2021

Time Series Analysis

Module number: DALT7010

Student Number: 19132761

MSc Course: Data Analytics

Word count: 1485

Table of Contents

1. Comment on the stationarity and describe how it is identified

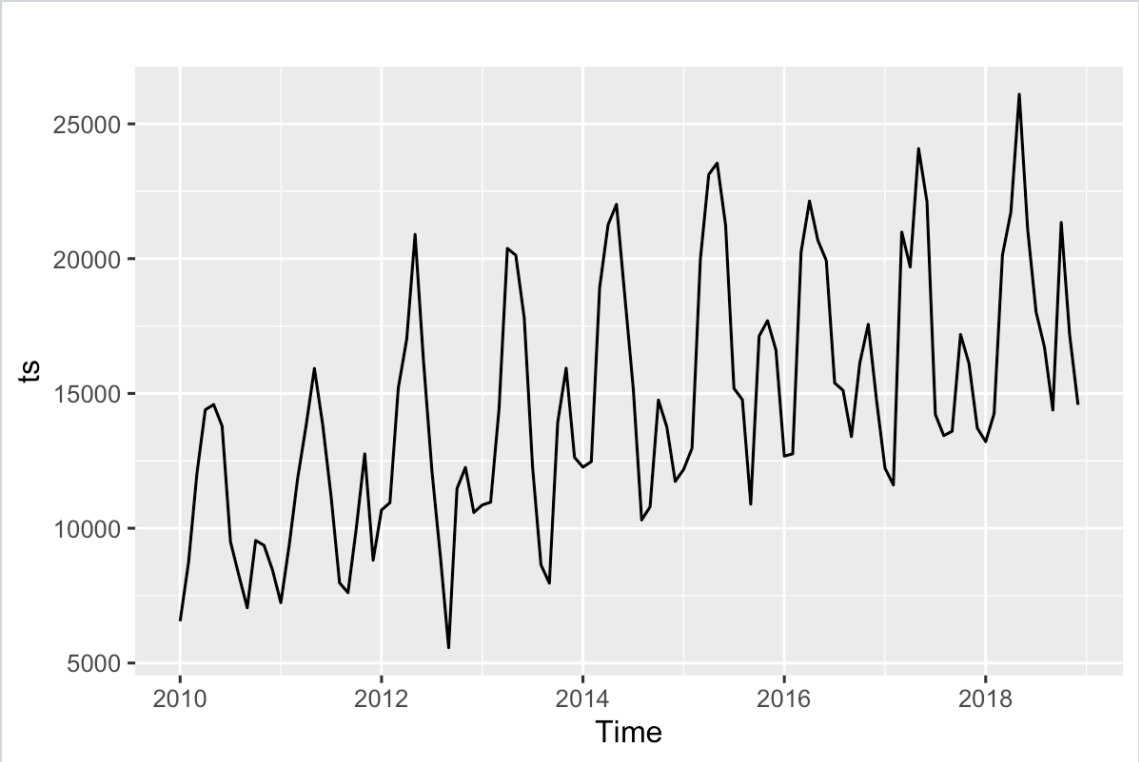
In the most intuitive sense, stationarity means that the statistical properties of a process generating a time series do not change over time. It does not mean that the series does not change over time, just that the way it changes does not itself change over time. (Palachy, 2019)

Stationarity is essential because many useful analytical tools and statistical tests and models rely on it. Therefore, determining whether a time series was generated by a stationary process is important.

Visualization

The most basic stationarity detection methods rely on plotting the data or functions of it and determining whether they present some known property of stationary (or non-stationary) data. (Shay, 2019)

```
# Plot time series
ts = ts(df, start = c(2010,1), frequency = 12)
autoplot(ts)
```



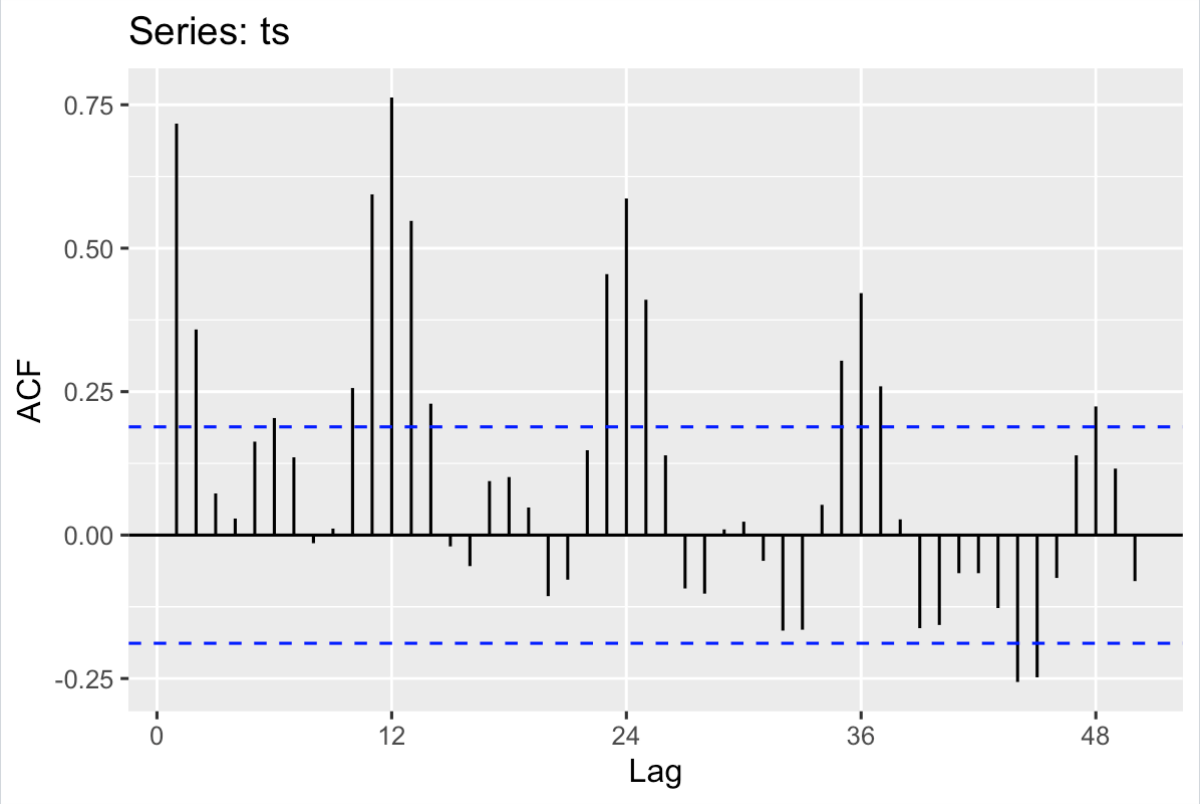
A moderate increasing trend and seasonality suggest that the series is not stationary.

