2021

Time Series Analysis

Module number: DALT7010

Student Number: 19132761

MSc Course: Data Analytics

Word count: 843

Table of Contents

1. Plot time series and Trend

Read the dataset and set column name:

```
# Check the data
str(df1)
```

```
## tibble [108 x 1] (S3: tbl_df/tbl/data.frame)
##  $ value: num [1:108] 6550 8728 12026 14395 14587 ...
```

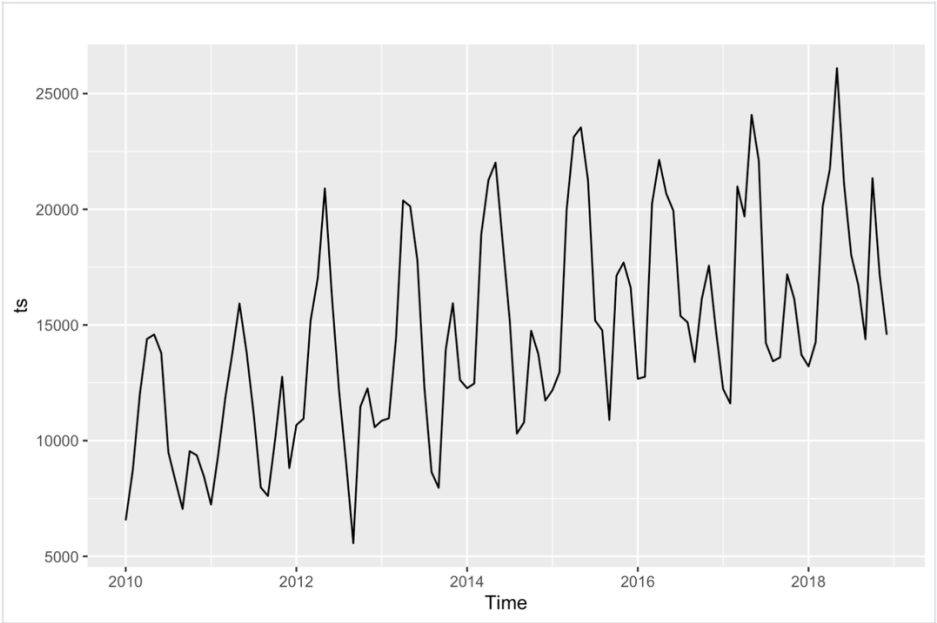
Convert the series to time series

```
# Convert to TS
ts <- ts(df1, start = c(2010,1), frequency = 12)
ts_info(ts)
```

```
##  The ts series is a ts object with 1 variable and 108 observations
##  Frequency: 12
##  Start time: 2010 1
##  End time: 2018 12
```

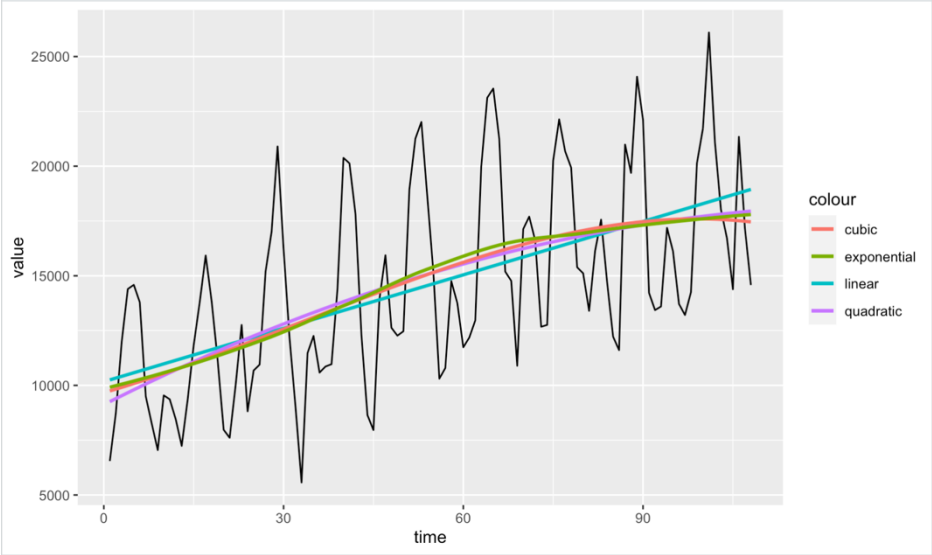
Plot the time series

```
# Plot the time series
autoplot(ts)
```



There is a clear, increasing trend and a strong seasonal pattern that increases in size as the level of the series increases.

