ME       RMSE        MAE    MPE     MAPE       MASE       ACF1 Theil's U
Training set     3.297683   53.99005   36.37015 2.1645 18.49922  0.4500533 0.01229775        NA
Test set     -3068.516561 3601.68514 3164.76800   -Inf      Inf 39.1616254 0.88630440         0
```

```
> add_forecastA
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
52.71429       2027.943 1958.366 2097.519 1921.535 2134.351
52.85714       2096.600 2007.416 2185.784 1960.205 2232.995
53.00000       2153.971 2046.895 2261.047 1990.212 2317.730
53.14286       2165.556 2041.462 2289.650 1975.770 2355.342
53.28571       2192.716 2052.074 2333.358 1977.623 2407.810
53.42857       2251.649 2094.704 2408.593 2011.623 2491.675
53.57143       2298.908 2125.767 2472.048 2034.112 2563.703
53.71429       2389.786 2200.465 2579.107 2100.245 2679.327
53.85714       2458.443 2252.895 2663.990 2144.085 2772.801
54.00000       2515.814 2293.950 2737.678 2176.502 2855.126
54.14286       2527.399 2289.096 2765.703 2162.946 2891.853
54.28571       2554.560 2299.671 2809.448 2164.742 2944.377
54.42857       2613.492 2341.856 2885.128 2198.061 3028.923
54.57143       2660.751 2372.191 2949.311 2219.436 3102.065
54.71429       2751.629 2445.959 3057.300 2284.146 3219.112
54.85714       2820.286 2497.311 3143.261 2326.339 3314.234
55.00000       2877.657 2537.179 3218.136 2356.940 3398.375
55.14286       2889.242 2531.056 3247.429 2341.444 3437.041
55.28571       2916.403 2540.302 3292.504 2341.206 3491.600
55.42857       2975.335 2581.110 3369.560 2372.420 3578.250
55.57143       3022.594 2610.035 3435.154 2391.639 3653.549
55.71429       3113.472 2682.367 3544.578 2454.153 3772.792
55.85714       3182.129 2732.265 3631.993 2494.122 3870.137
56.00000       3239.501 2770.666 3708.335 2522.480 3956.521
```

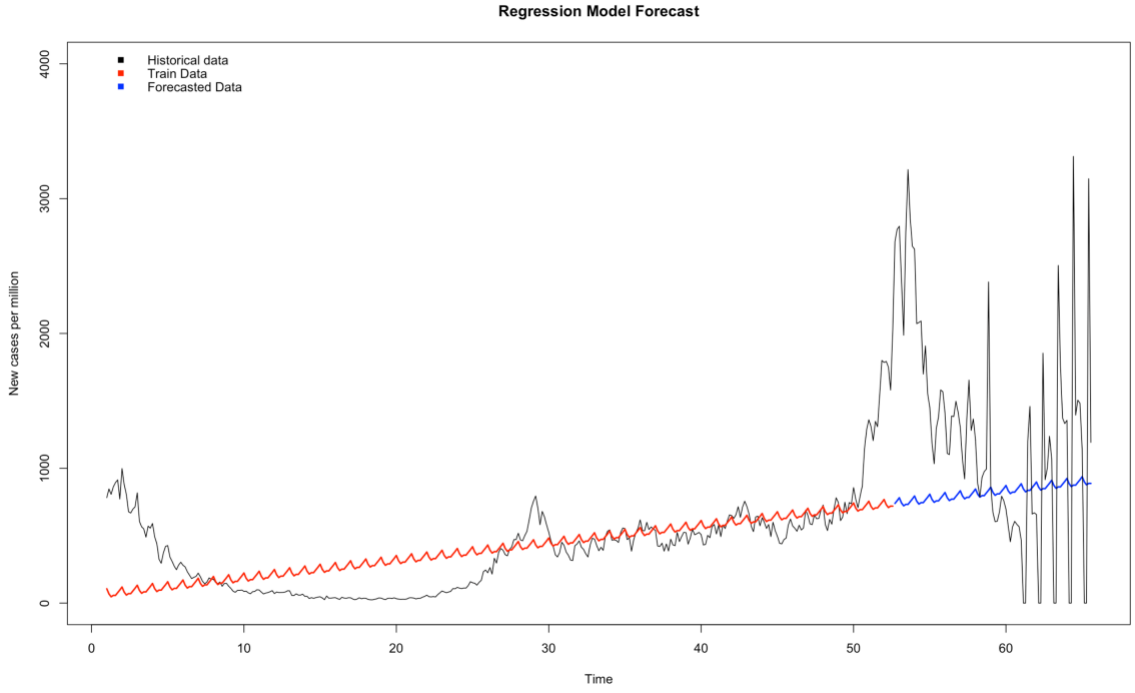


Regression Model with Dummy Variables