However, this is not a reliable method for detecting stationarity, and it is more often used for getting an initial impression of the data than for making an affirmative determination. Trying to determine whether a time series is stationary just by looking at its plot is a venture.

Autocorrelation Function plot

Autocorrelation is the correlation of a signal with a delayed copy – or a lag – of itself as a function of the delay. When plotting the value of the ACF for increasing lags (a plot called a correlogram), the values tend to degrade to zero quickly for stationary time series, while for non-stationary data, the degradation will happen more slowly. (Shay, 2019)

```
# Check the acf for further inspection
Acf(ts,lag=50)
```



Now we can see the clear monthly pattern of seasonality. It also shows significant autocorrelation coefficients (spikes outside the 95% CI) do not rapidly decay. The data is not stationary.

Another more rigorous approach to detect stationarity in time series data is to use statistical - parametric tests, which are developed to detect specific types of stationarity.

Ljung-Box test for independence

```
# Ljung-Box test for independence
Box.test(ts, lag=50, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  ts
## X-squared = 486.72, df = 50, p-value < 2.2e-16
```

The Ljung-Box test examines whether there is significant evidence for non-zero correlations at 50 lags, with the null hypothesis of independence in the time series (p-value small), the time series data is non-stationary.

Augmented Dickey–Fuller (ADF) test for unit root

The Dickey-Fuller test was the first statistical test developed to test the null hypothesis that a unit root is present in an autoregressive model of a given time series and that the process is thus not stationary. (Shay, 2019)

The ADF test determines whether the change in Y can be explained by a lagged value (e.g., a value at a previous time point Y [t-1]) and by a linear trend. If there is a linear trend but the lagged value cannot explain the change in Y over time, then our data will be deemed non-stationary. (Salas, n.d.)

```r
# Augmented Dickey-Fuller test
stationary.test(ts)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -0.959   0.335
## [2,]   1 -1.221   0.240
## [3,]   2 -1.047   0.303
## [4,]   3 -0.598   0.465
## [5,]   4 -0.341   0.545
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -4.29  0.0100
## [2,]   1 -5.45  0.0100
## [3,]   2 -5.16  0.0100
## [4,]   3 -3.43  0.0131
## [5,]   4 -2.66  0.0890
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -5.04    0.01
## [2,]   1 -7.10    0.01
## [3,]   2 -7.60    0.01
## [4,]   3 -5.35    0.01
## [5,]   4 -4.32    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

The null hypothesis of the presence of a unit root is not rejected. The test shows that the time series is stationarity.

Kwiatkowski-Phillips-Schmidt-Shin (KPSS) for a level or trend stationarity

Another prominent test for the presence of a unit root is the KPSS test. [Kwiatkowski et al., 1992] Conversely to the Dickey-Fuller family of tests, the null hypothesis assumes stationarity around a mean or a linear trend, while the alternative is the presence of a unit root. (Shay, 2019)

```r
# KPSS for level or trend stationarity
stationary.test(ts, method = "kpss")
```