Plot the models:



It is not clear which model fits the dataset best. So, we call CV() function to check the measure of predictive accuracy. We compare these values against the corresponding values from other models. For the CV, AIC, AICc and BIC measures, we want to find the model with the lowest value; for Adjusted R^2, we seek the highest value model. (Athanasopoulos, 2018)

```
# Linear model
linear <- lm(value~time)
CV(linear)
```

```
##           CV          AIC         AICc          BIC        AdjR2
## 1.436767e+07 1.782158e+03 1.782389e+03 1.790204e+03 3.094315e-01
```

```
# Exponential model
expo <- lm(log(value)~ time)
CV(expo)
```

```
##            CV           AIC          AICc           BIC         AdjR2
##    0.07131118 -283.03555828 -282.80478905 -274.98916460    0.33474627
```

```
# Quadratic model
quadratic <-lm(value ~ time + I(time^2))
CV(quadratic)
```

```
##           CV          AIC         AICc          BIC        AdjR2
## 1.441095e+07 1.782526e+03 1.782914e+03 1.793254e+03 3.133118e-01
```

```
# Cubic polynomial model
cubic <- lm(value ~ time + I(time^2) + I(time^3))
CV(cubic)
```

```
##           CV          AIC         AICc          BIC        AdjR2
## 1.461875e+07 1.784225e+03 1.784813e+03 1.797636e+03 3.086366e-01
```

From the CV, Adjusted R – squared of Exponential Model is biggest. It measures how well the model fits the historical data but not how well the model will forecast future data (Athanasopoulos, 2018). Looking at other measures, the Exponential model also has the minimum values of measures, the Exponential is the best forecasting model.

```
# Exponential model
summary(expo)
```

```
##
## Call:
## lm(formula = log(value) ~ time)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.78343 -0.19792 -0.03448  0.21727  0.56349
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 9.2084859  0.0513167 179.444  < 2e-16 ***
## time        0.0060526  0.0008173   7.405 3.29e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2648 on 106 degrees of freedom
## Multiple R-squared:  0.341,  Adjusted R-squared:  0.3347
## F-statistic: 54.84 on 1 and 106 DF,  p-value: 3.286e-11
```

The estimated Exponential Model equation is:

$$\mathbf{value = 9.2{*}0.006^{time}.}$$

2. Smooth the original series and Plot

Because both trend and seasonal components are present in the time series, so we use Holt-Winters Method to smooth the time series.

This method can be implemented with an "Additive" structure or a "Multiplicative" structure, where the choice of method depends on the data set. The Additive model is best used when the seasonal trend is of the same magnitude throughout the data set, while the Multiplicative Model is preferred when the magnitude of seasonality changes as time increases (Athanasopoulos, 2018). This data has seasonality and trend; however, it is unclear if seasonality is additive or multiplicative.

We will use the ets() function to identify the best fit model. The ets() function uses the AICc to select an appropriate model and returns information about the fitted model.

```
# Use ets identify "Additive"  or "Multiplicative"
ets(ts)
```

```
## ETS(A,A,A)
##
## Call:
##  ets(y = ts)
##
##   Smoothing parameters:
##     alpha = 0.1465
##     beta  = 2e-04
##     gamma = 0.0033
##
##   Initial states:
##     l = 9652.313
##     b = 91.8808
##     s = -2442.469 -37.0141 -562.6003 -4690.061 -3306.195 -1048.663
##            3780.367 6688.623 5003.571 2880.066 -2854.893 -3410.732
##
##   sigma:  1458.874
##
##      AIC     AICc      BIC
## 2096.004 2102.804 2141.600
```

