Predicting Patients: Using Time Series Analysis and Forecasting to Predict Patient Numbers at a Surgery.

by A Smith

Introduction

For this coursework, we were given a time series dataset containing the daily number of patients who visited a specialist surgery from April 2015 to March 2019. We will refer to this dataset as *patient numbers*. We were tasked with fitting various models to explain the data, and asked to offer a prediction of at least 7 days to the management team of the surgery in order to estimate the number of patients who will visit the surgery in the future.

For this problem, I plotted the time series which we were given and performed some rudimentary analysis using *R*, and decomposed the data to examine the underlying trend, seasonality and error. I then decided to experiment on the data by fitting different time series models to explain the series and analysed the effectiveness of each one based on a training and test set. These models include a baseline (*naïve*) model, extrapolation models (*Single Exponential Smoothing (SES), Holt Linear and Holt-Winters*), simple and multiple linear regression, as well as *ARIMA*s. After describing each model, I will fit the model to our data, give the required forecast and summarise the error statistics for that model.

Numerical Summaries

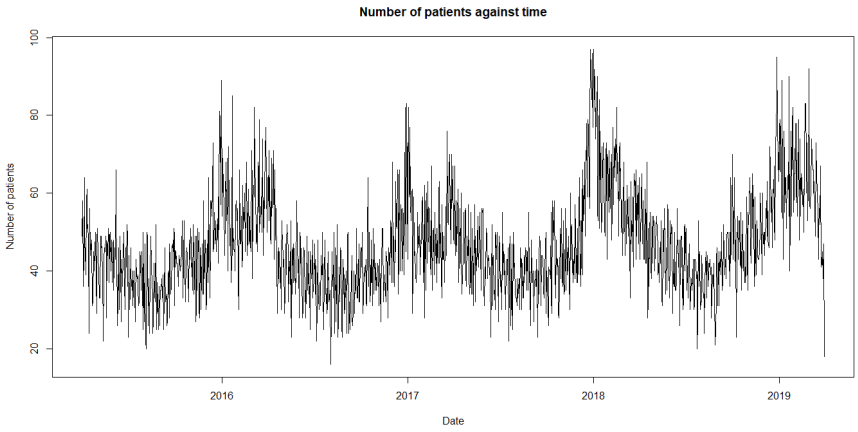
Our preliminary analysis on the time series will include finding the maximum and minimum values, the interquartile range, mean, median and standard deviation:

```
> summary(Patients)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  16.00   38.00   45.00   46.72   54.00   97.00
> sd(Patients)
[1] 12.9109
> IQR(Patients)
[1] 16
```

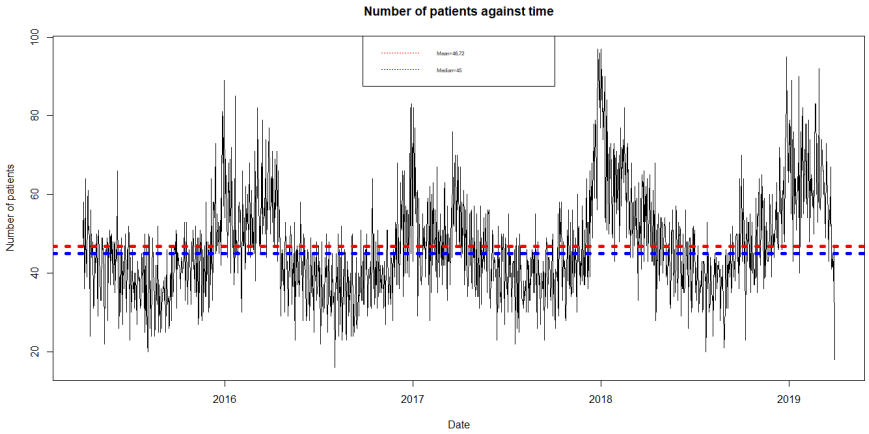
These summary statistics give us an indication of the spread of the data and the range of values it includes.

Graphical Summaries

It also helps to plot our data:

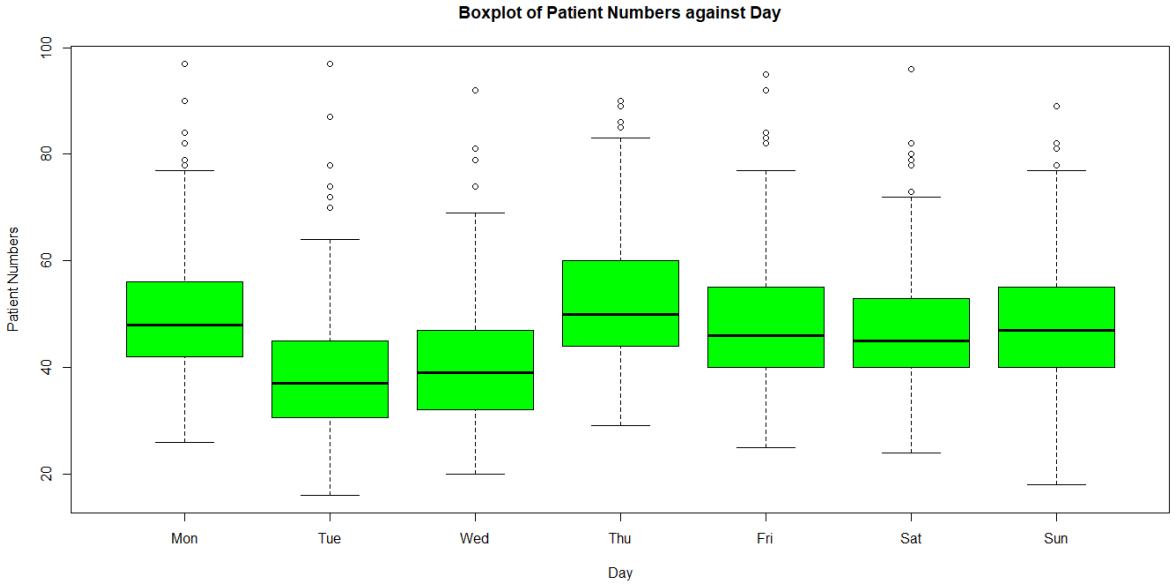


Plot of our dataset



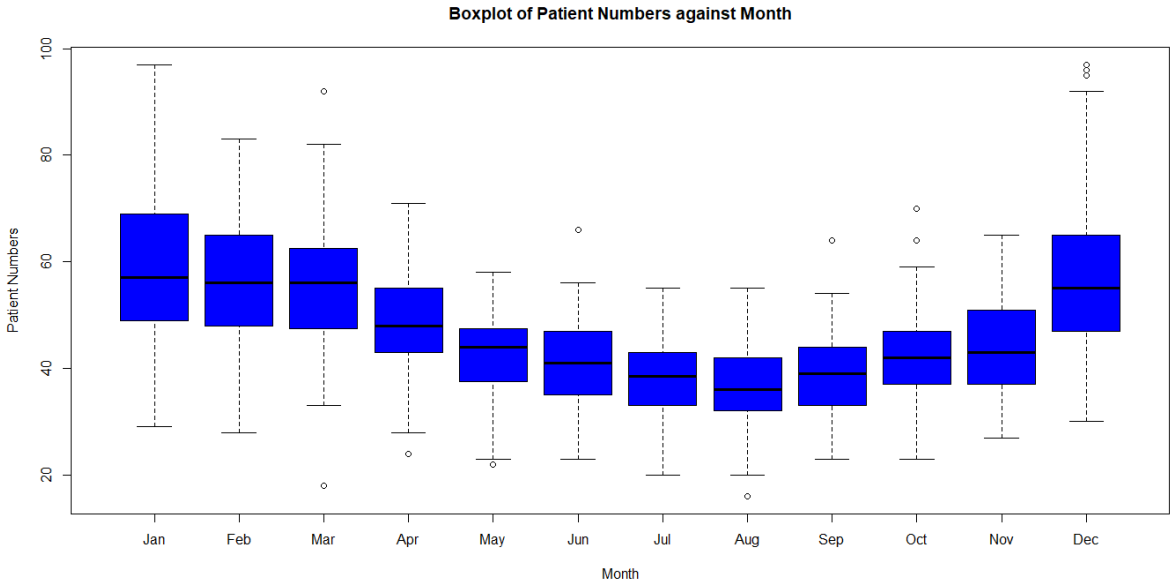
Plot of our dataset with red line representing overall mean (46.72485) and blue line representing overall median value (45).

From this plot, we begin to see that our data contains seasonality. Since we are examining data over a 4-year timeframe, we may break down our time series into smaller intervals of weeks, months and years. To examine our data's seasonality, we create boxplots of each of these intervals:



Boxplot of weekly variation

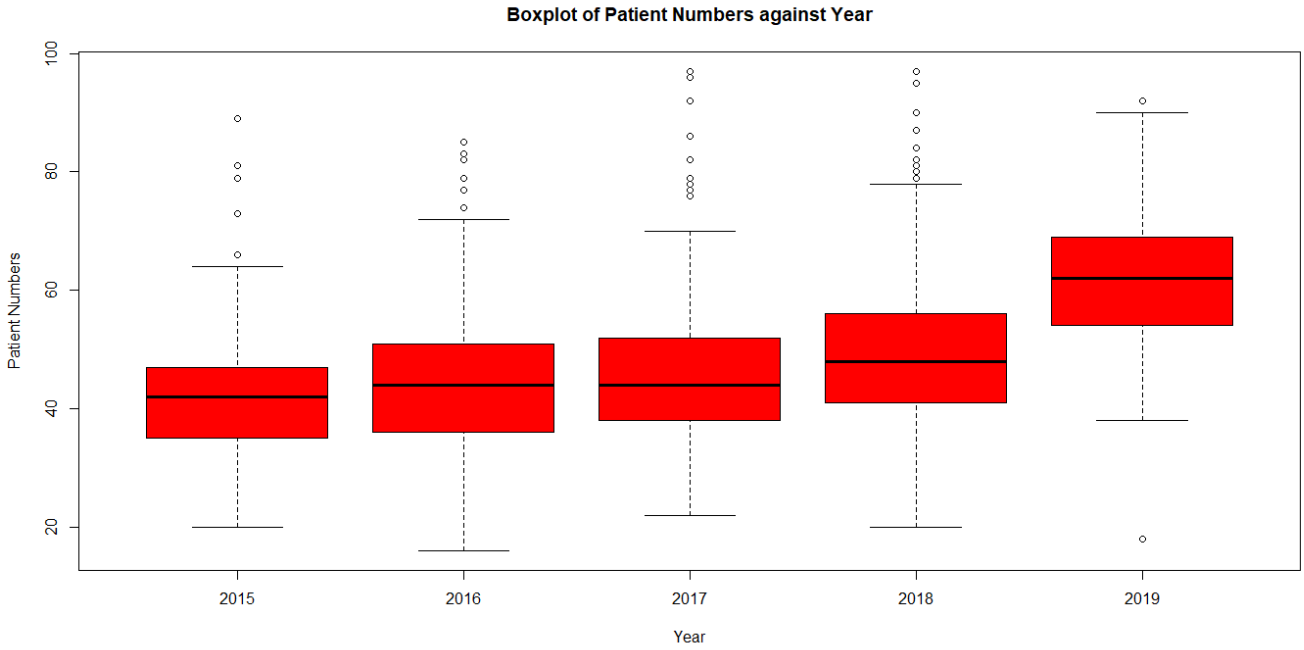
From the above boxplot, we notice that some days are, statistically, busier than others. The boxplot above suggests that the surgery has quieter days on Tuesdays and Wednesdays but is consistently busy across weekends and Mondays. Thursdays appear to be the busiest day of the week for the surgery as their mean and interquartile range is larger than all other days.