```r
#regression with dummy variables
season = seasonaldummy(train)
#creating time variable
time = 1:length(train)

reg_modelA = tslm(train ~ season+time)
summary(reg_modelA)

reg_forecastA = forecast(fitted(reg_modelA), h=length(test))
accuracy(reg_forecastA, test)
rbind(accuracy(reg_modelA), accuracy(reg_forecastA$mean,test))

plot(reg_forecastA, main = 'Regression Model Forecast', ylab= 'New cases per million', xlab = 'Time', ylim = c(0,4000))
lines(uk_ts, col='black', lwd=1)
lines(fitted(reg_modelA), col='red', lwd=2)
legend('topleft', c("Historical data", "Train Data", "Forecasted Data"), col=c("black",'red','blue'),
       pch=15, y.intersp=0.3, x.intersp=0.5, bty='n')
```

```
> summary(reg_modelA)

Call:
tslm(formula = train ~ season + time)

Residuals:
    Min      1Q  Median      3Q     Max
-326.34 -170.36  -55.99   36.54 1306.51

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 104.5420    46.2526   2.260   0.0244 *
season2     -40.2087    54.8170  -0.734   0.4637
season3     -63.5970    54.8175  -1.160   0.2468
season4     -54.6774    54.8184  -0.997   0.3192
season5     -56.8537    54.8197  -1.037   0.3004
season6     -38.3814    55.0853  -0.697   0.4864
season7     -20.2480    55.0860  -0.368   0.7134
time          1.8541     0.1406  13.188   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 279.5 on 354 degrees of freedom
Multiple R-squared:  0.3317,    Adjusted R-squared:  0.3184
F-statistic:  25.1 on 7 and 354 DF,  p-value: < 2.2e-16