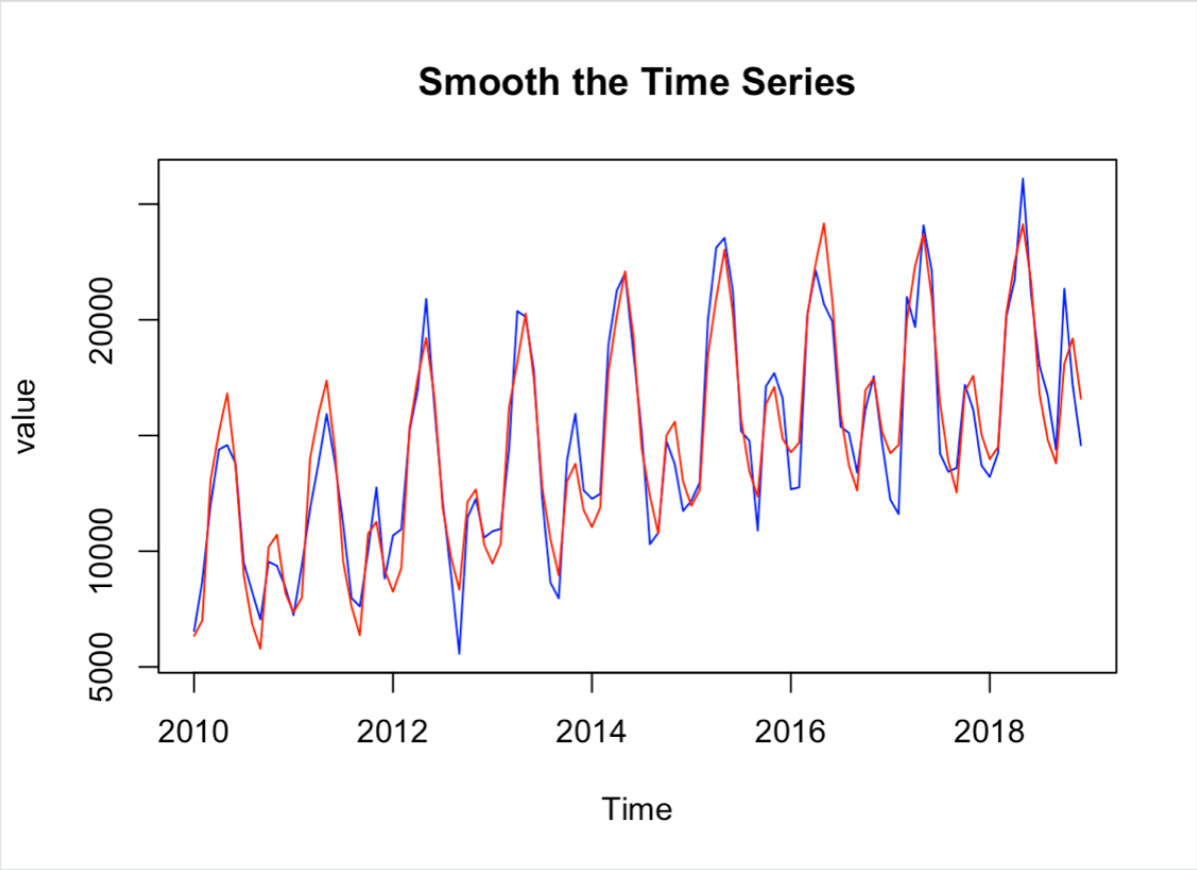
Now we have a time series that can be described using an additive model with increasing or decreasing trend and seasonality, we can use Holt-Winters exponential smoothing to make short-term forecasts.

```
# Applying Holt- winter's additive method
fit <- hw(ts, seasonal = "additive")
fitted(fit)
```

```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2010  6333.461  7012.944 13091.396 15150.852 16817.069 13673.596  8953.287
## 2011  7359.285  7993.950 14013.872 15911.103 17370.866 14349.705  9535.519
## 2012  8247.462  9261.174 15314.339 17514.536 19215.200 16657.077 11862.710
## 2013  9465.271 10325.144 16218.845 18168.209 20273.891 17440.074 12769.203
## 2014 11042.361 11875.054 17754.563 20158.367 22094.456 19272.995 14443.453
## 2015 11959.875 12643.033 18484.051 20933.515 23025.280 20288.868 15705.990
## 2016 14270.495 14688.840 20204.773 22442.838 24162.899 20841.004 15978.797
## 2017 14234.079 14589.401 19956.019 22335.700 23701.084 20952.901 16394.043
## 2018 13976.949 14509.596 20287.502 22481.332 24132.460 21618.799 16799.580
##            Aug       Sep       Oct       Nov       Dec
## 2010  6867.176  5777.914 10183.692 10707.688  8197.115
## 2011  7607.449  6368.295 10762.791 11267.808  9177.737
## 2012  9737.342  8339.840 12140.062 12664.842 10290.264
## 2013 10525.569  8953.242 13021.055 13777.743 11782.065
## 2014 12376.751 10780.850 15002.902 15594.373 13008.655
## 2015 13447.024 12355.365 16361.268 17097.330 14871.910
## 2016 13714.075 12623.717 16964.222 17464.699 15168.458
## 2017 13902.853 12535.333 16910.905 17575.184 15048.049
## 2018 14811.729 13798.062 18101.452 19194.946 16586.912
```

The original data and data smoothed with Holt-Winter's method is plotted.

```
# Plotting the smoothed data
plot.ts(ts, main = "Smooth the Time Series", col = "blue")
lines(fitted(fit),col = "red")
```




```
# Estimates of model parameters
fit$model
```

```
## Holt-Winters' additive method
##
## Call:
##  hw(y = ts, seasonal = "additive")
##
##   Smoothing parameters:
##     alpha = 0.1465
##     beta  = 2e-04
##     gamma = 0.0033
##
##   Initial states:
##     l = 9652.313
##     b = 91.8808
##     s = -2442.469 -37.0141 -562.6003 -4690.061 -3306.195 -1048.663
##           3780.367 6688.623 5003.571 2880.066 -2854.893 -3410.732
##
##   sigma:  1458.874
##
##      AIC     AICc      BIC
## 2096.004 2102.804 2141.600
```

