What the Holt?!

Joshua Bela

October 31, 2019

1. Select a scientific, biomedical, business or other issue that appeals to you and go looking online for relevant time series data sets. The good news here is that there are tons of free and interesting time series data sets online. If you have problems locating them let me know and I will help. Tell me what issues you are investigating and cite the data sets so that I know where they came from. I want you to select at **least two different data sets** and perform the following analyses on each data set. For the first data set find one **without seasonality** in it. You will use this data set in steps 1 through 8. **The second of these data sets has to have seasonality in it and will only be used in step 9.** Suck the data set into your platform(s) of choice.

**We will perform a time series analysis of the average sale price in USD of American homes from April of 1996 to December of 2017.**

import data

# import libraries  
library(randomForest)#na.roughfix

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

library(forecast)#Acf(); subset()

## Registered S3 method overwritten by 'xts':  
## method from  
## as.zoo.xts zoo

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

## Registered S3 methods overwritten by 'forecast':  
## method from   
## fitted.fracdiff fracdiff  
## residuals.fracdiff fracdiff

library(tseries)#kpss.test()  
library(dplyr)#select()

##   
## Attaching package: 'dplyr'

## The following object is masked from 'package:randomForest':  
##   
## combine

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(aTSA)#adf.test(); arch.test()

##   
## Attaching package: 'aTSA'

## The following objects are masked from 'package:tseries':  
##   
## adf.test, kpss.test, pp.test

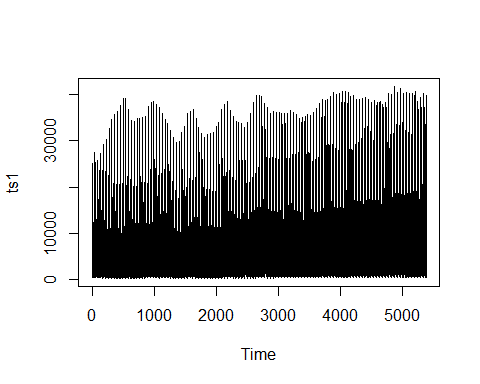
## The following object is masked from 'package:forecast':  
##   
## forecast

## The following object is masked from 'package:graphics':  
##   
## identify

library(forecast)#arima(); forecast.Arima()  
library(stats)#Box.test(x, lag = 1, type = c("Box-Pierce", "Ljung-Box"), fitdf = 0)  
  
# import data  
data = read.csv("data/State\_time\_series.csv")  
# filter variables  
data\_sub = data %>% select(Date, Sale\_Counts\_Seas\_Adj)  
# filter NA values  
data\_sub = data\_sub[!is.na(data\_sub$Date) & !is.na(data\_sub$Sale\_Counts\_Seas\_Adj),]  
# convert to time series object  
ts1 = ts(data\_sub$Sale\_Counts\_Seas\_Adj)

1. Plot out your time series variable.

# plot time series  
plot.ts(ts1)

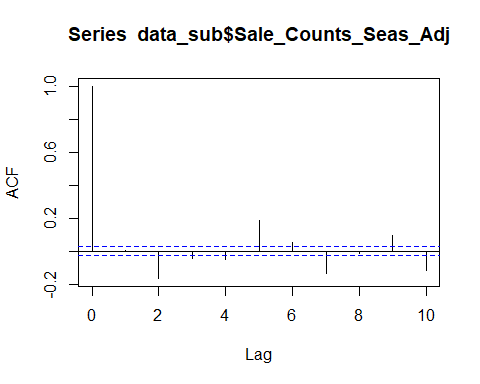


Tell me using your Mark I eyeball whether or not you think the time series data set is stationary in terms of constant mean and also constant variance.

**It appears as though the mean is increasing and the variance is increasing in turn. It appears non-stationary.**

1. Plot the ACF for the time series data set.

acf(data\_sub$Sale\_Counts\_Seas\_Adj, lag.max = 10)



Looking at ACF, does it look like there may be a trend or non-constant mean for each time series?

**I can’t confidently say visually. The only pattern that I detect is a decrease in absolute value, but I don’t see cycles.**

1. Now let’s examine the time series data set using unit root tests. First use the KPSS test for the time series data set and

kpss.test(ts1)

## KPSS Unit Root Test   
## alternative: nonstationary   
##   
## Type 1: no drift no trend   
## lag stat p.value  
## 16 86.5 0.01  
## -----   
## Type 2: with drift no trend   
## lag stat p.value  
## 16 13.8 0.01  
## -----   
## Type 1: with drift and trend   
## lag stat p.value  
## 16 0.31 0.01  
## -----------   
## Note: p.value = 0.01 means p.value <= 0.01   
## : p.value = 0.10 means p.value >= 0.10

tell me if the test suggests if there is a constant mean or not.