```
## KPSS Unit Root Test
## alternative: nonstationary
##
## Type 1: no drift no trend
##  lag  stat p.value
##    2 0.483     0.1
## -----
##  Type 2: with drift no trend
##  lag  stat p.value
##    2 0.354  0.0968
## -----
##  Type 1: with drift and trend
##  lag   stat p.value
##    2 0.0235     0.1
## ----------
## Note: p.value = 0.01 means p.value <= 0.01
##     : p.value = 0.10 means p.value >= 0.10
```

The test shows that the null hypothesis is false. Because the alternative hypothesis of the KPSS-test is the presence of a unit root, the results are conclusive, and the time series is non-stationary.

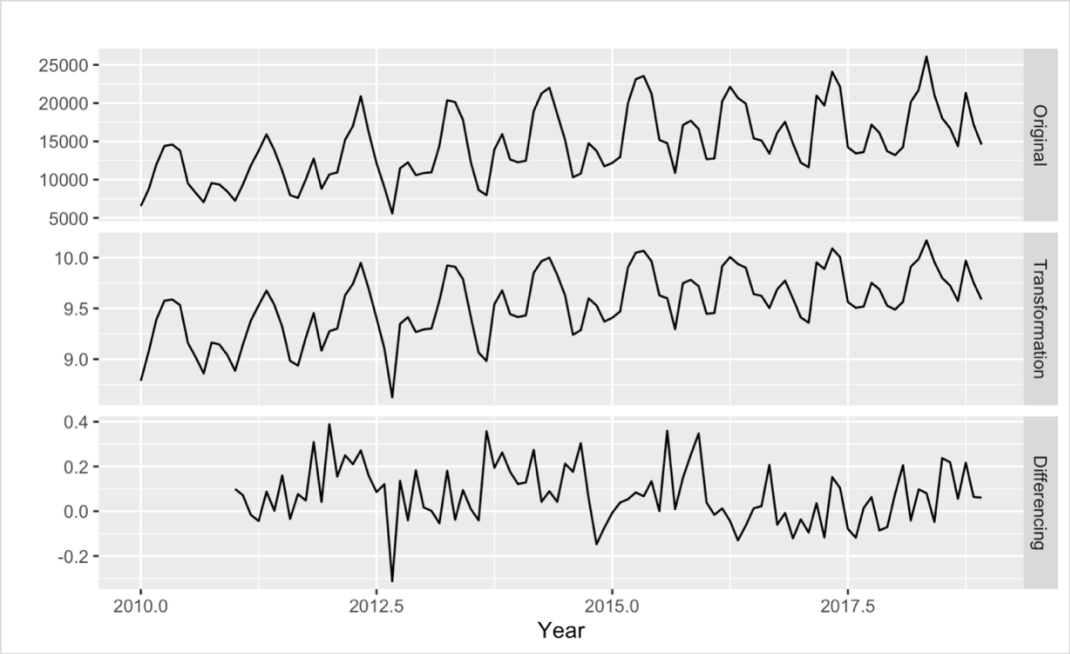
2. Explain possible differencing and transforming of the series to make it mean and variance stationery

A stationary time series is one whose properties do not depend on the time at which the series is observed. Thus, time series with trends, or with seasonality, are not stationary — the trend and seasonality will affect the value of the time series at different times. (Hyndman, 2014)

One way to make a non-stationary time series stationary – compute the differences between consecutive observations. This is known as differencing.

Transformations such as logarithms can help to stabilize the variance of a time series. Differencing can help stabilize the mean of a time series by removing changes in the level of a time series and eliminating (or reducing) trend and seasonality.

```r
# plot seasonal differencing
cbind("Original" = ts,
      "Transformation" = log(ts),
      "Differencing" = diff(log(ts),12)) %>%
  autoplot(facets=TRUE) +
  xlab("Year") + ylab("")
```



The transformation and differencing have made the series look relatively stationary. The logarithms stabilize the variance, and the seasonal differences remove the seasonality and trend.

Using unit root test to check whether differencing is required:

```r
# test
ts %>% log() %>% diff() %>% ur.kpss() %>% summary()
```

```
##
## ######################
## # KPSS Unit Root Test #
## ######################
##
## Test is of type: mu with 4 lags.
##
## Value of test-statistic is: 0.0517
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

The test statistic is much smaller than the 1% critical value, indicating that the null hypothesis is not rejected. That is, the time-series data is stationary. We use the KPSS test to determine the appropriate number of seasonal differencing.

```r
# seasonal differences D=1
nsdiffs(ts)
```

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

One seasonal differencing is enough to make the data stationary.

3. Investigate the sample autocorrelation function and partial autocorrelation function to determine the order of the SARIMA model

The seasonal ARIMA model incorporates both non-seasonal and seasonal factors in a multiplicative model.