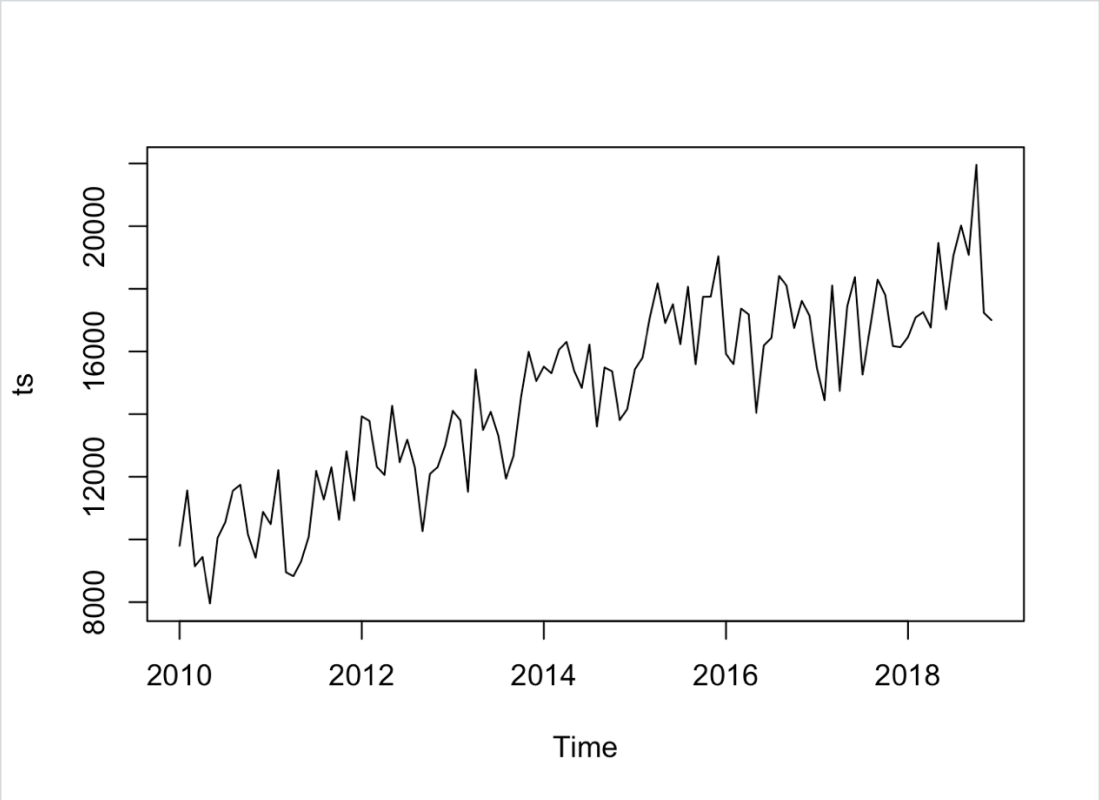
Holt-Winters exponential smoothing estimates the seasonal, level and slope component at the present time point. Three parameters control smoothing: alpha, beta, and gamma, for the estimates of the level, slope b of the trend component, and the seasonal component, respectively, at the current time point. The parameters alpha, beta and gamma all have values between 0 and 1, and close to 0 mean that relatively little weight is placed on the most recent observations when making forecasts of future values. (Coghlan, n.d.)

3. Seasonally adjust the series and Forecast one year ahead.

Using the estimate of the seasonal component, calculated and plot the seasonally adjusted time series:

Additive: Seasonally Adjusted = Time Series – Seasonal

```
# Using the estimate of the seasonal component calculated
decompose_ts <- decompose(ts, "additive")
ts_adj <- ts - decompose_ts$seasonal
# Plot the seasonally adjusted time series
plot(ts_adj)
```