> reg_forecastA = forecast(fitted(reg_modelA), h=length(test))
> accuracy(reg_forecastA, test)
                      ME         RMSE          MAE          MPE         MAPE         MASE      ACF1 Theil's U
Training set 1.575952e-13 3.998258e-13 2.280214e-13 4.168811e-14 8.702251e-14 1.756868e-14 0.2240025        NA
Test set     4.502771e+02 9.467488e+02 7.208474e+02         -Inf          Inf 5.554012e+01 0.5569550         0
> rbind(accuracy(reg_modelA), accuracy(reg_forecastA$mean,test))
                      ME     RMSE      MAE       MPE     MAPE     MASE      ACF1
Training set -3.082372e-14 276.4061 185.4190 -144.4323 169.5282 2.294421 0.9398958
Test set      4.502771e+02 946.7488 720.8474      -Inf      Inf 0.556955 0.0000000
```

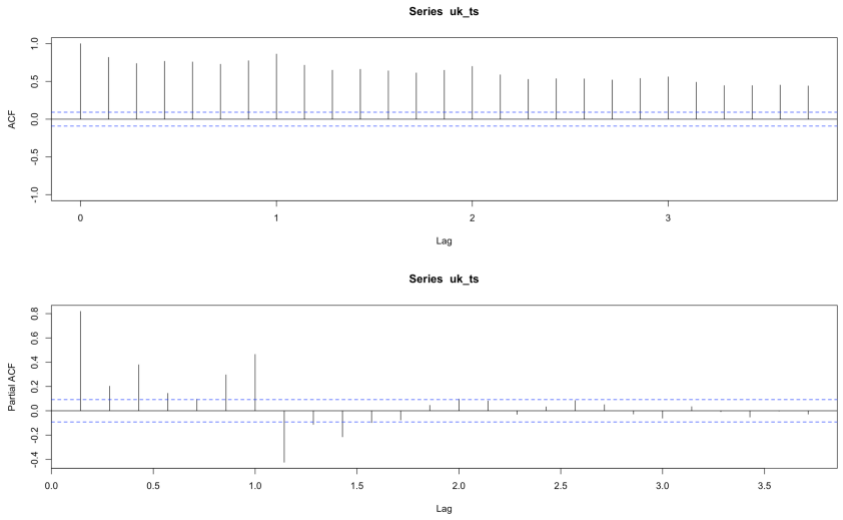
```
> reg_forecastA
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
52.71429       739.2071 739.2071 739.2071 739.2071 739.2071
52.85714       759.1946 759.1946 759.1946 759.1946 759.1946
53.00000       781.2967 781.2967 781.2967 781.2967 781.2967
53.14286       742.9422 742.9422 742.9422 742.9422 742.9422
53.28571       721.4080 721.4080 721.4080 721.4080 721.4080
53.42857       732.1817 732.1817 732.1817 732.1817 732.1817
53.57143       731.8596 731.8596 731.8596 731.8596 731.8596
53.71429       752.1859 752.1859 752.1859 752.1859 752.1859
53.85714       772.1734 772.1734 772.1734 772.1734 772.1734
54.00000       794.2756 794.2756 794.2756 794.2756 794.2756
54.14286       755.9211 755.9211 755.9211 755.9211 755.9211
54.28571       734.3868 734.3868 734.3868 734.3868 734.3868
54.42857       745.1606 745.1606 745.1606 745.1606 745.1606
54.57143       744.8384 744.8384 744.8384 744.8384 744.8384
54.71429       765.1648 765.1648 765.1648 765.1648 765.1648
54.85714       785.1523 785.1523 785.1523 785.1523 785.1523
55.00000       807.2545 807.2545 807.2545 807.2545 807.2545
55.14286       768.8999 768.8999 768.8999 768.8999 768.8999
55.28571       747.3657 747.3657 747.3657 747.3657 747.3657
55.42857       758.1395 758.1395 758.1395 758.1395 758.1395
55.57143       757.8173 757.8173 757.8173 757.8173 757.8173
55.71429       778.1436 778.1436 778.1436 778.1436 778.1436
55.85714       798.1312 798.1312 798.1312 798.1312 798.1312
56.00000       820.2333 820.2333 820.2333 820.2333 820.2333
56.14286       781.8788 781.8788 781.8788 781.8788 781.8788
56.28571       760.3445 760.3445 760.3445 760.3445 760.3445
56.42857       771.1183 771.1183 771.1183 771.1183 771.1183
56.57143       770.7961 770.7961 770.7961 770.7961 770.7961
56.71429       791.1225 791.1225 791.1225 791.1225 791.1225
```



Appendix III: Autocorrelation Analysis and Transforming to Stationary

Autocorrelation Analysis

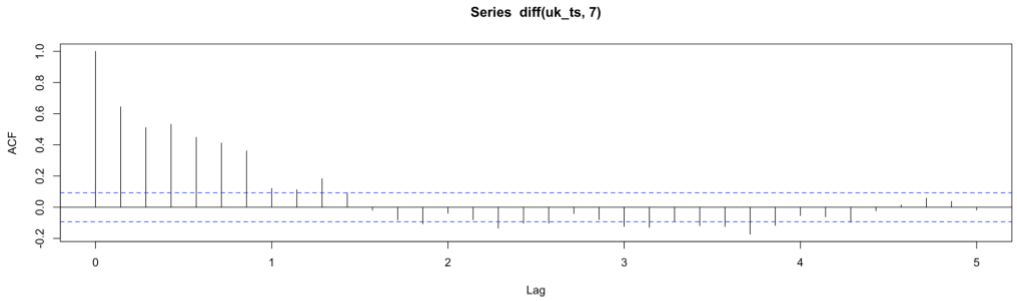
```r
#acf and pacf
par(mfrow = c(2,1))
acf(uk_ts, ylim = c(-1,1))
pacf(uk_ts)

#check for stationary
library(uroot)
ch.test(uk_ts)

#check for stationary in seasonal
library(tseries)
adf.test(uk_ts)

#check for stationary in trend
kpss.test(uk_ts, null=c("Trend")) #p-value < 0.05. Series is not stationary
```

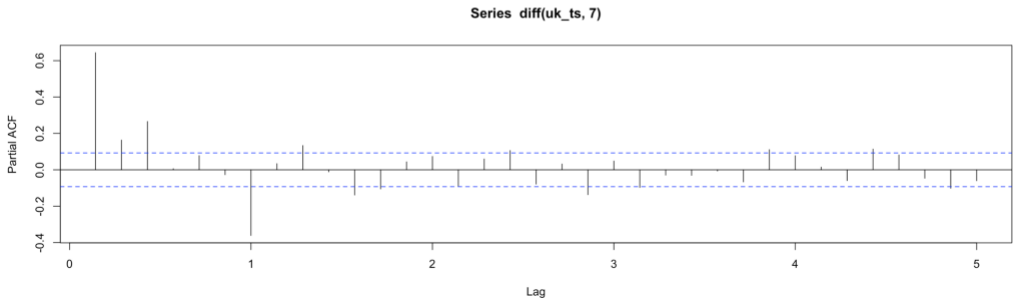