**Based on the p-value for the KPSS test, which is less than .05, the time series has a constant mean.**

Then see if you can confirm your KPSS evaluation using the Augmented Dickey Fuller (ADF) or the ADF-GLS test for each time series.

adf.test(ts1)

## Augmented Dickey-Fuller Test   
## alternative: stationary   
##   
## Type 1: no drift no trend   
## lag ADF p.value  
## [1,] 0 -45.09 0.01  
## [2,] 1 -31.56 0.01  
## [3,] 2 -22.01 0.01  
## [4,] 3 -17.48 0.01  
## [5,] 4 -11.56 0.01  
## [6,] 5 -9.67 0.01  
## [7,] 6 -9.21 0.01  
## [8,] 7 -8.03 0.01  
## [9,] 8 -6.61 0.01  
## [10,] 9 -7.08 0.01  
## Type 2: with drift no trend   
## lag ADF p.value  
## [1,] 0 -73.0 0.01  
## [2,] 1 -61.1 0.01  
## [3,] 2 -49.2 0.01  
## [4,] 3 -44.4 0.01  
## [5,] 4 -31.7 0.01  
## [6,] 5 -28.2 0.01  
## [7,] 6 -28.7 0.01  
## [8,] 7 -26.4 0.01  
## [9,] 8 -22.7 0.01  
## [10,] 9 -25.6 0.01  
## Type 3: with drift and trend   
## lag ADF p.value  
## [1,] 0 -74.4 0.01  
## [2,] 1 -63.1 0.01  
## [3,] 2 -51.6 0.01  
## [4,] 3 -47.4 0.01  
## [5,] 4 -34.3 0.01  
## [6,] 5 -30.9 0.01  
## [7,] 6 -31.9 0.01  
## [8,] 7 -29.9 0.01  
## [9,] 8 -26.1 0.01  
## [10,] 9 -30.0 0.01  
## ----   
## Note: in fact, p.value = 0.01 means p.value <= 0.01

What is your decision concerning constant mean?

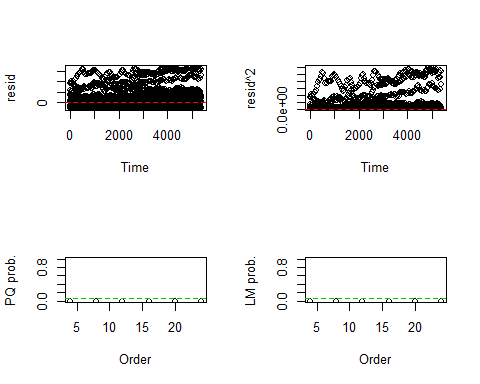
**According to the ADF test, the time series has a constant mean.**

**The mean is constant.**

1. Review the decisions in step #4. If the test suggests that there is a non-constant mean then use differencing to create a new differenced variable for the time series data set.  
   **N/A**
2. Test each of the time series data sets for constant variance using the ARCH test (GRETL does this nicely). Tell me which ones might have issues with constant variance and so not be so nicely stationary. Note that we will not do anything about this issue for the moment, but it’s good to know.

arch.test(ts1 %>% arima())

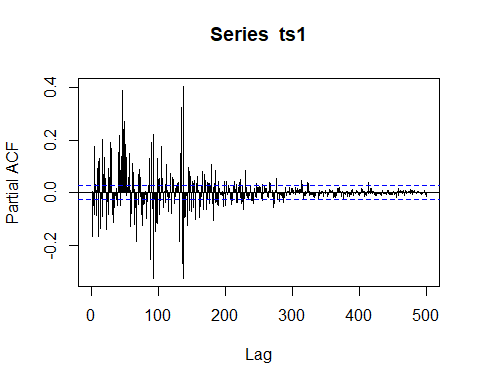
## ARCH heteroscedasticity test for residuals   
## alternative: heteroscedastic   
##   
## Portmanteau-Q test:   
## order PQ p.value  
## [1,] 4 24 8.17e-05  
## [2,] 8 690 0.00e+00  
## [3,] 12 749 0.00e+00  
## [4,] 16 836 0.00e+00  
## [5,] 20 911 0.00e+00  
## [6,] 24 960 0.00e+00  
## Lagrange-Multiplier test:   
## order LM p.value  
## [1,] 4 8256 0  
## [2,] 8 1944 0  
## [3,] 12 1008 0  
## [4,] 16 714 0  
## [5,] 20 567 0  
## [6,] 24 432 0



**Based on the p-values from the ARCH test, which are less than .05, there is non-constant variance, so the time series is not purely stationary.**

1. Plot the PACF for the time series data sets. Using the combined information from the ACF you plotted earlier along with the information in the PACF, tell me if you see autoregressive and/or moving average processes in the data set. To help with interpretation you may want to refer to online resources – here is a decent resource from Duke University <https://people.duke.edu/~rnau/411arim3.htm> or Penn State <https://onlinecourses.science.psu.edu/stat510/node/64>

pacf(ts1, lag.max = 500)



**Concerning the ACF plot, there are no significant lags. Concerning the PACF plot, the lags are decreasing geometrically, although the trend is only visible with n observed in hundreds. Conclusion: This is a moving average model with order q = 1.**