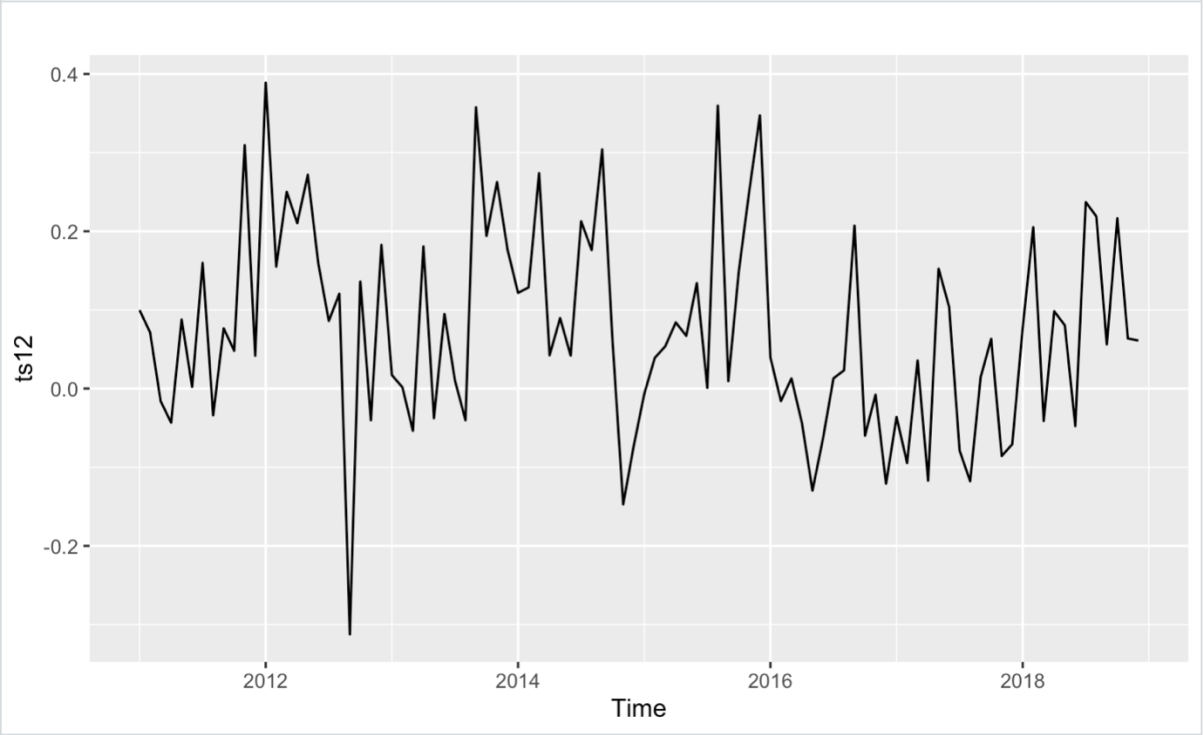
ARIMA (p,d,q) (P,D,Q)s

With:

p	*non-seasonal AR order*
d	*non-seasonal differencing*
q	*non-seasonal MA order*
P	*seasonal AR order*
D	*seasonal differencing*
Q	*seasonal MA order*
s	*period of repeating seasonal pattern.*

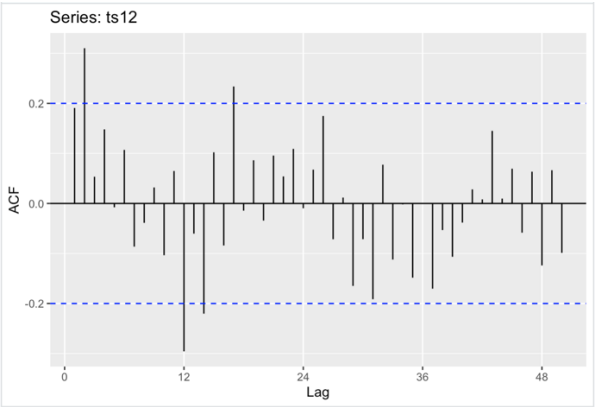
To stationarity the time series data, we did one seasonal differencing, so D = 1, d = 0 and s = 12.

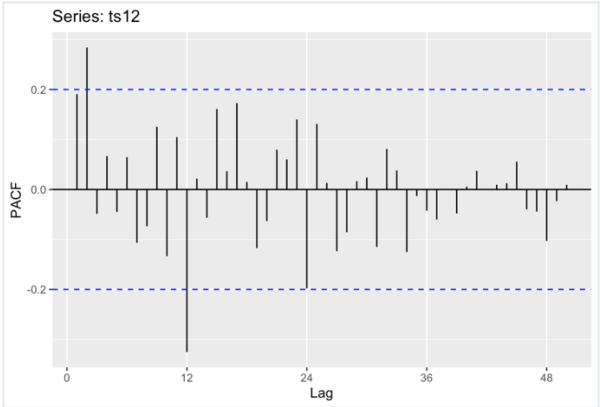
```r
# log and diff
ts12 <- diff(log(ts), lag = 12)
autoplot(ts12)
```




```
# Check acf and pacf
autoplot(Acf(ts12, lag=50))
```

```
autoplot(Pacf(ts12, lag=50))
```





Non-seasonal terms: Examine the early lags to judge non-seasonal terms. Spikes in the ACF indicate non-seasonal MA terms (q = 1). Spikes in the PACF indicate possible non-seasonal AR terms (p = 1).

Seasonal terms: Examine the patterns across lags that are multiples of s. (lag 12 and 24). Judge the ACF and PACF at the seasonal lags in the same way we do for the

earlier lags. We got Q = 1 and P = 0. However, we can consider Q = 1 and P = 1 because the spike at lag 24 is not obvious.

Suggested models: SARIMA(1,0,1)(0,1,1)$_{12}$ and SARIMA(1,0,1)(1,1,1)$_{12}$

4. Report the estimates together with a complete set of diagnostics

SARIMA(1,0,1)(0,1,1)$_{12}$

```
# SARIMA(1,0,1)(0,1,1)12
fit1 <- Arima(ts, order=c(1,0,1), seasonal=c(0,1,1))
summary(fit1)
```

```
## Series: ts
## ARIMA(1,0,1)(0,1,1)[12]
##
## Coefficients:
##          ar1      ma1     sma1
##       0.9909  -0.7910  -0.5726
## s.e.  0.0162   0.1134   0.1326
##
## sigma^2 estimated as 2451493:  log likelihood=-843.54
## AIC=1695.09   AICc=1695.53   BIC=1705.34
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE     MASE       ACF1
## Training set 123.1706 1452.931 1100.907 -0.1915507 7.752286 0.690215 0.06525147
```

```
checkresiduals(fit1)
```

```
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,1)(0,1,1)[12]
## Q* = 26.793, df = 19, p-value = 0.1096
##
## Model df: 3.   Total lags used: 22
```

SARIMA(1,0,1)(1,1,1)$_{12}$

```
# SARIMA(1,0,1)(1,1,1)12
fit2 <- Arima(ts, order=c(1,0,1), seasonal=c(1,1,1))
summary(fit2)
```

```
## Series: ts
## ARIMA(1,0,1)(1,1,1)[12]
##
## Coefficients:
##          ar1      ma1    sar1     sma1
##       0.9982  -0.7979  0.2705  -0.8643
## s.e.  0.0089   0.1045  0.2281   0.3460
##
## sigma^2 estimated as 2314843:  log likelihood=-842.96
## AIC=1695.93   AICc=1696.59   BIC=1708.75
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 115.6157 1404.245 1072.551 -0.2668505 7.520714 0.6724371
##                     ACF1
## Training set 0.06085107

##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,1)(1,1,1)[12]
## Q* = 26.96, df = 18, p-value = 0.07976
##
## Model df: 4.   Total lags used: 22
```

As we can see, both model's coefficients are significant, and in terms of $\sigma^{\hat{}2}$, information criteria, error measures, Q* and finally diagnostics, there is little between them.