```
> library(uroot)
> ch.test(uk_ts)

        Canova and Hansen test for seasonal stability

data:  uk_ts

        statistic pvalue
Season1    1.8504      0 ***
Season2     0.978  0.002 **
Season3    0.9967 0.0018 **
Season4    2.1851      0 ***
Season5    2.2108      0 ***
Season6     1.995      0 ***
Season7    2.1693      0 ***
joint      2.9907   0.01 **
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Test type: seasonal dummies
NW covariance matrix lag order: 10
First order lag: no
Other regressors: no
P-values: based on response surface regressions
> adf.test(uk_ts)

        Augmented Dickey-Fuller Test

data:  uk_ts
Dickey-Fuller = -3.142, Lag order = 7, p-value = 0.09795
alternative hypothesis: stationary

> #check for stationary in trend
> kpss.test(uk_ts, null=c("Trend")) #p-value < 0.05. Series is not stationary

        KPSS Test for Trend Stationarity

data:  uk_ts
KPSS Trend = 0.47439, Truncation lag parameter = 5, p-value = 0.01

Warning message:
In kpss.test(uk_ts, null = c("Trend")) :
  p-value smaller than printed p-value
```

Seasonal Differencing

```
#seasonal differencing is required
#first degree of seasonal differencing
nsdiffs(uk_ts)
par(mfrow = c(1,2))
acf(diff(uk_ts,7), lag.max=35)
pacf(diff(uk_ts,7), lag.max = 35)

#stationarity testing
ch.test(diff(uk_ts,7)) #all variables significant, Series is stationary
adf.test(diff(uk_ts,7)) #p-value = 0.01 < 0.05. Ho is rejected. Series is stationary
kpss.test(diff(uk_ts,7), null=c("Trend")) #p-value = 0.1 > 0.05. Ho is accepted. Series is stationary in trend
```






```
> #stationarity testing
> ch.test(diff(uk_ts,7)) #all variables significant, Series is stationary

        Canova and Hansen test for seasonal stability

data:  diff(uk_ts, 7)

        statistic pvalue
Season1    0.0806  0.709
Season2    0.1176 0.5213
Season3    0.1257  0.488
Season4    0.3331 0.1118
Season5    0.0745 0.7451
Season6    0.1326 0.4617
Season7    0.0707 0.7682
joint      0.8333 0.5434
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Test type: seasonal dummies
NW covariance matrix lag order: 10
First order lag: no
Other regressors: no
P-values: based on response surface regressions
> adf.test(diff(uk_ts,7)) #p-value = 0.01 < 0.05. Ho is rejected. Series is stationary

        Augmented Dickey-Fuller Test

data:  diff(uk_ts, 7)
Dickey-Fuller = -6.0515, Lag order = 7, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(diff(uk_ts, 7)) : p-value smaller than printed p-value
> kpss.test(diff(uk_ts,7), null=c("Trend")) #p-value = 0.1 > 0.05. Ho is accepted. Series is stationary in trend

        KPSS Test for Trend Stationarity

data:  diff(uk_ts, 7)
KPSS Trend = 0.082163, Truncation lag parameter = 5, p-value = 0.1

Warning message:
In kpss.test(diff(uk_ts, 7), null = c("Trend")) :
  p-value greater than printed p-value
```

Appendix IV: Proposing ARIMA or SARIMA Models

Proposing ARIMA(3,0,0)(1,1,0)

```
#proposing arima models
arima_modelA = Arima(uk_ts, order = c(3,0,0), seasonal = c(1,1,0))#arima(3,0,0)(1,1,0)
summary(arima_modelA)
accuracy(arima_modelA)#arima(3,0,0)(1,1,0)[7]
```

```
> #proposing arima models
> arima_modelA = Arima(uk_ts, order = c(3,0,0), seasonal = c(1,1,0))#arima(3,0,0)(1,1,0)
> summary(arima_modelA)
Series: uk_ts
ARIMA(3,0,0)(1,1,0)[7]