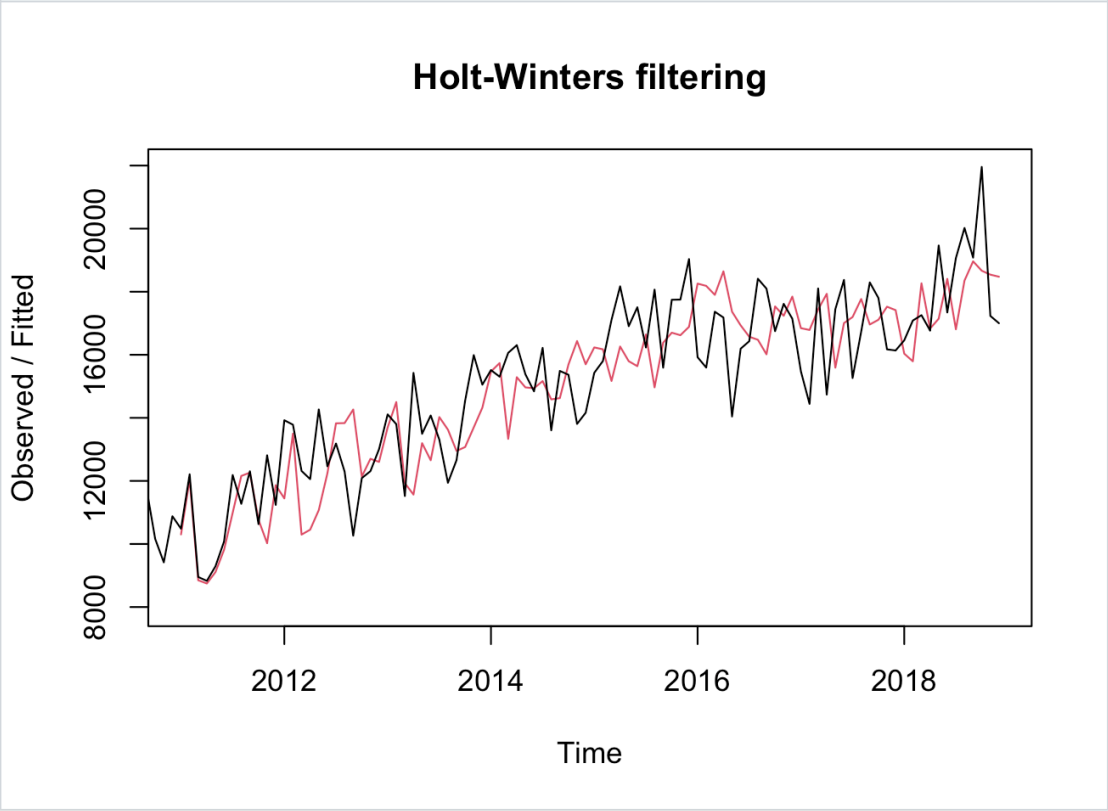
```r
# Fit a predictive model
ts_fore <- HoltWinters(ts_adj)
ts_fore
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = ts_adj)
##
## Smoothing parameters:
##  alpha: 0.1371459
##  beta : 0.00517786
##  gamma: 0.5104313
##
## Coefficients:
##            [,1]
## a   18037.70360
## b      61.38162
## s1   -276.63724
## s2   -250.59745
## s3    972.00926
## s4     36.61335
## s5   1349.47683
## s6    744.45349
## s7    697.52863
## s8   1612.08455
## s9   1249.85672
## s10  2272.92266
## s11  -391.76784
## s12  -413.94750
```

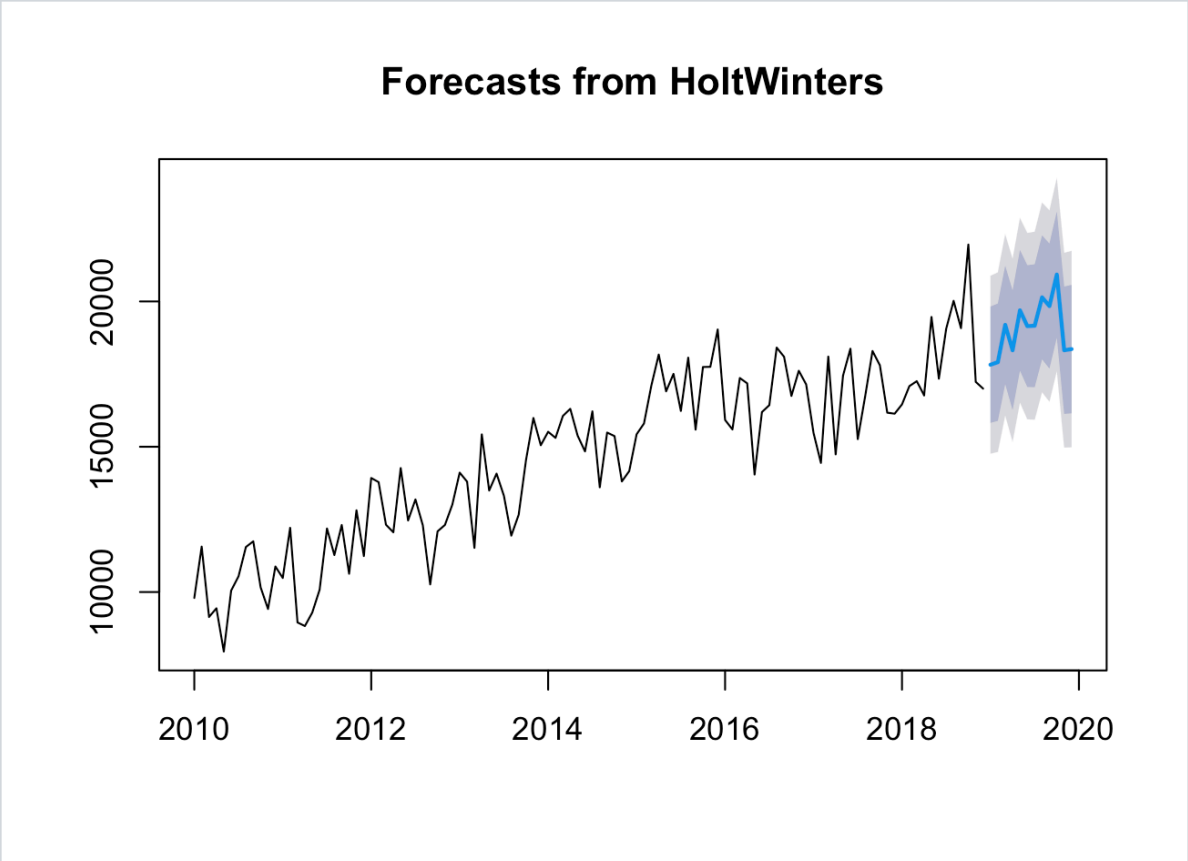
The estimated values of alpha, beta and gamma are 0.137, 0.005, 0.51, respectively. The value of alpha (0.137) is relatively low, indicating that the estimate of the level at the current time point is based upon both recent observations and some observations in the more distant past. The value of beta is 0.005, indicating that the estimate of the slope b of the trend component is not updated over the time series and instead is set equal to its initial value. This makes good intuitive sense, as the level changes quite a bit over the time series, but the slope b of the trend component remains roughly the same. In contrast, the gamma (0.51) value is high, indicating that the estimate of the seasonal component at the current time point is just based upon very recent observations. (Coghlan, n.d.)