However, we can use auto.arima function in R to select the best p, d, q coordinates automatically based on the model with the lowest Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC).

auto.arima() function

The auto.arima() function uses nsdiffs() to determine D (the number of seasonal differences to use), and ndiffs() to determine d (the number of ordinary differences to use). The selection of the other model parameters (p, q, P and Q) is all determined by minimizing the AICc, as with non-seasonal ARIMA models. Unit root test by the KPSS.

```r
# auto.arima
fitauto <- auto.arima(ts, test ="kpss")
summary(fitauto)
```

```
## Series: ts
## ARIMA(1,0,1)(0,1,1)[12] with drift
##
## Coefficients:
##          ar1      ma1     sma1    drift
##       0.7663  -0.5394  -0.5619  83.3957
## s.e.  0.1685   0.2208   0.1337  12.8709
##
## sigma^2 estimated as 2328583:  log likelihood=-840.23
## AIC=1690.45   AICc=1691.12   BIC=1703.27
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 15.74514 1408.406 1076.439 -0.9011266 7.535356 0.6748744
##                      ACF1
## Training set -0.01349079
```

```r
checkresiduals(fitauto)
```

```
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,1)(0,1,1)[12] with drift
## Q* = 26.969, df = 18, p-value = 0.07957
##
## Model df: 4.   Total lags used: 22
```

The auto.arima() function suggests the best model is ARIMA$(1,0,0)(0,1,1)_{12}$ (it has the smallest AICc value). It is matched with one of the models we selected manually.

5. Report the estimates together with a complete set of diagnostics

```r
# show all the options
auto.arima(ts, trace = TRUE, test ="kpss")
```

AICc values of SARIMA models

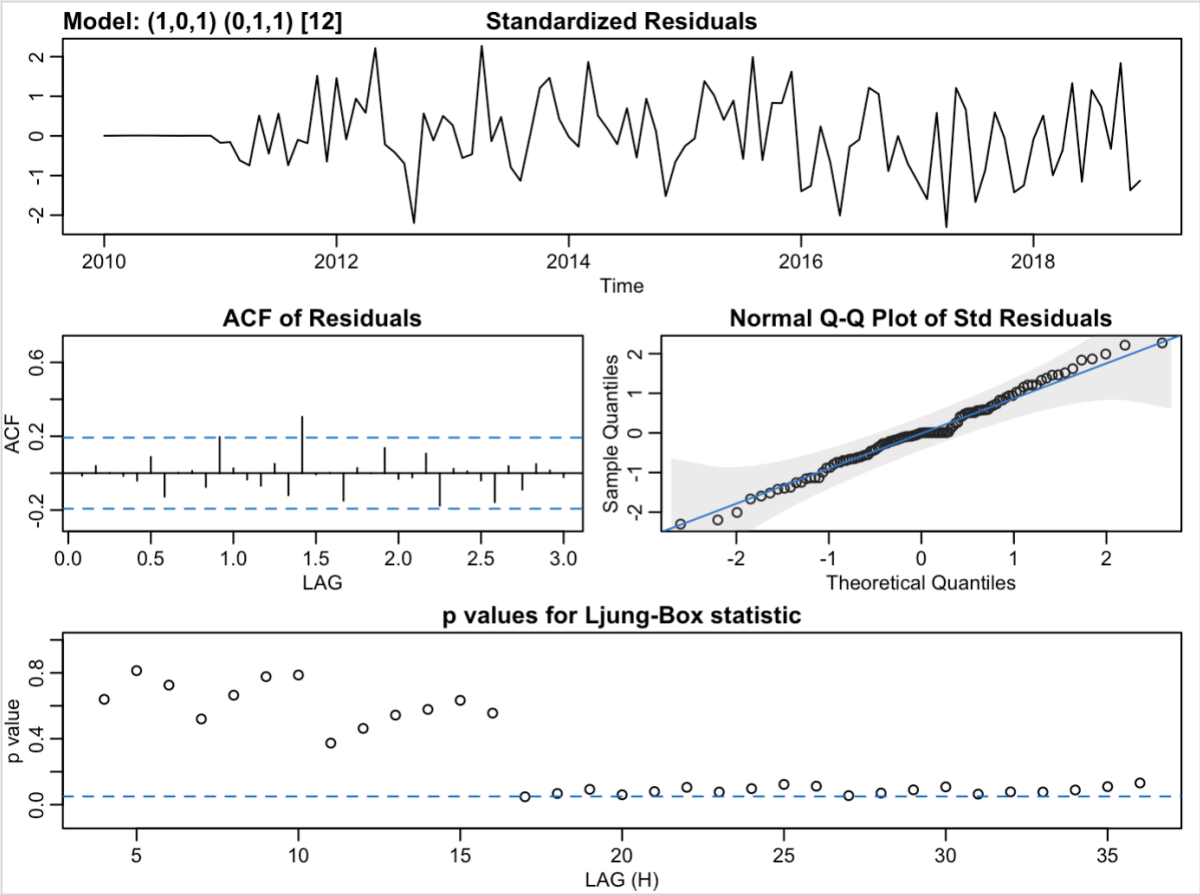
p, d, q	P, D, Q	AICc value
2,0,2	1,1,1	Inf
0,0,0	0,1,0	1716.109
1,0,0	1,1,0	1707.371