Coefficients:
         ar1     ar2     ar3     sar1
      0.5098  0.0691  0.2633  -0.2999
s.e.  0.0458  0.0522  0.0458   0.0463

sigma^2 estimated as 35652:  log likelihood=-2968.94
AIC=5947.88   AICc=5948.01   BIC=5968.38

Training set error measures:
                    ME     RMSE      MAE MPE MAPE      MASE        ACF1
Training set 0.4444999 186.5096 86.98201 NaN  Inf 0.6164004 0.006392149
> accuracy(arima_modelA)#arima(3,0,0)(1,1,0)[7]
                    ME     RMSE      MAE MPE MAPE      MASE        ACF1
Training set 0.4444999 186.5096 86.98201 NaN  Inf 0.6164004 0.006392149
```

Proposing ARIMA with auto.arima() function – ARIMA(5,0,2)(1,1,0)

```
arima_modelB = auto.arima(uk_ts, trace=TRUE) #arima(5,0,2)(1,1,0)[7]
summary(arima_modelB)
accuracy(arima_modelB)#arima(5,0,2)(1,1,0)[7]
```

```
> arima_modelB = auto.arima(uk_ts, trace=TRUE) #arima(5,0,2)(1,1,0)[7]

 Fitting models using approximations to speed things up...

 ARIMA(2,0,2)(1,1,1)[7] with drift         : 5886.321
 ARIMA(0,0,0)(0,1,0)[7] with drift         : 6186.385
 ARIMA(1,0,0)(1,1,0)[7] with drift         : 5925.494
 ARIMA(0,0,1)(0,1,1)[7] with drift         : 6023.684
 ARIMA(0,0,0)(0,1,0)[7]                    : 6184.572
 ARIMA(2,0,2)(0,1,1)[7] with drift         : 5883.067
 ARIMA(2,0,2)(0,1,0)[7] with drift         : 5917.803
 ARIMA(2,0,2)(0,1,2)[7] with drift         : 5882.606
 ARIMA(2,0,2)(1,1,2)[7] with drift         : 5888.393
 ARIMA(1,0,2)(0,1,2)[7] with drift         : 5888.165
 ARIMA(2,0,1)(0,1,2)[7] with drift         : 5887.585
 ARIMA(3,0,2)(0,1,2)[7] with drift         : 5878.477
 ARIMA(3,0,2)(0,1,1)[7] with drift         : 5879.762
 ARIMA(3,0,2)(1,1,2)[7] with drift         : 5883.889
 ARIMA(3,0,2)(1,1,1)[7] with drift         : 5881.775
 ARIMA(3,0,1)(0,1,2)[7] with drift         : 5876.415
 ARIMA(3,0,1)(0,1,1)[7] with drift         : 5878.125
 ARIMA(3,0,1)(1,1,2)[7] with drift         : 5881.801
 ARIMA(3,0,1)(1,1,1)[7] with drift         : 5879.694
 ARIMA(3,0,0)(0,1,2)[7] with drift         : 5874.627
 ARIMA(3,0,0)(0,1,1)[7] with drift         : 5876.064
 ARIMA(3,0,0)(1,1,2)[7] with drift         : 5880.024
 ARIMA(3,0,0)(1,1,1)[7] with drift         : 5877.899
 ARIMA(2,0,0)(0,1,2)[7] with drift         : 5903.727
```