Plot the adjusted time series as a black line, with the forecasted values as a red line on top of that.

```r
# Plot the adjusted time series
plot(ts_fore)
```




```
# Make forecast
ts_fore2 <- forecast:::forecast.HoltWinters(ts_fore, h=12)
forecast:::plot.forecast(ts_fore2)
```



The forecasts are shown as a blue line, and the grey and light grey shaded areas show 80% and 95% prediction intervals, respectively.

```
# Summary
summary(ts_fore2)
```

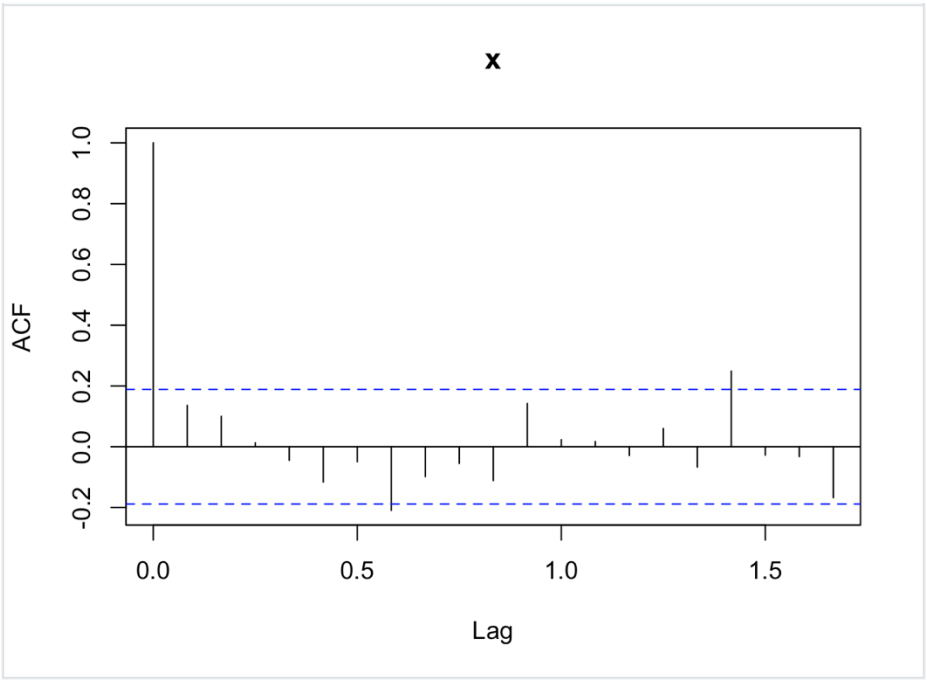
```
##
## Forecast method: HoltWinters
##
## Model Information:
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = ts_adj)
##
## Smoothing parameters:
##  alpha: 0.1371459
##  beta : 0.00517786
##  gamma: 0.5104313

## Coefficients:
##            [,1]
## a   18037.70360
## b      61.38162
## s1   -276.63724
## s2   -250.59745
## s3    972.00926
## s4     36.61335
## s5   1349.47683
## s6    744.45349
## s7    697.52863
## s8   1612.08455
## s9   1249.85672
## s10  2272.92266
## s11  -391.76784
## s12  -413.94750
##
## Error measures:
##                      ME     RMSE      MAE       MPE     MAPE     MASE      ACF1
## Training set 179.7967 1563.491 1231.476 0.6970466 8.074457 0.772075 0.1359301
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2019       17822.45 15821.60 19823.30 14762.41 20882.48
## Feb 2019       17909.87 15890.10 19929.64 14820.89 20998.85
## Mar 2019       19193.86 17155.14 21232.57 16075.92 22311.80
## Apr 2019       18319.84 16262.17 20377.51 15172.91 21466.78
## May 2019       19694.09 17617.44 21770.73 16518.13 22870.05
## Jun 2019       19150.45 17054.81 21246.09 15945.44 22355.45
## Jul 2019       19164.90 17050.25 21279.56 15930.82 22398.99
## Aug 2019       20140.84 18007.15 22274.53 16877.65 23404.03
## Sep 2019       19839.99 17687.26 21992.73 16547.66 23132.33
## Oct 2019       20924.44 18752.63 23096.25 17602.94 24245.94
## Nov 2019       18321.13 16130.23 20512.04 14970.43 21671.83
## Dec 2019       18360.34 16150.32 20570.36 14980.40 21740.27
```