Perhaps the most drastic display of seasonality in our data comes from the analysis of patient numbers by month.

Aside from the odd extreme value (shown in our boxplot as single dots), the surgery seems busier in the winter and quieter in late spring and summer. We may theorise that the dots represent outliers (perhaps because there was a particular incident involving many people on these days).

We may also imagine that the numbers are highest through winter and early spring due to its coincidence with the regular flu season. We may also speculate that there are fewer seasonal illnesses in the summer and many people may be on holiday and therefore not using the surgery.

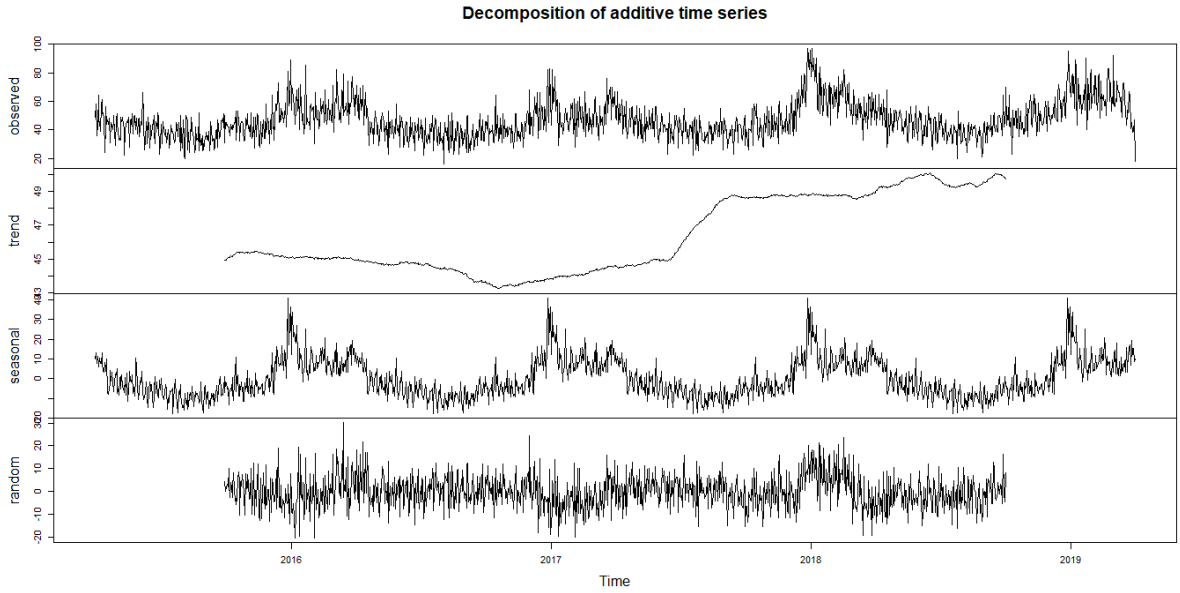


Next we examine the boxplot of the data broken down by year. This is a good opportunity to examine trend without performing a decomposition of the time series. We may be tempted to conclude from the above plot that there is a rising trend in the data. However this may be a premature conclusion when we consider that, for 2015, the data only includes *patient numbers* from April, and the data for 2019 only includes data up to March. This truncating of 2015 and 2019 may skew the plot, therefore we should not rely too much on the year boxplot to examine overall trend. We will shortly use the `decomposition` function in *R* to perform a decomposition and explore trend and seasonality.

Decomposition of the Data to Examine Trend, Seasonality and Error

When decomposing our time series, we must assess whether to use an *additive* or *multiplicative* decomposition. If the effects of the decomposition appear to be additive, we use *additive decomposition*. If, however, seasonality tends to increase as the mean increases, we favour *multiplicative decomposition*[1]. Since I could not see any drastic increase in seasonality throughout the data, I opted for *additive decomposition*.

[1] Henrik Madsen (2008) *Time Series Analysis*, Boca Raton, FL, USA: Taylor & Francis Group. p.60



From the above decomposition plot, there appears to be an underlying general downward trend, from mid-2015 until late 2016. There is also an underlying general upward trend from late 2016 until late 2018.

The seasonality of the data appears consistent across the years, further strengthening our earlier hypothesis that the surgery is consistently busy during flu season and quieter during the height of summer. As we may expect, the random error component of our decomposition plot appears randomly distributed with mean 0.

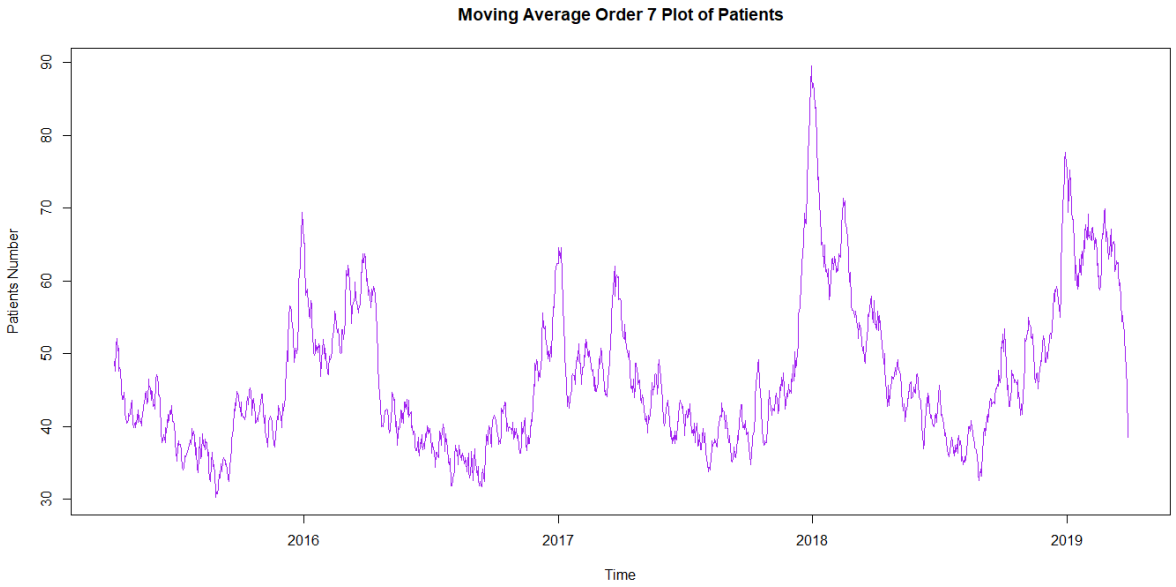
Moving Averages (MA)

Moving averages take the mean of a fixed subset and then shift forward to continue taking the mean of the next subset of the series. This has the effect of smoothing out fluctuations and extreme values and helps to smooth the data. For a time series $Y_1, Y_2, \ldots, Y_n$ we define the moving average of period k (MA(k)), we write the *simple moving average* formula as[2]:

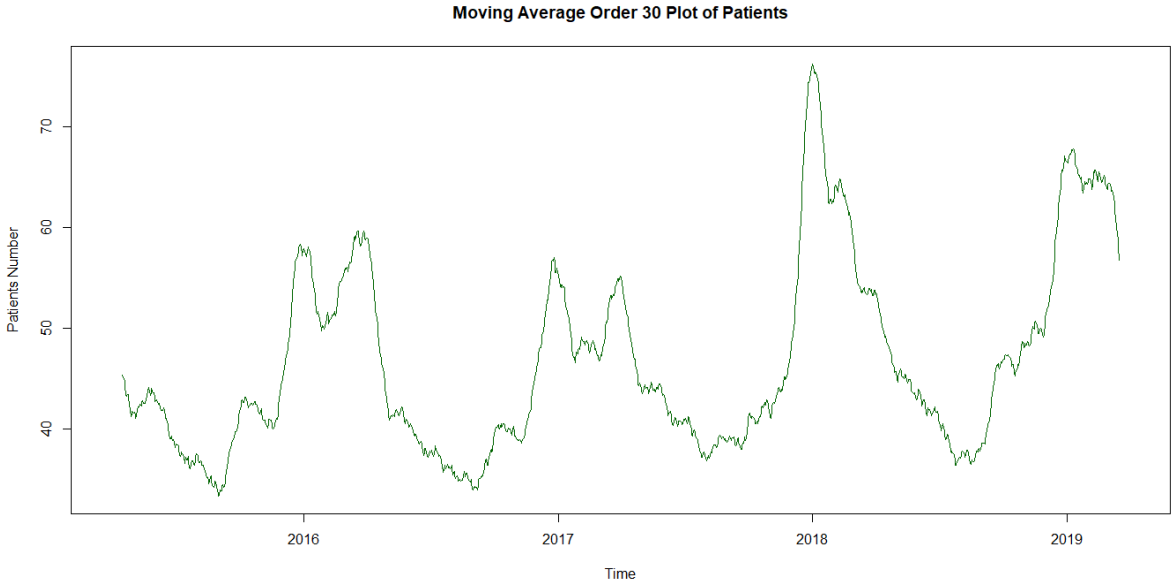
$$\frac{(Y_1 + Y_2 + \ldots + Y_k)}{k}, \frac{(Y_2 + \ldots + Y_{k+1})}{k}, \frac{(Y_3 + \ldots + Y_{k+2})}{k}, \ldots$$

We will examine a plot of MA(7), MA(30) and MA(90) models. These moving average approaches smooth out the weekly, monthly and season seasonality.