```
ARIMA(5,0,1)(0,1,2)[7] with drift         : 5880.131
ARIMA(4,0,2)(0,1,2)[7] with drift         : 5874.193
ARIMA(5,0,0)(0,1,2)[7] with drift         : 5878.108
ARIMA(5,0,2)(0,1,2)[7] with drift         : 5854.722
ARIMA(5,0,2)(0,1,1)[7] with drift         : 5851.214
ARIMA(5,0,2)(0,1,0)[7] with drift         : 5860.79
ARIMA(5,0,2)(1,1,1)[7] with drift         : 5847.01
ARIMA(5,0,2)(1,1,0)[7] with drift         : 5844.998
ARIMA(5,0,2)(2,1,0)[7] with drift         : 5854.729
ARIMA(5,0,2)(2,1,1)[7] with drift         : Inf
ARIMA(4,0,2)(1,1,0)[7] with drift         : 5878.337
ARIMA(5,0,1)(1,1,0)[7] with drift         : 5883.76
ARIMA(5,0,3)(1,1,0)[7] with drift         : 5845.36
ARIMA(4,0,1)(1,1,0)[7] with drift         : 5880.777
ARIMA(4,0,3)(1,1,0)[7] with drift         : Inf
ARIMA(5,0,2)(1,1,0)[7]                    : 5842.93
ARIMA(5,0,2)(0,1,0)[7]                    : 5858.784
ARIMA(5,0,2)(2,1,0)[7]                    : 5852.627
ARIMA(5,0,2)(1,1,1)[7]                    : 5844.917
ARIMA(5,0,2)(0,1,1)[7]                    : 5849.137
ARIMA(5,0,2)(2,1,1)[7]                    : Inf
ARIMA(4,0,2)(1,1,0)[7]                    : 5876.278
ARIMA(5,0,1)(1,1,0)[7]                    : 5881.725
ARIMA(5,0,3)(1,1,0)[7]                    : 5843.29
ARIMA(4,0,1)(1,1,0)[7]                    : 5878.723
ARIMA(4,0,3)(1,1,0)[7]                    : Inf

Now re-fitting the best model(s) without approximations...

ARIMA(5,0,2)(1,1,0)[7]                    : 5916.123

Best model: ARIMA(5,0,2)(1,1,0)[7]
```

Appendix V: Summary Output of Proposed SARIMA Models

ARIMA(3,0,0)(1,1,0)

```
> summary(arima_modelA)
Series: uk_ts
ARIMA(3,0,0)(1,1,0)[7]

Coefficients:
         ar1     ar2     ar3     sar1
      0.5098  0.0691  0.2633  -0.2999
s.e.  0.0458  0.0522  0.0458   0.0463

sigma^2 estimated as 35652:  log likelihood=-2968.94
AIC=5947.88   AICc=5948.01   BIC=5968.38

Training set error measures:
                    ME     RMSE      MAE MPE MAPE      MASE        ACF1
Training set 0.4444999 186.5096 86.98201 NaN  Inf 0.6164004 0.006392149
> accuracy(arima_modelA)#arima(3,0,0)(1,1,0)[7]
                    ME     RMSE      MAE MPE MAPE      MASE        ACF1
Training set 0.4444999 186.5096 86.98201 NaN  Inf 0.6164004 0.006392149
```

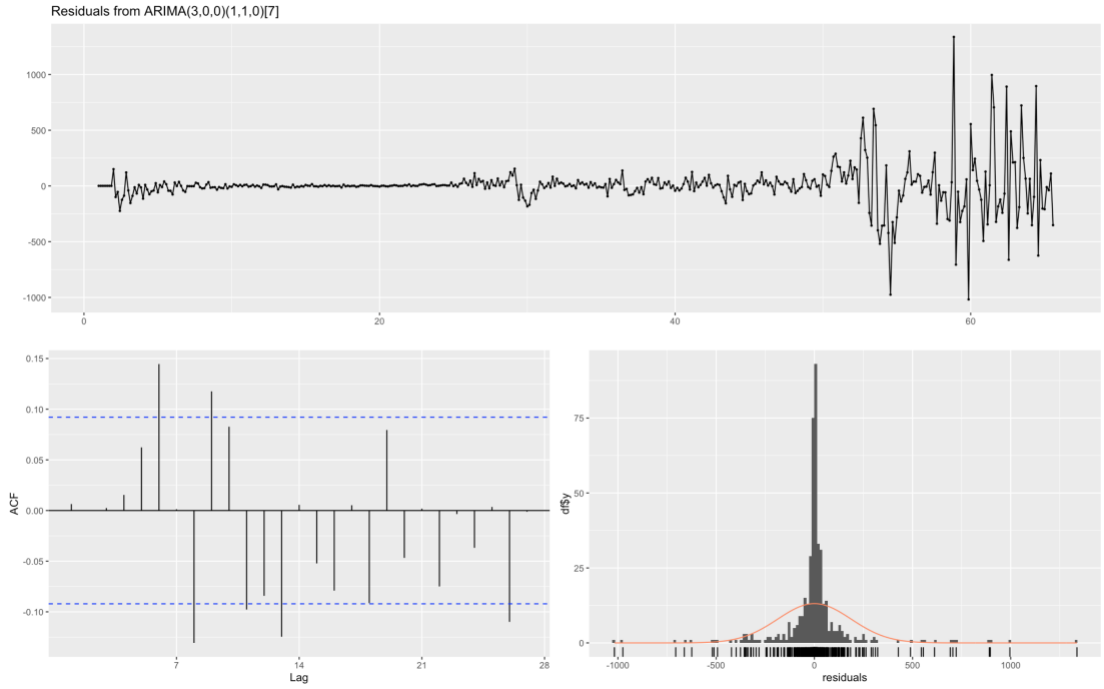
ARIMA(5,0,2)(1,1,0)