The forecasted values for the next one year i.e. assuming it is for 2019, is displayed. The values are calculated at 80% and 95% confidence interval.

We can investigate whether the prediction model can be improved by testing if the forecast errors in the sample display non-zero autocorrelations at lags 1-20.

Making a correlogram and carrying out the Ljung-Box test.

```
# The Ljung-Box test
acf(ts_fore2$residuals, na.action = na.pass, lag.max=20)
```

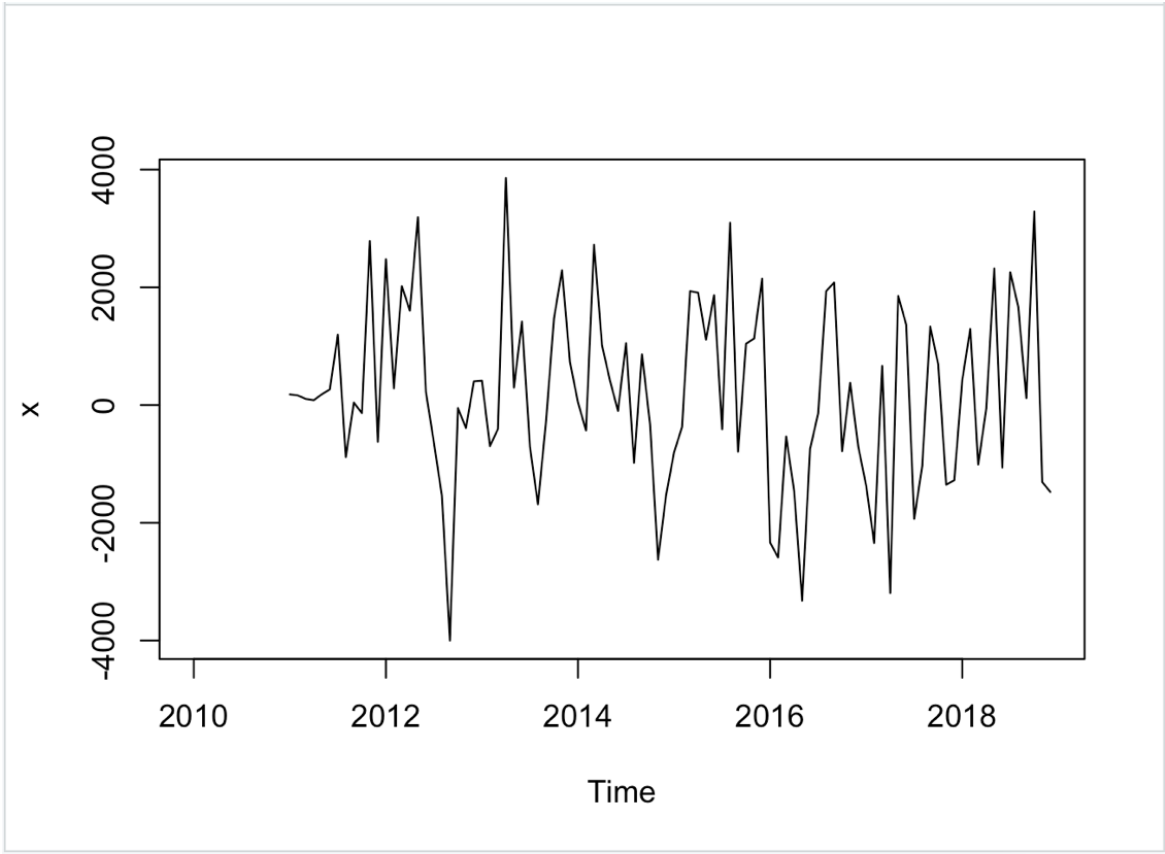



```
Box.test(ts_fore2$residuals, lag=20, type="Ljung-Box")
```