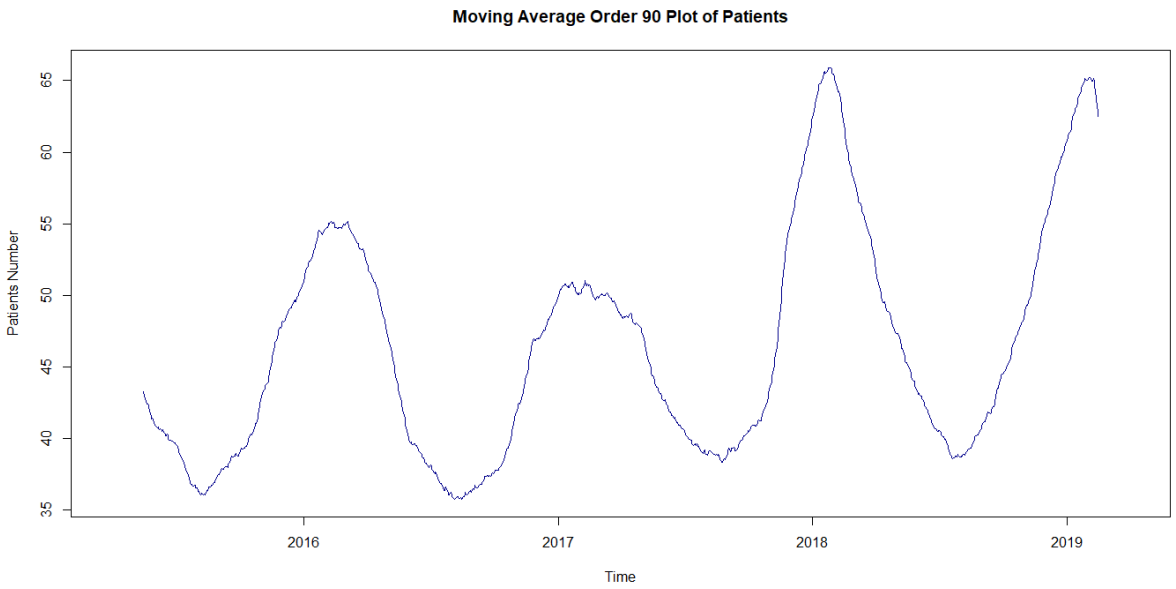
[2] P.N. Jani (2014) *Business Statistics: Theory and Applications*, New Delhi, India: PHI Learning. p.432



MA(7) plot of our patient data, with weekly variation smoothed.



MA(30) plot of our patient data, with within-month variation smoothed.



MA(90) plot of our patient data, with within-season variation smoothed. This is perhaps our clearest view of the effect of flu season on our data.

Stationarity

We may ask if our data is stationary. A stationary time series has a constant mean, variance and autocorrelation. If we can establish that our time series is stationary, it simplifies the prediction of future values, as we may imagine that predicted values will reflect previous observations. To assess whether our time series is stationary, we require the `adf.test` function in *R*, this function performs an *Augmented Dickey-Fuller Test* for stationarity and is available through the `tseries` package. We test the null hypothesis that the data is not stationary against the alternative hypothesis that the time series is stationary.

```
> adf.test(Patients)#low p-value suggests that data is stationary

        Augmented Dickey-Fuller Test

data:  Patients
Dickey-Fuller = -4.285, Lag order = 11, p-value = 0.01
alternative hypothesis: stationary
```

The extremely low p-value in our Augmented Dickey-Fuller Test provides strong statistical evidence to reject our null hypothesis and consider that our data is stationary. This is somewhat surprising given that our decomposition appeared to show an upward trend.

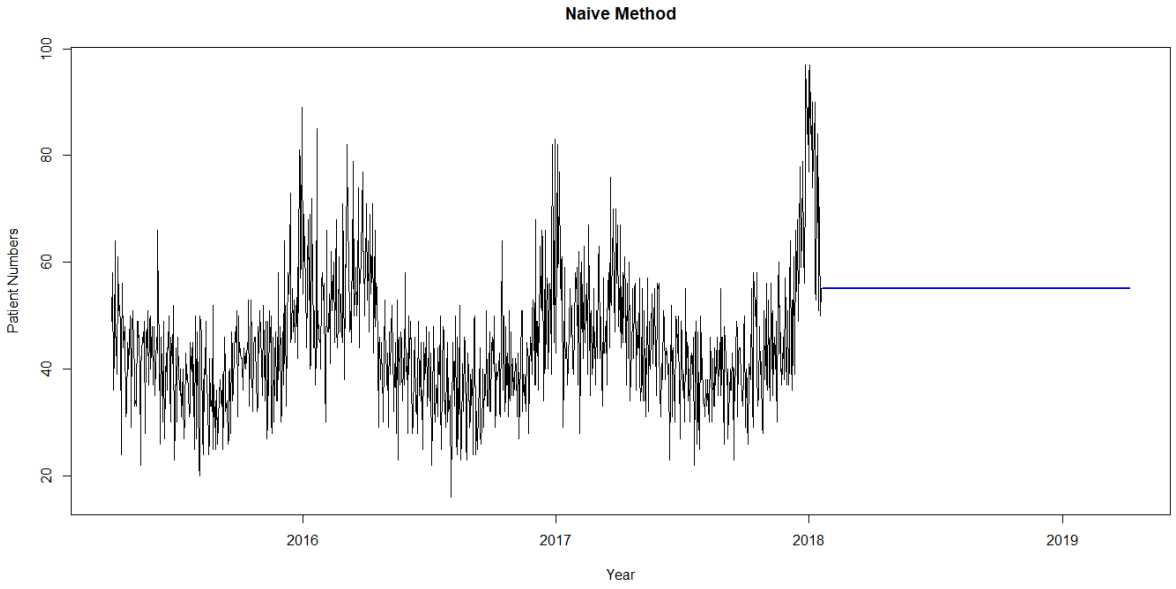
Baseline model (naïve)

Before performing any modelling, I partitioned the entire *patient numbers* dataset into a training set (70%) and a test set (30%) against which the effectiveness of the model would be measured.

Simple naïve model:

$$\hat{Y}_{t+1} = Y_t$$

The *naïve model* is the simplest form of model and is often used as a baseline with which to compare other models. The naïve model assumes that the forecast for time $t + 1$ is equal to the previous observed value. Using the 'forecast' package available in *R*, I plotted a naïve model to our *patient numbers* time series:



Plot of predicted values for naïve model

Our baseline (naïve) model predicts a constant number, repeating the value of the last observation. This value is 55 and will be a constant prediction:

```
> predict(NaiveForTestPatients,h=10)
          Point Forecast Lo FALSE Hi
2018.0493             55       55
2018.0521             55       55
2018.0548             55       55
2018.0575             55       55
2018.0603             55       55
2018.0630             55       55
2018.0658             55       55
2018.0685             55       55
2018.0712             55       55
2018.0740             55       55
```

The prediction for the next 10 days using the Naïve Model on the training patient numbers set. We note that the model predicts a constant 55 visitors to the surgery. We would expect, from this model, that the next 10 days into the test patient numbers set will be consistently 55.

It is perhaps worth noting that one weakness of the naïve model is that it's predicted values are dependent on the last observation which may vary. For example, when the naïve model is applied to the entire dataset:

```
> predict(NaiveEntirePatients,h=10)
          Point Forecast      Lo 80
2019.2493             18   2.518041
2019.2521             18  -3.894796
2019.2548             18  -8.815539
2019.2575             18 -12.963918
2019.2603             18 -16.618712
2019.2630             18 -19.922899
2019.2658             18 -22.961413
2019.2685             18 -25.789592
2019.2712             18 -28.445876
2019.2740             18 -30.958252
```

We read the 'Point Forecast' column for our prediction. Here the model predicts a consistent 18, due to the last value of the entire patient numbers dataset. It is worth noting that this is an extreme value and therefore this is probably a weak prediction.

Sources of error

Here we discuss two metrics to measure the error of a model. We will use the mean squared error (MSE) value given by[3]:

$$MSE = \frac{1}{n}\sum(y - \hat{y})^2$$

where n is the number of datapoints in our series, and $(y - \hat{y})^2$ represents the square of the difference between our observed value y and our predicted value $\hat{y}$

and mean absolute percentage error (MAPE) value, given by[3]:

$$MAPE = \frac{100\%}{n}\sum\left|\frac{y - \hat{y}}{y}\right|$$

We may calculate these values using the `accuracy` function available through the '*forecast*' package in *R*. We examine the naïve model presented earlier:

```
> accuracy(NaiveForTestPatients)
                     ME     RMSE      MAE       MPE     MAPE      MASE       ACF1
Training set 0.005870841 12.04631 9.547945 -3.864688 22.51347 0.9755509 -0.4273527
```

R returns the RMSE, this is the *square root* of the MSE. Therefore we find:

$$MSE = 12.04631^2 = 145.113584$$

MAPE is given in the *R* return as

$$MAPE = 22.51347$$

Since these are measures of error, we wish to choose a model which minimises these values. We continue our search for a model by discussing *seasonal naïve models*.

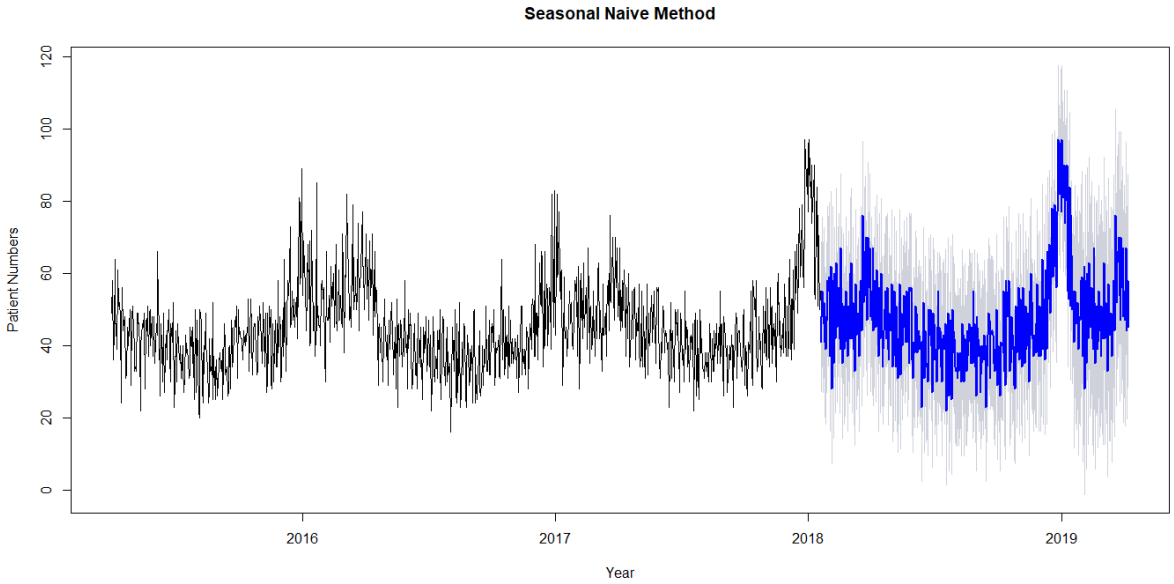
---

[3] DataQuest, *Tutorial: Understanding Regression Error Metrics in Python*, : https://www.dataquest.io/blog/understanding-regression-error-metrics/. Accessed 11th March 2020

Seasonal naïve model:

$$\hat{Y}_{t+1} = Y_{t-k} \text{ for seasonal lag } k$$

The seasonal naïve model is like the naïve model; however, it accounts for seasonality by basing its prediction on the previous observation of the same season. We examine the prediction made by the seasonal naïve model on our training dataset:



Seasonal naïve model predicted values, with the blue section representing the predicted values, the lighter colouring around the prediction represents the bounds for the 90% confidence intervals.