```
> summary(arima_modelB)
Series: uk_ts
ARIMA(5,0,2)(1,1,0)[7]

Coefficients:
         ar1      ar2     ar3      ar4     ar5     ma1     ma2     sar1
      0.3907  -0.7583  0.7531  -0.0406  0.3156  0.1512  0.9588  -0.2313
s.e.  0.0480   0.0528  0.0510   0.0515  0.0506  0.0203  0.0180   0.0608

sigma^2 estimated as 32734:  log likelihood=-2948.85
AIC=5915.71   AICc=5916.12   BIC=5952.61

Training set error measures:
                     ME     RMSE      MAE MPE MAPE      MASE         ACF1
Training set 0.5569243 177.9051 87.38254 NaN  Inf 0.6192388 -0.02494337
> accuracy(arima_modelB)#arima(5,0,2)(1,1,0)[7]
                     ME     RMSE      MAE MPE MAPE      MASE         ACF1
Training set 0.5569243 177.9051 87.38254 NaN  Inf 0.6192388 -0.02494337
```

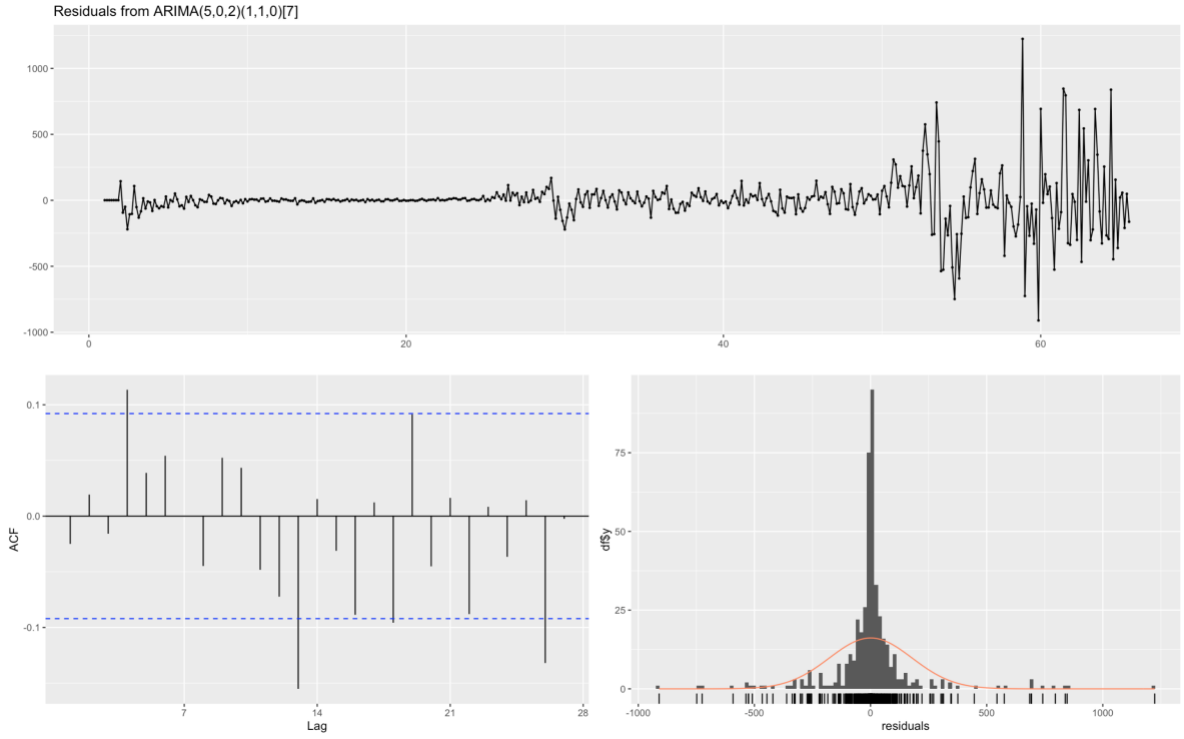
Appendix VI: Adequacy of Proposed SARIMA Models