1. For your time series data set, experiment with different ARIMA models for them. As you try them, list out the results of the various models and
2. Comment on how each one is working and compare it to the previous model using various metrics such as SBC, BIC, Box Leung, etc. Most students end up creating a small table with these statistics across the models tried so it is easy to compare them.
3. Pick one of the models as your favorite and tell me why you like that one the best.

df = data.frame()  
arimas = list()  
for(p in 6:8){  
 for(d in 0:2){  
 for(q in 6:8){  
 arima1 = Arima(ts1, order = c(p,d,q))  
 arimas[[length(arimas) + 1]] = arima1  
 df = df %>% rbind(c(  
 p, d, q, arima1$bic, Box.test(arima1$fitted, lag = 1, type = "Ljung-Box", fitdf = 0)$p.value  
 %>% round(digits = 2)  
 ))  
 }  
 }  
}  
colnames(df) = c("p", "d", "q", "BIC", "Ljung-Box")  
df = df[order(df$BIC),]  
print(df)

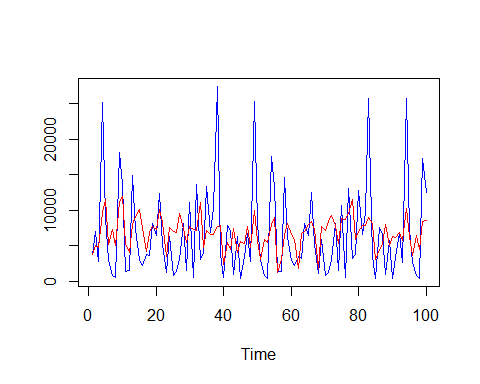
## p d q BIC Ljung-Box  
## 6 6 1 8 110070.3 0.00  
## 14 7 1 7 110097.8 0.00  
## 15 7 1 8 110245.1 0.00  
## 5 6 1 7 110348.4 0.00  
## 21 8 0 8 110350.7 0.00  
## 13 7 1 6 110374.8 0.64  
## 23 8 1 7 110379.6 0.00  
## 4 6 1 6 110381.6 0.38  
## 22 8 1 6 110394.5 0.00  
## 3 6 0 8 110400.2 0.05  
## 11 7 0 7 110401.1 0.86  
## 24 8 1 8 110405.5 0.00  
## 1 6 0 6 110414.1 0.47  
## 20 8 0 7 110417.6 0.00  
## 19 8 0 6 110434.8 0.00  
## 12 7 0 8 110446.6 0.00  
## 10 7 0 6 110487.3 0.00  
## 2 6 0 7 110494.8 0.00  
## 27 8 2 8 110794.1 0.00  
## 26 8 2 7 110919.2 0.00  
## 17 7 2 7 110950.4 0.00  
## 25 8 2 6 111043.3 0.00  
## 18 7 2 8 111142.0 0.00  
## 16 7 2 6 111193.9 0.51  
## 8 6 2 7 111261.4 0.95  
## 9 6 2 8 111379.8 0.00  
## 7 6 2 6 111385.1 0.00

**I built nested for loops to try multiple combinations of p, d, and q, then printed a table with each model’s BIC and Ljung-Box score, sorted in descending order of BIC.**

**The best model, according to the BIC and Ljung-Box statistics, is ARIMA(6, 1, 8).**

1. Plot the observed versus fitted data for the time series data set and comment on how well the model seems to be working

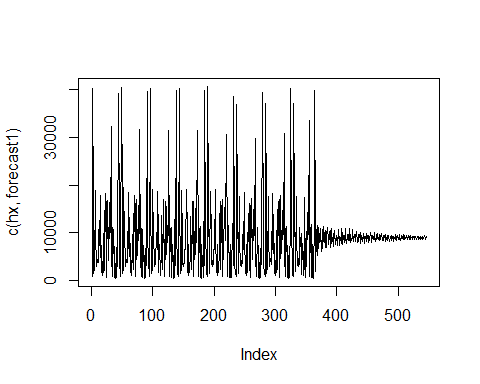
arima1 = Arima(ts1, order = c(df$p[1],df$d[1],df$q[1]))  
  
arima1$x %>% subset(start = 1, end = 100) %>% ts.plot(  
 arima1$fitted %>% subset(start = 1, end = 100),  
 col = c("blue", "red")  
)



**Considering the variability of the original data, this model appears to perform admirably well, although it can’t predict the extreme divergences from the mean that happen frequently with this data set.**

1. Forecast out your favorite model for the next 6 time periods and plot your time series plus the forecasted data. Does it look good or funky?

hx = ts1[(length(ts1) - 365.25\*1):length(ts1)]  
  
forecast1 = (forecast::forecast(arima1, h = 30\*6))$mean %>% as.numeric()  
  
plot(c(hx, forecast1), type = "l")



**It’s okay. It has a hard time predicting extreme oscillations, as the original data is unpredictably volatile. There’s not enough predictive data to account for this variance, but it’s doing the best it can. Moreover, this problem of informational dearth worsens the further out the forecast is, since forecasting further out is more difficult.**