We next check the error scores for this method:

```
> accuracy(SNaiveForTestPatients)
                     ME     RMSE      MAE       MPE     MAPE MASE       ACF1
Training set 1.045593 12.68834 9.787234 -1.967742 22.6792    1 0.07070351
```

$$MSE = 12.68834^2 = 160.993972$$
$$MAPE = 22.6792$$

Interestingly, despite appearing like a more intricate model, the seasonal naïve approach scored a higher error value than the baseline naïve method.

The predicted values for the next 10 days from this method based on the entire *patient numbers* dataset:

```
> predict(SNaiveEntirePatients,h=10)
          Point Forecast    Lo 80
2019.2493             64 47.39876 80.
2019.2521             43 26.39876 59.
2019.2548             52 35.39876 68.
2019.2575             53 36.39876 69.
2019.2603             58 41.39876 74.
2019.2630             65 48.39876 81.
2019.2658             47 30.39876 63.
2019.2685             54 37.39876 70.
2019.2712             58 41.39876 74.
2019.2740             43 26.39876 59.
```

The prediction made by the seasonal naïve model. While this model has larger error values, it does at least account for the variability in patient numbers we would probably see in the coming days.

Extrapolation models

Next we will explore the effectiveness of fitting 3 extrapolation models to our data: *Single Exponential Smoothing* (SES), *Holt Linear*, and *Holt-Winters* models. An extrapolation model, in general, takes the form:

$$\hat{Y}_{t+1} = f(Y_t, Y_{t-1}, Y_{t-2}, \ldots)$$

That is, our predicted value $\hat{Y}_{t+1}$ is produced by performing a function on all preceding observations. We choose our function f according to the model being used[4].

Single Exponential Smoothing (SES)

We will first examine the effectiveness of plotting a single exponential smoothing (SES) model, this method was first proposed by Robert Goodell Brown.

Single exponential smoothing models take the form:

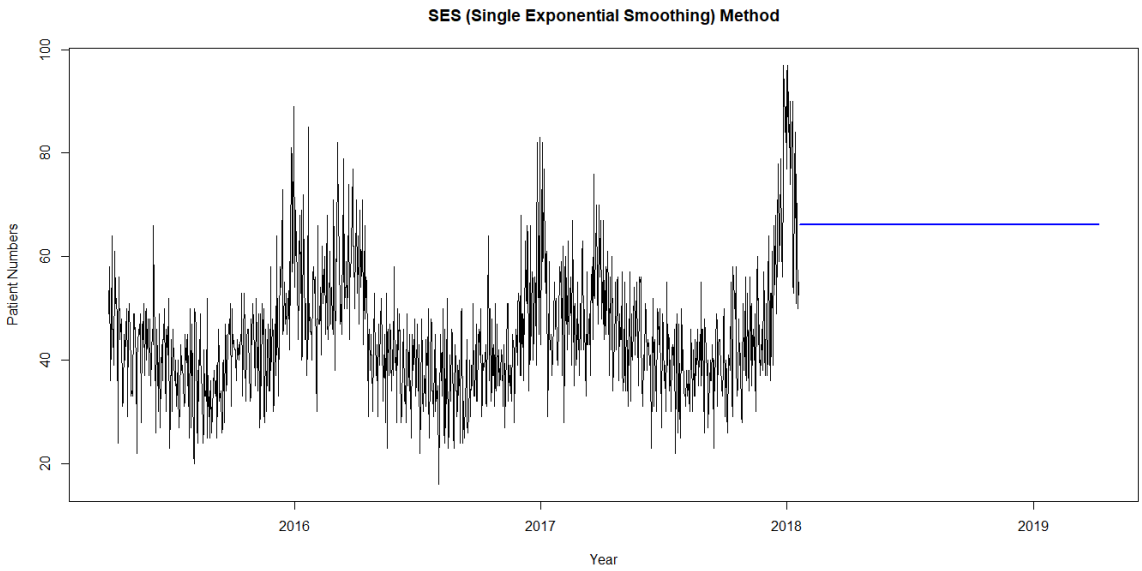
$$\hat{Y}_{t+1} = \hat{Y}_t + \alpha(Y_t - \hat{Y}_t)$$

That is, our predicted value $\hat{Y}_{t+1}$ represents the previous period's predicted value plus a smoothing parameter for the level (α), based on the error in the previous prediction, where $0 \leq \alpha \leq 1$ [5]. Note that, by default, *R* will estimate the value of α, otherwise we can specify it as an argument in the function.

We now perform the SES method on our *patient numbers* training dataset:

[4] Cliff T. Ragsdale (2008) *Spreadsheet Modeling & Decision Analysis*, 5th edn., Mason, OH, USA: Thomson Higher Education. p.486

[5] Cliff T. Ragsdale (2008), p.495



SES predicted values. We see again that this method predicts a constant number of patients for the future.

We again use the *accuracy* diagnostic in *R* to check the suitability of the model:

```
> accuracy(SESForTestPatients)
                    ME     RMSE      MAE       MPE     MAPE      MASE          ACF1
Training set 0.1074682 9.214965 7.297629 -3.923907 17.43325 0.7456273 -0.0006676631
```

$$MSE = 9.214965^2 = 84.915580$$
$$MAPE = 17.43325$$

These values are the lowest error statistics we have calculated so at the moment we prefer this model.

The predicted values for the next 10 days from this method based on the entire *patient numbers* dataset:

```
> predict(SESEntirePatients,h=10)
          Point Forecast    Lo 80    Hi 80
2019.2493       43.39404 31.46238 55.32570
2019.2521       43.39404 31.31102 55.47706
2019.2548       43.39404 31.16153 55.62655
2019.2575       43.39404 31.01385 55.77423
2019.2603       43.39404 30.86791 55.92017
2019.2630       43.39404 30.72365 56.06443
2019.2658       43.39404 30.58101 56.20707
2019.2685       43.39404 30.43995 56.34813
2019.2712       43.39404 30.30040 56.48768
2019.2740       43.39404 30.16233 56.62575
```

The prediction for the next 10 days using the SES method. This result is reminiscent of the results we predicted from the naïve method, although these seem like more realistic values.

Holt Linear model

Following our examination of single exponential smoothing, we now discuss the Holt Linear method proposed by Charles C. Holt. This method is most useful for data which displays a linear trend. The method uses the value of the time series at time t to estimate the base level of the time series (E_t) and the trend per time period (T_t). The function for Holt's linear method is given by:

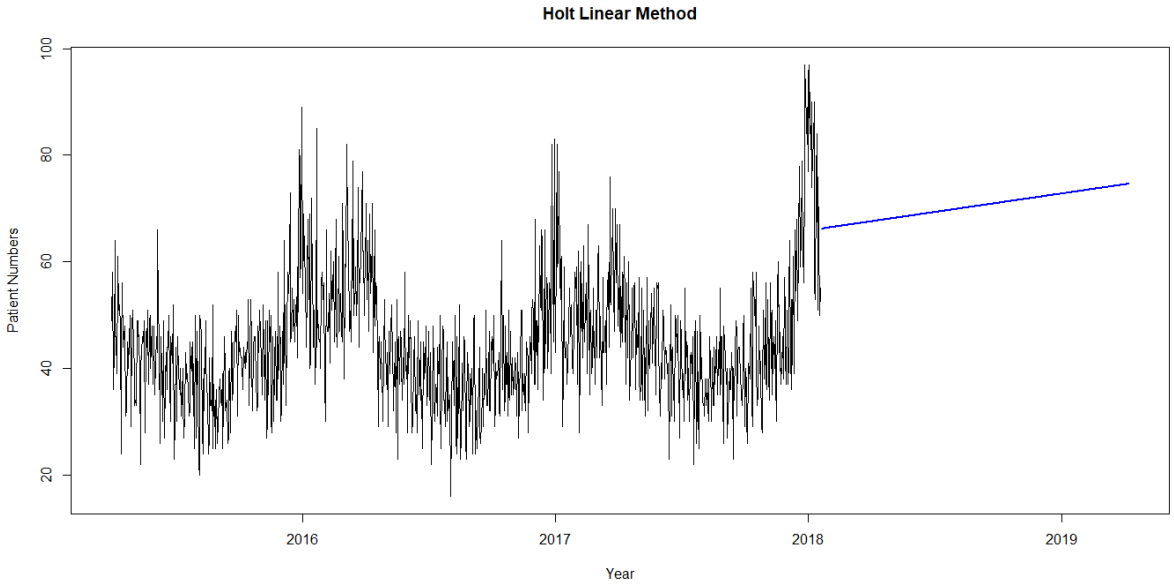
$$\hat{Y}_{t+n} = E_t + nT_t$$

Where[6]

$$E_t = \alpha Y_t + (1-\alpha)(E_{t-1} + T_{t-1})$$
$$T_t = \beta(E_t - E_{t-1}) + (1-\beta)T_{t-1}$$

These formulae give the forecast for n future time periods. Note that $n \in \mathbb{N}$. Our smoothing parameter for trend β, like α, is subject to the constraint $0 \leq \beta \leq 1$. Again, unless we specify a value for β as an argument, *R* will estimate a suitable value for our smoothing parameter β.

We now perform Holt's method on our *patient numbers* dataset:



Holt Linear predicted values. This method predicts a linear increase in the number of patients. It does not, however, seem to account for the seasonality in the data.

We perform a diagnostic check on the Holt Linear method:

```
> accuracy(HoltForTestPatients)
                     ME     RMSE      MAE       MPE     MAPE      MASE         ACF1
Training set 0.04306192 9.217996 7.304733 -4.062164 17.46929 0.7463532 -0.001216471
```

$$MSE = 9.217996^2 = 84.971450$$
$$MAPE = 17.46929$$

These values indicate that the error from the Holt Linear method is larger than the error from the SES method. Therefore, we prefer to use the SES model over the Holt Linear model.

[6] Cliff T. Ragsdale (2008), p.511

For completeness, we will compute the prediction for future values based on the Holt method:

```
> predict(HoltEntirePatients,h=10)
          Point Forecast    Lo 80    Hi 80
2019.2493       43.31440 31.37056 55.25825
2019.2521       43.30919 31.21132 55.40707
2019.2548       43.30398 31.05383 55.55414
2019.2575       43.29877 30.89803 55.69952
2019.2603       43.29357 30.74384 55.84329
2019.2630       43.28836 30.59123 55.98548
2019.2658       43.28315 30.44013 56.12617
2019.2685       43.27794 30.29049 56.26539
2019.2712       43.27273 30.14226 56.40320
2019.2740       43.26752 29.99540 56.53964
```

Interestingly, when extrapolated to the entire dataset, Holt's linear method predicts a slightly decreasing trend.