0,0,1	0,1,1	1704.494
0,0,0	0,1,0	1738.717
0,0,1	0,1,0	1717.629
0,0,1	1,1,1	Inf
0,0,1	0,1,2	1706.609
0,0,1	1,1,0	1709.034
0,0,1	1,1,2	Inf
0,0,0	0,1,1	1705.384
1,0,1	0,1,1	1703.275
1,0,1	0,1,0	1716.745
1,0,1	1,1,1	Inf
1,0,1	0,1,2	1706.021
1,0,1	1,1,0	1708.027
1,0,1	1,1,2	Inf
1,0,0	0,1,1	1702.279
1,0,0	0,1,0	1716.089
1,0,0	1,1,1	Inf
1,0,0	0,1,2	1704.618
1,0,0	1,1,2	Inf
2,0,0	0,1,1	1703.588
2,0,1	0,1,1	1707.677
1,0,0	0,1,1	1723.344

The best model is ARIMA(1,0,0)$(0,1,1)_{12}$ selected based on smallest AICc value.

```
##  Best model: ARIMA(1,0,1)(0,1,1)[12] with drift

## Series: ts
## ARIMA(1,0,1)(0,1,1)[12] with drift
##
## Coefficients:
##          ar1      ma1     sma1    drift
##       0.7663  -0.5394  -0.5619  83.3957
## s.e.  0.1685   0.2208   0.1337  12.8709
##
## sigma^2 estimated as 2328583:  log likelihood=-840.23
## AIC=1690.45   AICc=1691.12   BIC=1703.27
```



There are a few significant spikes in the ACF, and the model fails the Ljung-Box test. The model can still be used for forecasting, but the prediction intervals may not be accurate due to the correlated residuals.

6. Provide forecasts for one year ahead and comparing with HoltWinters method

Forecasts for one year ahead

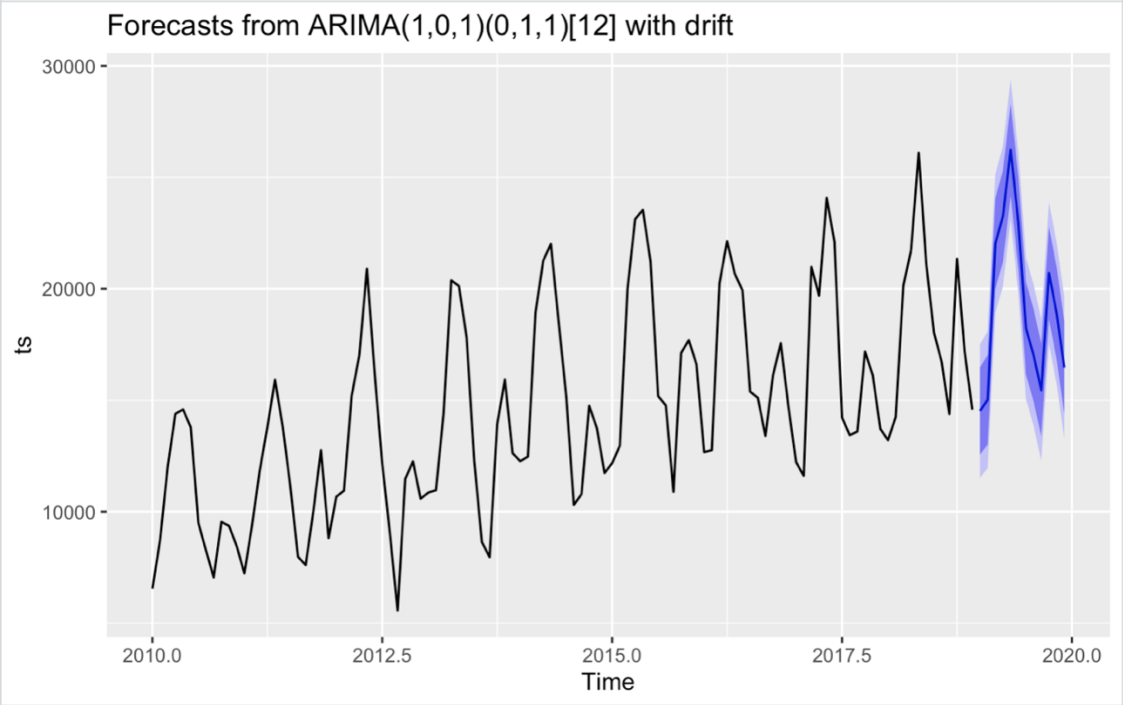
```
# Forecast
forecast <- forecast(fitauto,12)
forecast
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2019       14521.38 12565.75 16477.02 11530.50 17512.27
## Feb 2019       15022.90 13017.53 17028.27 11955.95 18089.85
## Mar 2019       22029.19 19995.18 24063.21 18918.44 25139.95
## Apr 2019       23265.85 21215.20 25316.49 20129.65 26402.04
## May 2019       26231.38 24171.03 28291.73 23080.34 29382.42
## Jun 2019       22992.49 20926.46 25058.52 19832.77 26152.21
## Jul 2019       18226.15 16156.79 20295.50 15061.34 21390.95
## Aug 2019       17018.69 14947.38 19090.00 13850.89 20186.48
## Sep 2019       15451.17 13378.72 17523.63 12281.63 18620.72
## Oct 2019       20703.49 18630.37 22776.62 17532.92 23874.06
## Nov 2019       18906.93 16833.41 20980.45 15735.75 22078.10
## Dec 2019       16463.57 14389.82 18537.32 13292.04 19635.10
```