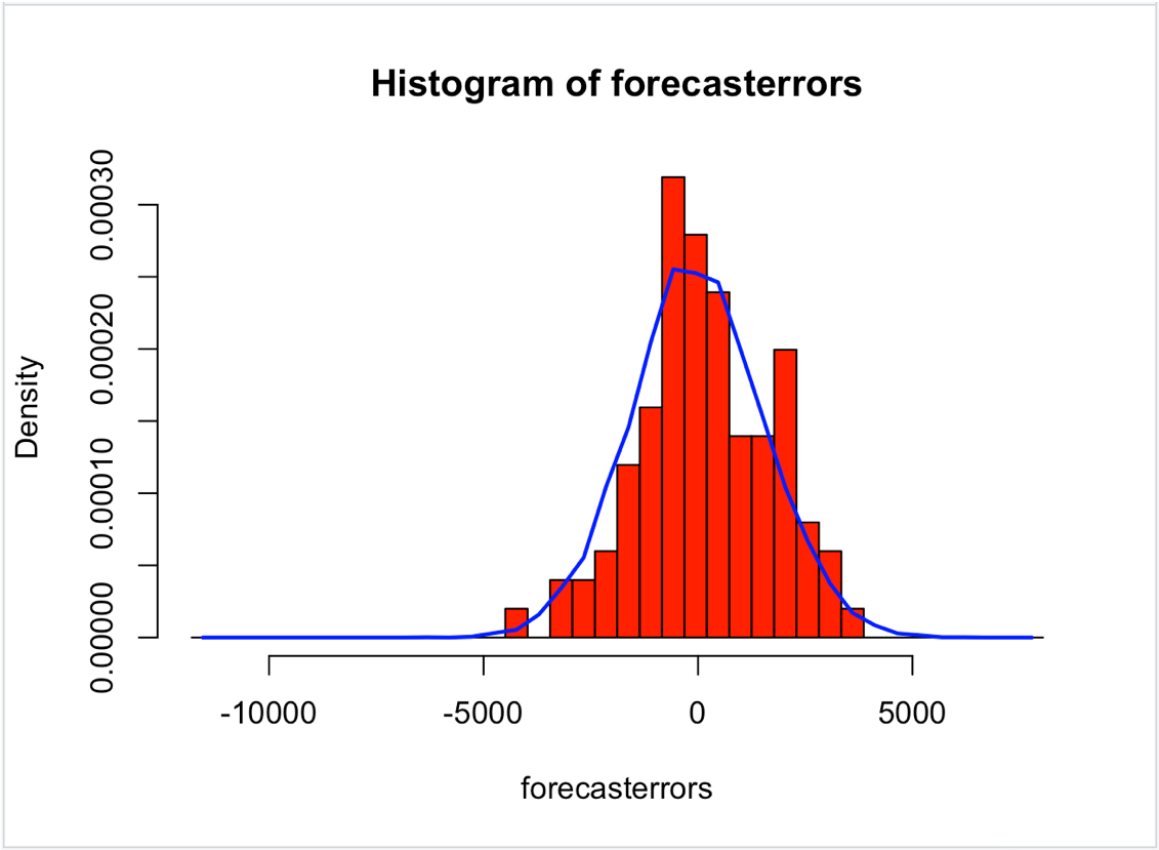
```
##
##  Box-Ljung test
##
## data:  ts_fore2$residuals
## X-squared = 26.491, df = 20, p-value = 0.1502
```

The correlation chart shows that the autocorrelation for predictive errors in the sample do not almost always exceed the significant limit for lags 1-20. Furthermore, the p-value for the Ljung-Box test is 0.1502, suggesting that there is very little evidence of nonzero autocorrelations at lags 1-20.

```
# Time plot
plot.ts(ts_fore2$residuals)
```




```
# Histogram
plotForecastErrors(ts_fore2$residuals)
```



From the time plot, it seems plausible that forecast errors have constant variances over time. The histogram of forecast errors seems plausible when the forecast errors are normally distributed with mean zero.

Consequently, there is little evidence at lag 1-20 of autocorrelation for forecast errors, and the forecast errors appear to be normally distributed with the mean zero and constant variance over time. It suggests that Holt-Winters' exponential smoothing provides an appropriate forecast model, which probably cannot be improved. Furthermore, the assumptions based on the predicted time period may be valid.

Bibliography

Athanasopoulos, R. J. (2018). *Forecasting: Principles and Practice.*

Coghlan, A. (n.d.). *Using R for Time Series Analysis.* Retrieved from Read the docs: https://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/src/timeseries.html