Holt-Winters model

The Holt-Winters method is an extension of Holt's Linear method but altered to allow for seasonality. The forecasting function is:

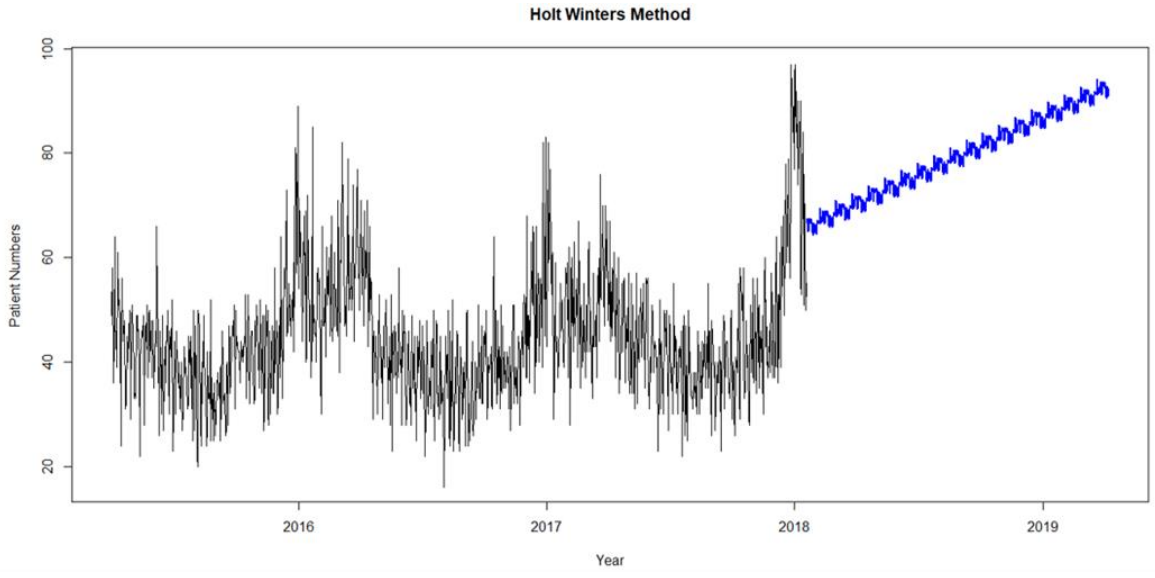
$$\hat{Y}_{t+n} = E_t + nT_t + S_{t+n-p}$$

Where E_t and T_t are calculated using the previous formulae. Again, we take n future time periods, and let p represent the number of seasons in our data. The formula for S_t is given by:

$$S_t = \gamma(Y_t - E_t) + (1 - \gamma)S_{\mathrm{t-p}}$$

Where γ represents the value of the smoothing parameter for the seasonal component. In my calculations I have left this argument empty, so the value for γ is estimated by *R*.

We now perform the Holt-Winters on our *patient numbers* dataset:



The prediction of future values resulting from the Holt-Winters method. The prediction values do display some seasonality. Unfortunately, Holt-Winters is not very effective for time series with a high frequency. Hence the prediction shows a frequency of 24 and not 365.

As mentioned in the caption, the Holt-Winters method is not particularly effective for time series with a high frequency, such as data taken daily over a year. If our data was quarterly or monthly, the Holt-Winters method would be much more effective.

We assess the error statistics:

```
> accuracy(HoltWintersForTestPatients)
                     ME      RMSE       MAE        MPE     MAPE      MASE         ACF1
Training set 0.3752714 9.176351 7.290365 -3.117998 17.29955 0.6864349 -0.01067281
```

$$MSE = 9.176351^2 = 84.205418$$
$$MAPE = 17.29955$$

This method has the least error of any of the extrapolation methods we have discussed. Therefore, if we were looking to use an extrapolation model to fit the data, we would choose the Holt-Winters model. The 10-day prediction given by this model is:

```
> predict(HoltWintersEntirePatients,h=10)
         Point Forecast    Lo 80    Hi 80
2079.625       43.48422 31.47192 55.49653
2079.667       42.24932 30.07301 54.42563
2079.708       42.05598 29.71765 54.39430
2079.750       40.44827 27.94983 52.94671
2079.792       42.83909 30.18237 55.49580
2079.833       41.34777 28.53455 54.16099
2079.875       43.03147 30.06344 55.99950
2079.917       41.29148 28.17029 54.41267
2079.958       41.81918 28.54641 55.09196
2080.000       42.99496 29.57215 56.41778
```

The model predicts that the number of customers over the next week will be in the low 40s.

Regression

Aside from extrapolation models, we can also model the data using linear or multiple linear regression.

Linear regression

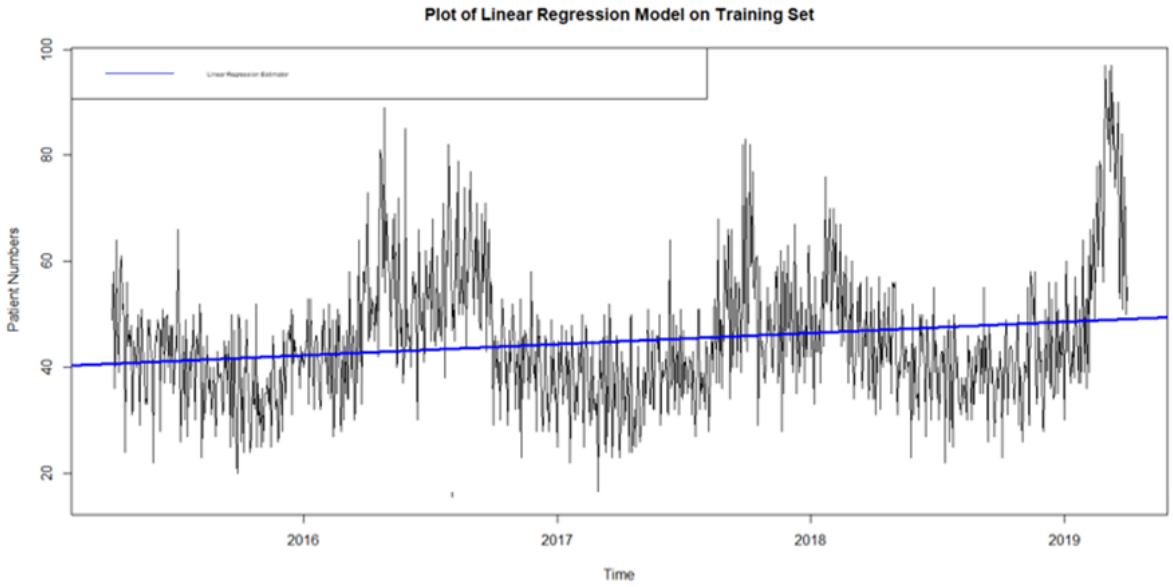
Linear regression functions may be generalised as having form:

$$y_t = \beta_0 + \beta_1 x_t + \epsilon_t$$

That is, y_t is equal to some constant β_0 plus some coefficient β_1 multiplied by a corresponding x_t value and added with a random error component ϵ_t[7]. We assume that the error terms ϵ_t are i.i.d with a normal distribution and have mean 0.

We fit a linear regression model to our data:

[7] Rob J Hyndman, George Athanasopoulos (2018) *Forecasting: principles and practice*, 2nd edn., Published online: OTexts. p.103

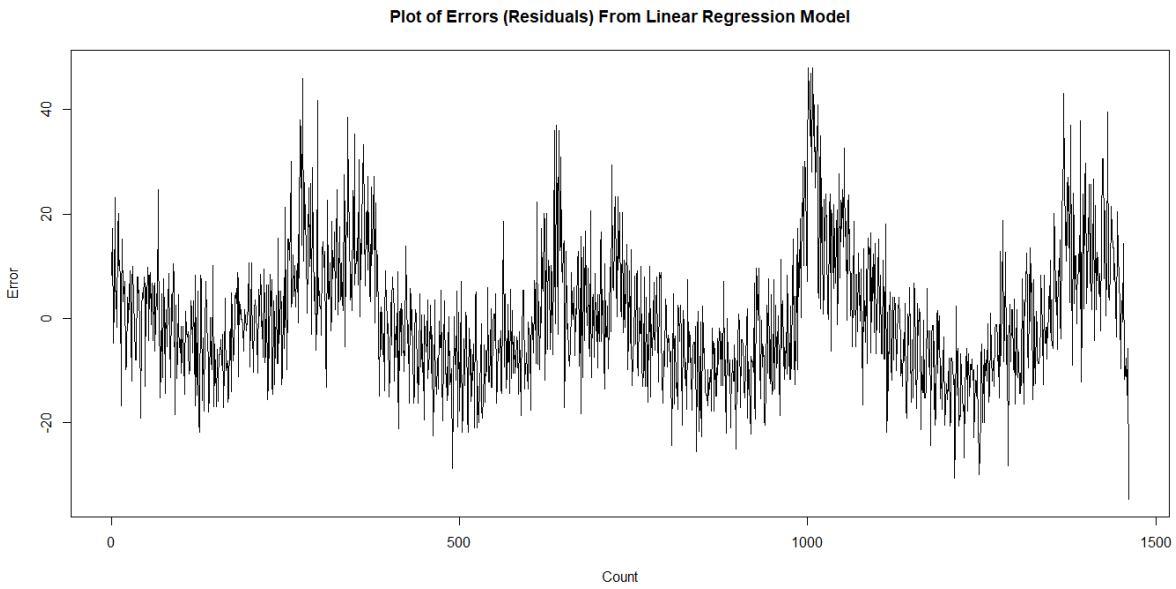


We see that the linear regression model follows a linearly increasing trend. The values given for β_0 and β_1 can be extracted from R:

```
> simple.fit = lm(Patients ~ c(1:length(Patients)))
> simple.fit

Call:
lm(formula = Patients ~ c(1:length(Patients)))

Coefficients:
          (Intercept)  c(1:length(Patients))
              40.7304                 0.0082
```

We see the intercept (β_0) and x coefficient (β_1) have values 40.7304 and 0.0082 respectively.

Plotting the residuals from the linear regression gives:



Plot of error

This distribution appears random, and has mean 5.44282×10^{-16}, effectively zero, as required by our assumptions on normally distributed errors with mean 0.

We assess the error statistics from the linear regression method:

```
> accuracy(simple.fit)
                     ME     RMSE      MAE       MPE     MAPE      MASE
Training set 5.784786e-16 12.43447 9.68752 -7.142806 22.32391 0.9744012
```

$$MSE = 12.43447^2 = 154.616044$$
$$MAPE = 22.32391$$

From these values, we realise that the linear regression model provides a worse fit than the best-performing extrapolation model. We nonetheless examine the predicted values from the linear regression model:

```
> predict(simple.fit$fitted.values,h=10)
     Point Forecast    Lo 80    Hi 80
1462       52.71928 52.71928 52.71928 52
1463       52.72748 52.72748 52.72748 52
1464       52.73568 52.73568 52.73568 52
1465       52.74388 52.74388 52.74388 52
1466       52.75208 52.75208 52.75208 52
1467       52.76028 52.76028 52.76028 52
1468       52.76848 52.76848 52.76848 52
1469       52.77668 52.77668 52.77668 52
1470       52.78488 52.78488 52.78488 52
1471       52.79308 52.79308 52.79308 52
```

The linear model predicts a constant 53 patients for the next 10 days.