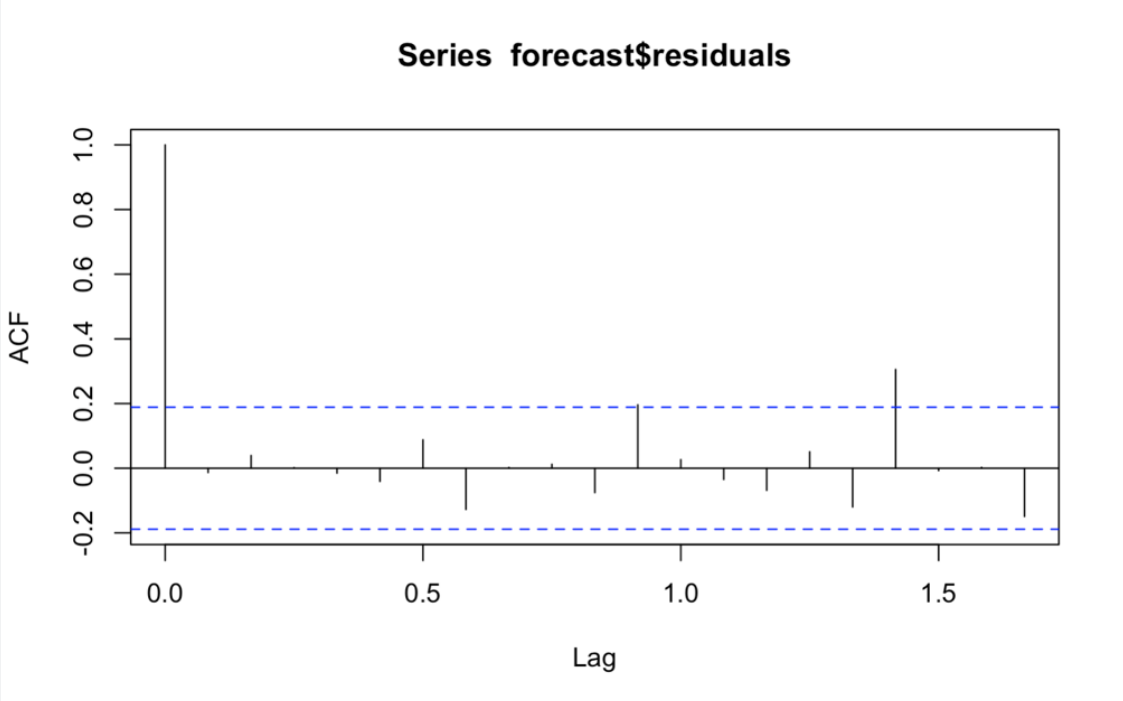
```
# Forecast
forecast <- forecast(fitauto,12)
forecast
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2019       14521.38 12565.75 16477.02 11530.50 17512.27
## Feb 2019       15022.90 13017.53 17028.27 11955.95 18089.85
## Mar 2019       22029.19 19995.18 24063.21 18918.44 25139.95
## Apr 2019       23265.85 21215.20 25316.49 20129.65 26402.04
## May 2019       26231.38 24171.03 28291.73 23080.34 29382.42
## Jun 2019       22992.49 20926.46 25058.52 19832.77 26152.21
## Jul 2019       18226.15 16156.79 20295.50 15061.34 21390.95
## Aug 2019       17018.69 14947.38 19090.00 13850.89 20186.48
## Sep 2019       15451.17 13378.72 17523.63 12281.63 18620.72
## Oct 2019       20703.49 18630.37 22776.62 17532.92 23874.06
## Nov 2019       18906.93 16833.41 20980.45 15735.75 22078.10
## Dec 2019       16463.57 14389.82 18537.32 13292.04 19635.10
```

```
# Plot
autoplot(forecast)
```




```
# Test
acf(forecast$residuals, lag.max=20)
```

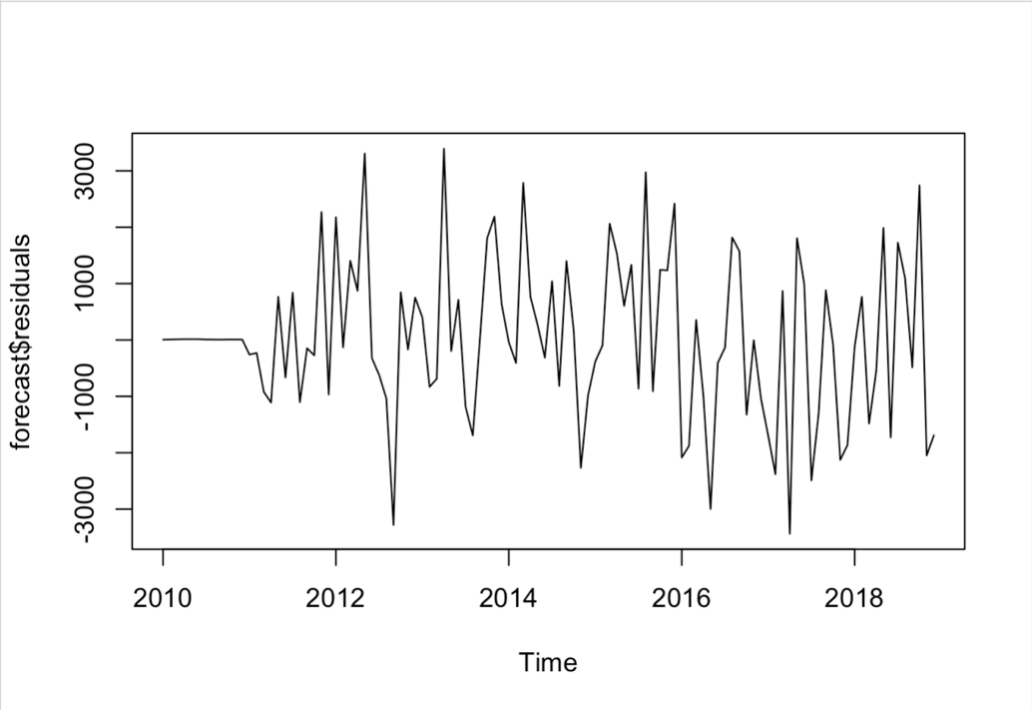



```
Box.test(forecast$residuals, lag=20, type="Ljung-Box")
```