```r
#check adequacy of each model
library(lmtest)
checkresiduals(arima_modelA)
checkresiduals(arima_modelB)
```

```
> checkresiduals(arima_modelA)

        Ljung-Box test

data:  Residuals from ARIMA(3,0,0)(1,1,0)[7]
Q* = 44.119, df = 10, p-value = 3.133e-06

Model df: 4.   Total lags used: 14
```




```
> checkresiduals(arima_modelB)

        Ljung-Box test

data:  Residuals from ARIMA(5,0,2)(1,1,0)[7]
Q* = 26.519, df = 6, p-value = 0.0001781

Model df: 8.   Total lags used: 14
```


Residuals from ARIMA(5,0,2)(1,1,0)[7]
1000
500
0
-500
-1000
0
20
40
60
0.1
0.0
-0.1
ACF
7
14
21
28
Lag
75
50
25
0
df$y
-1000
-500
0
500
1000
residuals

Appendix VII: Significant of Parameter Coefficients of Proposed SARIMA

```r
#check significant of coefficients
coeftest(arima_modelA) #only three variables are significant, ar2 is not significant
coeftest(arima_modelB) #ar4 not significant
```

```
> #check significant of coefficients
> coeftest(arima_modelA) #only three variables are significant, ar2 is not significant

z test of coefficients:

      Estimate Std. Error z value  Pr(>|z|)
ar1   0.509785   0.045820 11.1258 < 2.2e-16 ***
ar2   0.069124   0.052244  1.3231    0.1858
ar3   0.263298   0.045775  5.7520 8.817e-09 ***
sar1 -0.299880   0.046274 -6.4805 9.141e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> coeftest(arima_modelB) #ar4 not significant

z test of coefficients:

      Estimate Std. Error  z value  Pr(>|z|)
ar1   0.390730   0.047994   8.1412 3.913e-16 ***
ar2  -0.758252   0.052807 -14.3590 < 2.2e-16 ***
ar3   0.753145   0.051037  14.7569 < 2.2e-16 ***
ar4  -0.040579   0.051469  -0.7884 0.4304510
ar5   0.315634   0.050577   6.2407 4.356e-10 ***
ma1   0.151157   0.020295   7.4482 9.466e-14 ***
ma2   0.958764   0.018035  53.1621 < 2.2e-16 ***
sar1 -0.231293   0.060755  -3.8070 0.0001407 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Appendix VIII: Performance Evaluation using Forecast Errors for Proposed SARIMA Models

```
accuracy(arima_modelA)#arima(3,0,0)(1,1,0)[7]
accuracy(arima_modelB)#arima(5,0,2)(1,1,0)[7]
```

```
> accuracy(arima_modelA)#arima(3,0,0)(1,1,0)[7]
                     ME     RMSE      MAE MPE MAPE      MASE         ACF1
Training set 0.4444999 186.5096 86.98201 NaN  Inf 0.6164004 0.006392149
> accuracy(arima_modelB)#arima(5,0,2)(1,1,0)[7]
                     ME     RMSE      MAE MPE MAPE      MASE        ACF1
Training set 0.5569243 177.9051 87.38254 NaN  Inf 0.6192388 -0.02494337
```

Appendix IX: Forecasting with optimum ARIMA or SARIMA model

```
arima_forecastB = forecast(arima_modelB, h = 5) ; arima_forecastB
plot(arima_forecastB)
lines(fitted(arima_modelB), col='red', lwd=2)
legend('topleft', c("Historical data", "Trained Data", "Forecasted Data"), col=c("black",'red','blue'),
       pch=15, y.intersp=0.6, x.intersp=0.5, bty='n')
```

```
> arima_forecastB
         Point Forecast     Lo 80     Hi 80     Lo 95     Hi 95
65.71429      1320.8082 1088.9424 1552.6740  966.2001 1675.4163
65.85714      1464.6991 1200.9788 1728.4195 1061.3736 1868.0246
66.00000      1095.1863  814.6780 1375.6947  666.1858 1524.1869
66.14286      -147.5308 -451.3485  156.2868 -612.1799  317.1182
66.28571       -57.5292 -366.8980  251.8396 -530.6680  415.6096
```