Multiple linear regression

For multiple variables, we may use a more sophisticated method of regression, called *multiple linear regression*. The model takes general form:

$$y_t = \beta_0 + \beta_1 x_{1,t} + \beta_2 x_{2,t} + \ldots + \beta_k x_{k,t} + \epsilon_t$$

[8] for k predictor variables. To perform multiple linear regression, I added three new variables to the dataset, corresponding to the day of the week, month of the year and the year of each observation.

Fitting this model to our data gave the following output:

```
Residuals:
    Min      1Q  Median      3Q     Max
-44.033  -5.446  -0.319   5.074  47.604

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         60.05233    1.16662  51.475  < 2e-16 ***
DayFr               -2.59553    0.85365  -3.040 0.002405 **
DaySa               -3.46166    0.85363  -4.055 5.28e-05 ***
DaySu               -2.35445    0.85354  -2.758 0.005881 **
DayTh                1.88087    0.85364   2.203 0.027727 *
DayTu              -11.47329    0.85455 -13.426  < 2e-16 ***
DayWe               -9.52950    0.85357 -11.164  < 2e-16 ***
MonthApr            -9.23990    1.15700  -7.986 2.83e-15 ***
MonthAug           -21.32005    1.14806 -18.570  < 2e-16 ***
MonthDec            -0.36568    1.14817  -0.318 0.750157
MonthFeb            -2.91265    1.13333  -2.570 0.010270 *
MonthJul           -19.71428    1.14813 -17.171  < 2e-16 ***
MonthJun           -16.93975    1.15699 -14.641  < 2e-16 ***
MonthMar            -4.19686    1.10686  -3.792 0.000156 ***
MonthMay           -14.41538    1.14803 -12.557  < 2e-16 ***
MonthNov           -13.70786    1.15694 -11.848  < 2e-16 ***
MonthOct           -15.21813    1.14807 -13.255  < 2e-16 ***
MonthSep           -18.67561    1.15701 -16.141  < 2e-16 ***
YearTwentyEighteen   4.88557    0.71190   6.863 1.00e-11 ***
YearTwentyNineteen   8.53204    1.22153   6.985 4.35e-12 ***
YearTwentySeventeen  1.18244    0.71190   1.661 0.096940 .
YearTwentySixteen   -0.07062    0.71177  -0.099 0.920977
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 8.714 on 1439 degrees of freedom
Multiple R-squared:  0.551,     Adjusted R-squared:  0.5445
F-statistic:  84.1 on 21 and 1439 DF,  p-value: < 2.2e-16
```

The output gives our value for β_0 as 60.05233. We note that *R* takes the first value alphabetically for each variable and compares coefficients for the variable against it. I put an asterisk (*) in front of *Monday, January* and *TwentyFifteen* so that these would be the base values for each of the *day, month and year* variables. Therefore, all coefficients we see are calculated in relation to these base values.

```
> accuracy(multreg)
                      ME     RMSE      MAE       MPE     MAPE      MASE
Training set 8.1551e-16 8.648074 6.581149 -3.265133 14.67293 0.6619526
```

$$MSE = 8.648074^2 = 74.789184$$
$$MAPE = 14.67293$$

These model error statistics are drastically lower than those from simple linear regression and marginally lower than any of the extrapolation models. Because of this low error value, we favour this approach at present. This model's prediction for the entire *patient numbers* dataset is:

[8] Rob J Hyndman, George Athanasopoulos (2018), p.106

```
> predict(multreg$fitted.values,h=10)
     Point Forecast    Lo 80    Hi 80
1462       60.58719 54.12560 67.04878
1463       60.66167 54.18486 67.13849
1464       60.72197 54.21701 67.22692
1465       60.77078 54.22448 67.31707
1466       60.81029 54.21034 67.41024
1467       60.84227 54.17781 67.50674
1468       60.86816 54.12996 67.60637
1469       60.88912 54.06961 67.70864
1470       60.90609 53.99920 67.81298
1471       60.91982 53.92082 67.91883
```

The prediction given by our multiple linear regression model; the model predicts a constant 61 patients over the next 10 days.

Auto Regressive Integrated Moving Average (ARIMA) models

An ARIMA model attempts to explain the time series based on its previous observations. The ARIMA model consists of three terms:

p represents the order of the Auto Regressive (AR) term
q represents the order of the Moving Average (MA) term
d represents the number of differencing required to achieve stationarity in the data[9]

The formula for an Auto Regressive Moving Average (ARMA(p, q)) model is given by:

$$Y_t = c + \Phi_1 y_{t-1} + \ldots + \Phi_p y_{t-1} + \theta_1 \epsilon_{t-1} + \ldots + \theta_q \epsilon_{t-q} + \epsilon_t$$ [10]

Where $\Phi(y)$ is the characteristic polynomial for the AR component and $\theta(y)$ is the characteristic polynomial for the MA component. This formula becomes an ARIMA formula by performing differencing d times.

We note that:
ARIMA($p, 0, q$) = ARMA(p, q)
ARIMA($p, 0, 0$) = AR(p)
ARIMA($0, 0, q$) = MA(q)

We can select an ARIMA model based on examination of the plots of the ACF (Autocorrelation Function) and PACF (Partial Autocorrelation Function) of our time series. The characteristics of these plots help us determine which ARIMA model to fit. We imagine each of the following models to display the following properties:

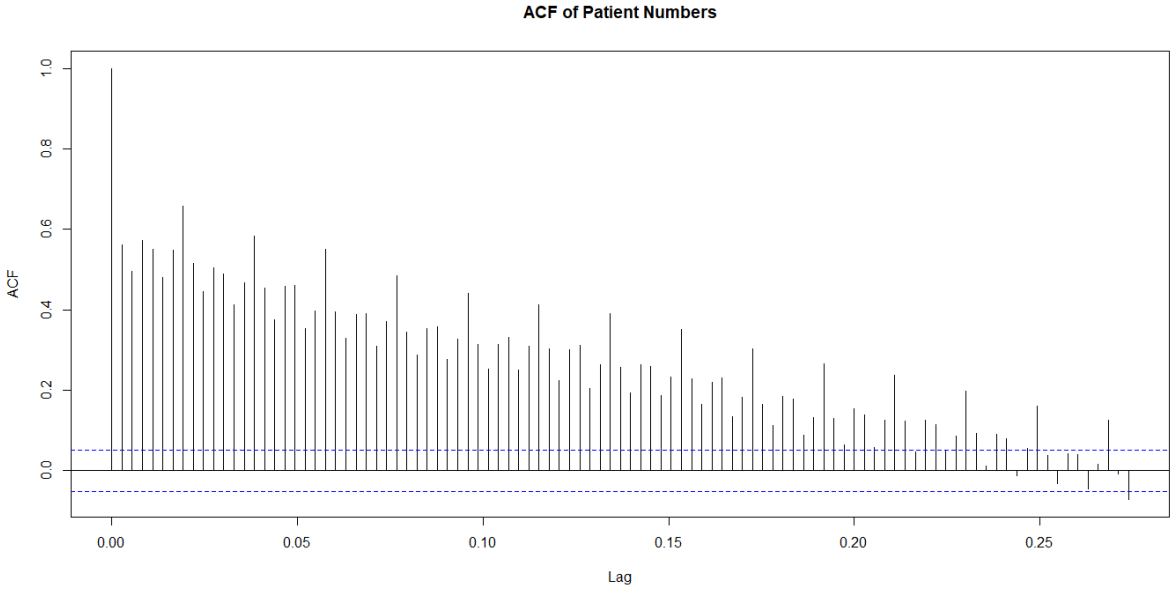
Model	**ACF**	**PACF**
AR(p)	Dies away geometrically	Zero after lag p
MA(q)	Zero after lag q	Dies away geometrically

[9] Selva Prabhakaran (2019) *ARIMA Model – Complete Guide to Time Series Forecasting in Python*, : https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/. Accessed 13th March 2020
[10] Rob J Hyndman (2014) *Forecasting: Principles & Practice*, : University of Western Australia, https://robjhyndman.com/uwafiles/fpp-notes.pdf. p.73. Accessed 13th March 2020

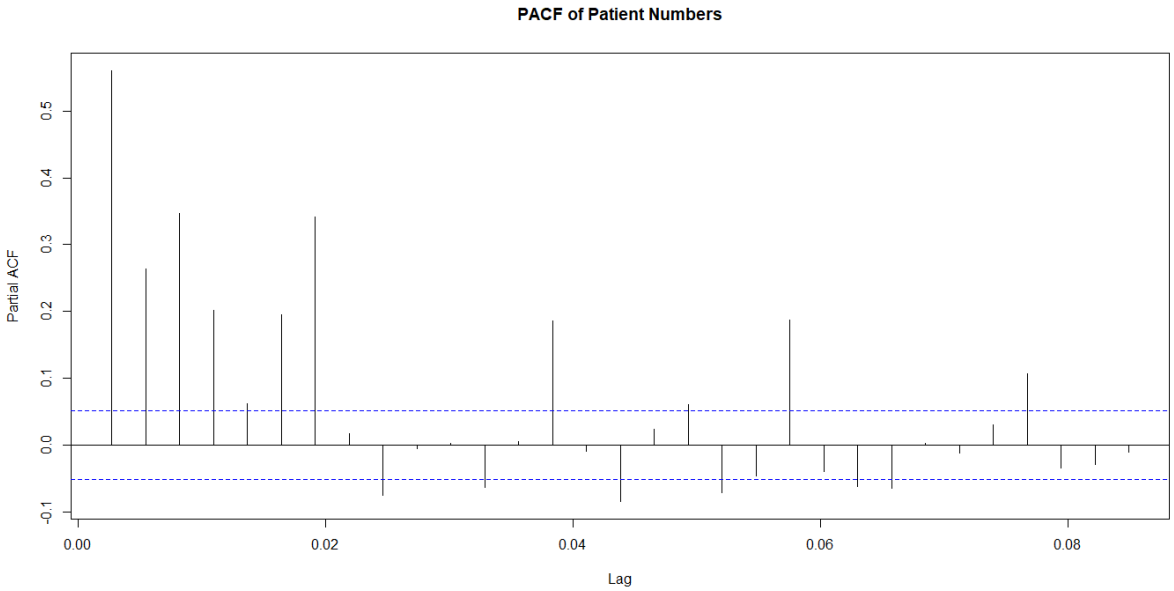
ARMA(p, q)	Dies away geometrically	Dies away geometrically

If the ACF dies away slowly, an ARIMA(p, d, q) model is preferred. We may also difference the data and examine the resulting ACF and PACF plots to assess if an ARMA(p, q) model can be fitted.