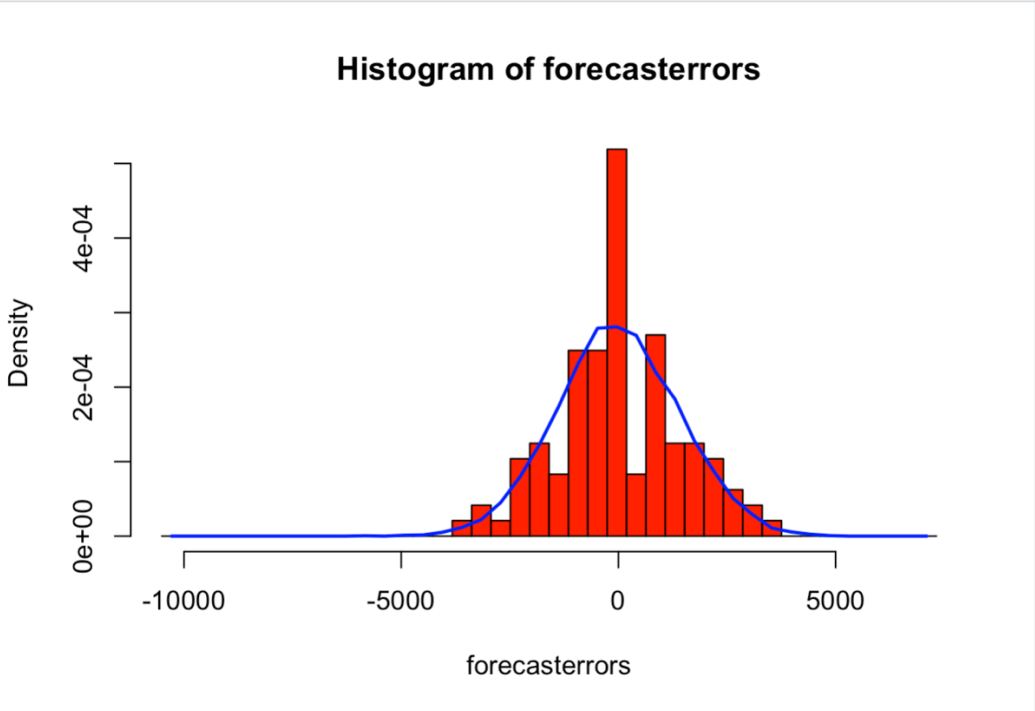
```
##
##  Box-Ljung test
##
## data:  forecast$residuals
## X-squared = 26.858, df = 20, p-value = 0.1393
```

The p-value for the Ljung-Box test is 0.1393, suggesting that there is little evidence for non-zero autocorrelations in the forecast errors for lags 1-20.

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




```
# Histogram
plotForecastErrors(forecast$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 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 ARIMA exponential smoothing provides an appropriate forecast model, which probably cannot be improved. Furthermore, the assumptions based on the predicted time period may be valid.

Comparing Box-Jenkins and Holt-Winters'

The p-value for the Ljung-Box test of Holt-Winters' is 0.1502, indicating that Holt-Winters' method gives a better forecast. However, let's look at the forecast accuracy measures to make sure Holt-Winters' is the winner.

```
# The forecast accuracy measures for ARIMA
accuracy(forecast)
```

```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 15.74514 1408.406 1076.439 -0.9011266 7.535356 0.6748744
##                      ACF1
## Training set -0.01349079
```

```
# The forecast accuracy measures for HoltWinters
accuracy(ts_fore2)
```

```
##                     ME     RMSE      MAE       MPE     MAPE     MASE      ACF1
## Training set 179.7967 1563.491 1231.476 0.6970466 8.074457 0.772075 0.1359301
```

The forecast accuracy measures

Model	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
ARIMA	15.7	1408.4	1076.4	-0.9	7.5	0.67	-0.013
HoltWinters	179.8	1563.5	123.5	0.697	8.07	0.77	0.136

It is obvious from the graph that all of the results point for the Holt-Winters' method give the best of these two methods. The Holt-Winters' method is the best for these data.

Bibliography

Hyndman, R. J. (2014). *Forecasting: Principles & Practice.* University of Western Australia.

Palachy, S. (2019, April 8). *Stationarity in time series analysis.* Retrieved from Towards Data Science: https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322

Salas, A. D. (n.d.). *A Language, not a Letter: Learning Statistics in R Intro R-programming.* Retrieved from Ademos: https://ademos.people.uic.edu/Chapter23.html#41_determining_stationarity_with_an_augmented_dickey-fuller_test

Shay, P. (2019, July 21). *Detecting stationarity in time series data.* Retrieved from Towards Data Science: https://towardsdatascience.com/detecting-stationarity-in-time-series-data-d29e0a21e638