We explore the effect of fitting a basic ARIMA(1,1,1) model to our data. Firstly, we examine a plot of the autocorrelation function of our *patient numbers* time series.

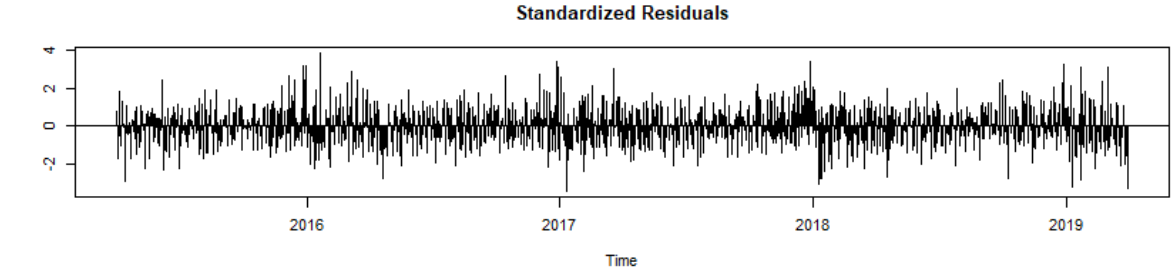


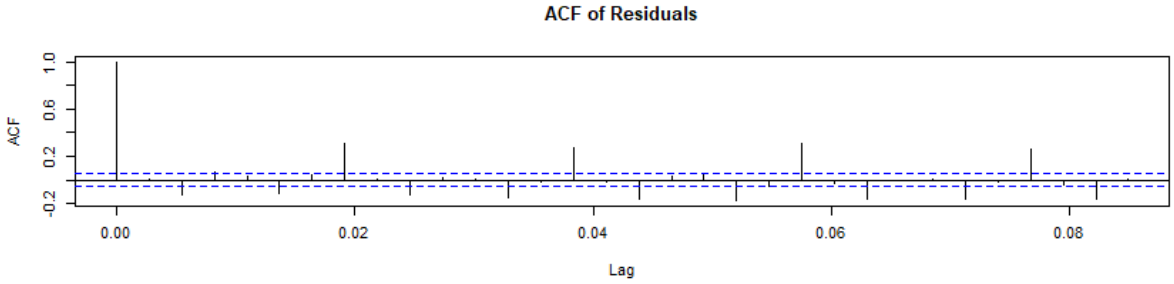
The ACF for our dataset dies away slowly therefore our time series requires differencing

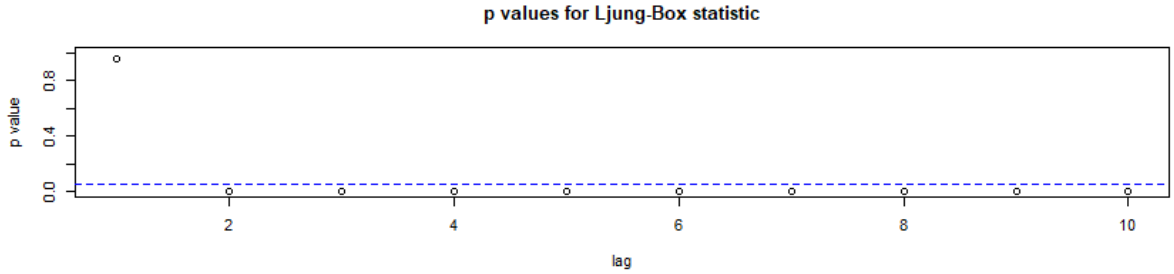


Plot of PACF on our patient numbers dataset.

These plots suggest that an ARIMA(1,1,1) model may not be suitable for modelling our time series. We may run the `tsdiag` function in *R*, which examines the suitability of a given ARIMA model. The function plots the standardised residuals, the ACF of the residuals and the *p*-values of the *Ljung-Box statistics*, a test statistic based on the values of the autocorrelations:







Here we see that many of the Ljung-Box statistic values are significant at the 5% level, this casts serious doubt about the adequacy of the ARIMA(1,1,1) model.

The above plot demonstrates that the residuals are i.i.d with mean zero as required. However, many of the ACF values fall outside the 5% significance level. The low values for the *Ljung-Box statistics* also cast serious doubt over the suitability of the ARIMA(1,1,1) model.

We examine the error statistics from the ARIMA(1,1,1) model:

```
> accuracy(Patientsar1i1ma1)
                     ME     RMSE      MAE       MPE     MAPE      MASE         ACF1
Training set -0.02520836 9.303378 7.382208 -4.003961 17.01431 0.7746729 0.001229048
```

$$MSE = 9.303378^2 = 86.552842$$
$$MAPE = 17.01431$$

The predicted values for our ARIMA(1,1,1) model are:

```
> predict(Patientsar1i1ma1$arma,h=10)
   Point Forecast     Lo 80    Hi 80
 8       52.57247 -140.8468 245.9918
 9       52.57247 -140.8468 245.9918
10       52.57247 -140.8468 245.9918
11       52.57247 -140.8468 245.9918
12       52.57247 -140.8468 245.9918
13       52.57247 -140.8468 245.9918
14       52.57247 -140.8468 245.9918
15       52.57247 -140.8468 245.9918
16       52.57247 -140.8468 245.9918
17       52.57247 -140.8468 245.9918
```

The prediction for the next 10 days is a constant 53 according to our ARIMA(1,1,1) model.

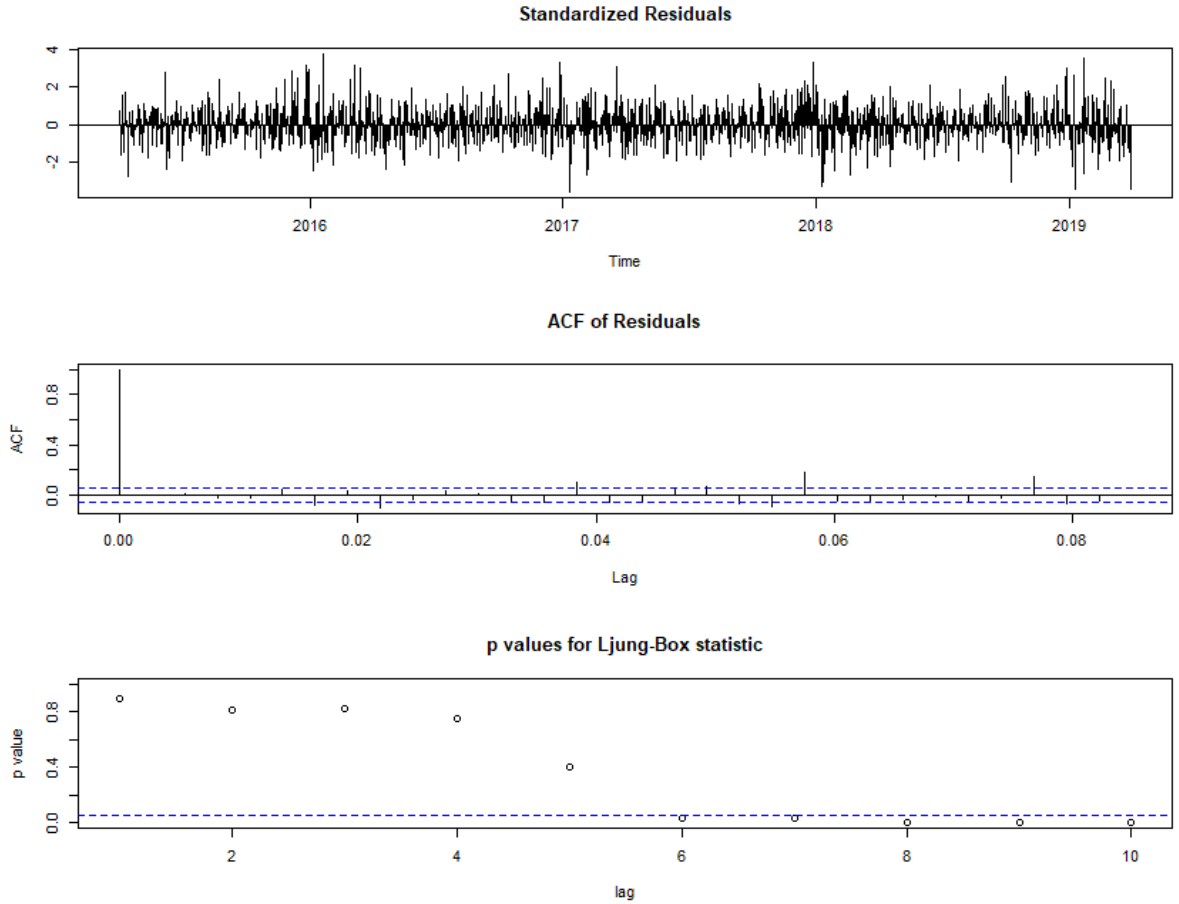
We may use the `auto.arima` function in *R* to calculate which ARIMA model should be used. The function finds the ARIMA with the lowest AIC (*Akaike Information Criterion)* value. We note that the function also considers seasonal ARIMAs of the form ARIMA$(p,d,q)(P,D,Q)$ where P,D,Q represent the additional seasonal component.

Running the `auto.arima` function on our *patient numbers* dataset, *R* recommends fitting an ARIMA(5,1,3) model to the time series, with no additional seasonal component.

```
 ARIMA(5,1,3)                    : 10512.04
 ARIMA(5,1,3)(1,0,0)[365]        : Inf
 ARIMA(5,1,3)(0,0,1)[365]        : Inf
 ARIMA(5,1,3)(1,0,1)[365]        : Inf
 ARIMA(4,1,3)                    : Inf
 ARIMA(5,1,2)                    : 10515.22
 ARIMA(5,1,4)                    : Inf
 ARIMA(4,1,2)                    : 10631.2
 ARIMA(4,1,4)                    : Inf

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

 ARIMA(5,1,3)                    : 10521.17

 Best model: ARIMA(5,1,3)

Series: Patients
ARIMA(5,1,3)

Coefficients:
         ar1      ar2      ar3      ar4      ar5      ma1     ma2      ma3
      0.2185  -0.6138  -0.2063  -0.2746  -0.3013  -1.0659  0.7179  -0.1315
s.e.  0.0971   0.0462   0.0445   0.0295   0.0361   0.1008  0.1049   0.0586

sigma^2 estimated as 78.28:  log likelihood=-5251.52
AIC=10521.04   AICc=10521.17   BIC=10568.62
```

The results of the `auto.arima` *function, recommending an ARIMA(5,1,3) model to fit to our data.*

We fit an ARIMA(5,1,3) model and assess the diagnostics and error statistics:



A better fit: fewer ACFs fall outside of the 5% significance level. Similarly, fewer Ljung-Box values are outside the 5% level.

Assessing the error statistics for the ARIMA(5,1,3) model:

```
> accuracy(Patientsar5i1ma3)
                     ME     RMSE      MAE       MPE     MAPE      MASE          ACF1
Training set -0.03930517 8.82018 6.908282 -3.499103 15.80766 0.7249401 -0.003337825
```

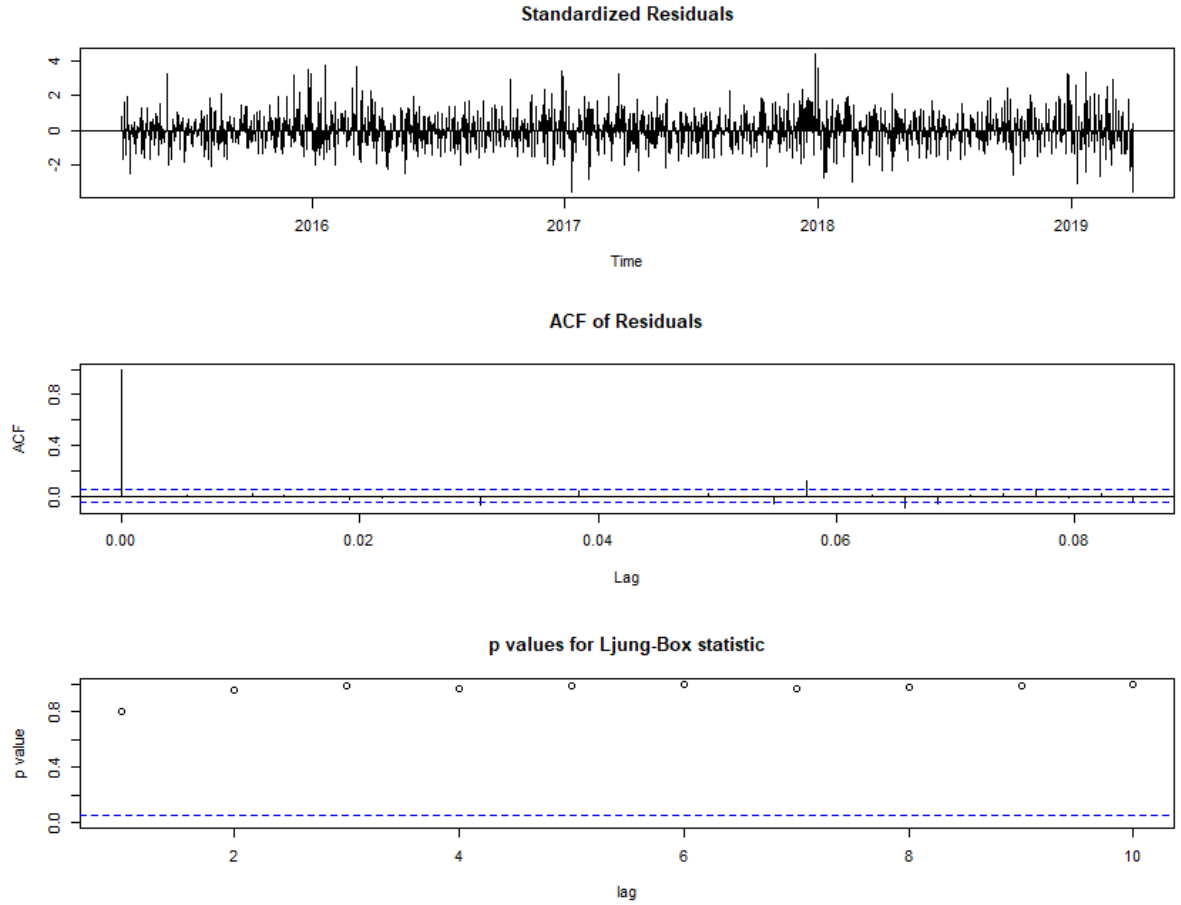
$$MSE = 8.82018^2 = 77.795575$$
$$MAPE = 15.80766$$

This model performs much better than the ARIMA(1,1,1) model and outperforms all the extrapolation models. The forecast for the next 10 days is:

```
> predict(Patientsar5i1ma3$arma,h=10)
   Point Forecast     Lo 80    Hi 80
 8       53.42458 -139.4811 246.3303
 9       53.42458 -139.4811 246.3303
10       53.42458 -139.4811 246.3303
11       53.42458 -139.4811 246.3303
12       53.42458 -139.4811 246.3303
13       53.42458 -139.4811 246.3303
14       53.42458 -139.4811 246.3303
15       53.42458 -139.4811 246.3303
16       53.42458 -139.4811 246.3303
17       53.42458 -139.4811 246.3303
```

The model predicts a consistent 53 for the next 10 days

When using ARIMA models, we must consider the *principle of parsimony.* That is, we must model the smallest possible number of parameters for adequate representation of the data. Otherwise, we run the danger of overfitting our model. However, for experimental purposes I have tried fitting an ARIMA(12,1,3) to the data to try and account for monthly variation.



An even better fit: Fewer ACFs fall outside of the 5% significance level. Also, no Ljung-Box values are below the 5% level.

We assess the accuracy statistics for our ARIMA(12,1,3) model:

```
> accuracy(Patientsarl2i1ma3)
                     ME     RMSE      MAE       MPE     MAPE      MASE          ACF1
Training set 0.1349055 8.19467 6.379177 -2.885963 14.54388 0.669417 -0.006344949
```

$$MSE = 8.19467^2 = 67.152616$$
$$MAPE = 14.54388$$

These values are our lowest error statistics yet and suggest that this is a better fit than all models we have previously seen. We next examine our 10-day forecast:

```
> predict(Patientsar12i1ma3$arma,h=10)
   Point Forecast     Lo 80    Hi 80
 8       54.40821 -137.9552 246.7716 -
 9       54.40821 -137.9552 246.7716 -
10       54.40821 -137.9552 246.7716 -
11       54.40821 -137.9552 246.7716 -
12       54.40821 -137.9552 246.7716 -
13       54.40821 -137.9552 246.7716 -
14       54.40821 -137.9552 246.7716 -
15       54.40821 -137.9552 246.7716 -
16       54.40821 -137.9552 246.7716 -
17       54.40821 -137.9552 246.7716 -
```

The model predicts a consistent 54 for the next 10 days

Summary of results

Next we summarise the different modelling methods, including their error statistics and their predictions. I will rank the effectiveness of each model based on the mean squared error. We note that the figures for the prediction have been rounded to the nearest integer value, which is why methods with different error values give the same prediction:

Model	**Mean Squared Error**	**Mean Absolute Error Percentage**	**Prediction for next 10 days**	**Ranking of effectiveness within model type**	**Overall ranking of effectiveness**
Baseline models:					
Naïve	145.11	22.51	Constant 18	1	8
Seasonal Naïve	160.99	22.68	64, 43, 52, 53, 58, 65, 47, 54, 58, 43	2	10
Extrapolation models:					
SES	84.92	17.43	Constant 43	2	5
Holt Linear	84.97	17.47	Constant 43	3	6
Holt-Winters	84.21	17.30	43, 42, 42, 40, 43, 41, 43, 41, 42, 43	1	4
Regression models:					
Simple Linear	154.62	22.32	Constant 53	2	9
Multiple Linear	74.79	14.67	Constant 61	1	**2**
ARIMAs					
ARIMA(1,1,1)	86.55	17.01	Constant 53	3	7
ARIMA(5,1,3)	77.80	15.81	Constant 53	2	**3**
ARIMA(12,1,3)	67.15	14.54	Constant 54	1	**1**

From the above summary table, we would recommend using an ARIMA(12,1,3) model, which would give the prediction for the next 10 days after the end of the *patient numbers* time series:

Date	Day	Predicted patients
01/04/2019	Monday	54
02/04/2019	Tuesday	54
03/04/2019	Wednesday	54
04/04/2019	Thursday	54
05/04/2019	Friday	54
06/04/2019	Saturday	54
07/04/2019	Sunday	54
08/04/2019	Monday	54
09/04/2019	Tuesday	54
10/04/2019	Wednesday	54
⋮	⋮	⋮

Our ARIMA(12,1,3) prediction for the next 10 days

This prediction, whilst being the most accurate that we found, fails to consider that there will be variation between days of the week and it is extremely unlikely that the surgery will receive a consistent 54 visitors across the 10 days. From our earlier boxplots, we imagine that the surgery will be busier on certain days of the week. We may therefore consider a hybrid model, taking the base values from the ARIMA(12,1,3) model, but accounting for the day coefficient values from the multiple linear regression model (which also performed well). This model will combine the good performance of the ARIMA(12,1,3) model with our expected weekly variation.

Date	Day	ARIMA(12,1,3) Prediction	Day Coefficient	ARIMA Prediction + Day Coefficient (Hybrid method)	Hybrid Method Prediction
01/04/2019	Monday	54	0	54	54
02/04/2019	Tuesday	54	-11.4733	42.52671	43
03/04/2019	Wednesday	54	-9.25295	44.74705	45
04/04/2019	Thursday	54	1.88087	55.88087	56
05/04/2019	Friday	54	-2.59553	51.40447	51
06/04/2019	Saturday	54	-3.46166	50.53834	51
07/04/2019	Sunday	54	-2.35445	51.64555	52
08/04/2019	Monday	54	0	54	54
09/04/2019	Tuesday	54	-11.4733	42.52671	43
10/04/2019	Wednesday	54	-9.25295	44.74705	45
⋮	⋮	⋮	⋮	⋮	⋮

Our hybrid method prediction for the next 10 days.

Conclusion

In this project, we have examined 10 different types of time series models. We have applied each of these models to our *patient numbers* dataset. We have also calculated certain measures of error associated with each model and used these values to judge the suitability of the model. Finally, for each model we have predicted the next 10 observations into the future.

We started with a baseline (naïve) model, which is seen as the level against which we should measure. Any model that appears to perform better than the baseline should be investigated and any models which perform worse should be discarded. We discovered that a number of extrapolation, regression and ARIMA models outperformed the naïve model. Methods involving simple linear regression performed badly. The poor performance of simple linear models could be explained by the high seasonality in the time series. We found that ARIMA and multiple linear regression models fitted the data best, and yielded the lowest error statistics when the test set was compared against the model's predictions.

We then listed the error and predictions of each model in a summary table before ranking each model by its effectiveness. The best-ranking model was the ARIMA(12,1,3) model as it had the lowest MSE (mean square error). We examined the prediction of the ARIMA(12,1,3) and realised that the prediction was that the surgery would be visited by a consistent 54 people per day. This value did not reflect the weekly variation in the time series, so I proposed, as our final prediction, combining the predictions from the ARIMA(12,1,3) model with the day coefficients from the multiple linear regression method. This 10-day prediction was then presented in a table.

We would recommend to the surgery management team using an ARIMA(12,1,3) model but expect there to be weekly variation in the numbers of patients, consistent with variation seen in other weeks.