Kiriakos\_MSDS 664\_X70\_Week 6: Timeseries Analysis

Cathy Kiriakos

April 14, 2020

## Week 6: Timeseries Analysis

The following data set is the sales of new one-family houses, USA, from 1963. (United States Census Bureaus)

We will start by loading our dataset as shown below, I did remove the top columns and convert the data that we’d be using for this timeseries analysis to csv as shown below:

For some exploratory analysis:

str(histCons)

## 'data.frame': 230 obs. of 6 variables:  
## $ Period : Factor w/ 230 levels "","1 May reflect revisions not distributed to months. ",..: 3 4 5 6 7 8 9 10 11 12 ...  
## $ United.States: Factor w/ 207 levels "","103,400","106,000",..: 43 50 44 51 51 52 52 56 62 57 ...  
## $ Northeast : Factor w/ 207 levels "","105,700","108,800",..: 47 46 38 46 45 39 44 54 52 55 ...  
## $ Midwest : Factor w/ 214 levels "","103,000","104,000",..: 47 48 49 66 57 67 58 77 89 77 ...  
## $ South : Factor w/ 216 levels "","100,000","100,900",..: 50 39 40 39 48 50 50 49 58 47 ...  
## $ West : Factor w/ 202 levels "","104,200","113,700",..: 43 44 46 47 48 52 53 56 57 60 ...

This reminded me that I need to delete some notes at the bottom of the data set:

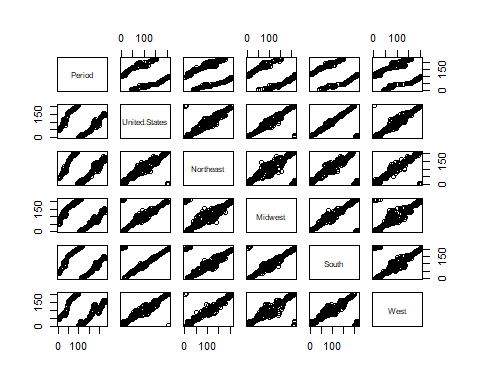
histCons<-histCons[-c(229,230),]  
tail(histCons)

## Period United.States Northeast Midwest South West  
## 223 2018Q3 330,900 503,700 292,100 296,100 408,600  
## 224 2018Q4 322,800 519,700 300,300 291,600 404,300  
## 225 2019Q1 313,000 480,300 288,700 280,000 402,000  
## 226 2019Q2 322,500 453,500 273,800 292,400 411,400  
## 227 2019Q3 318,400 543,400 294,700 289,900 399,600  
## 228 2019Q4 324,500 472,500 286,300 283,500 407,800

Beautiful

We can first get a look at the timeseries looking at the south with a plot, which will be useful to compare against the timeseries plot, we can see a strong linear relationship between time and housing price.

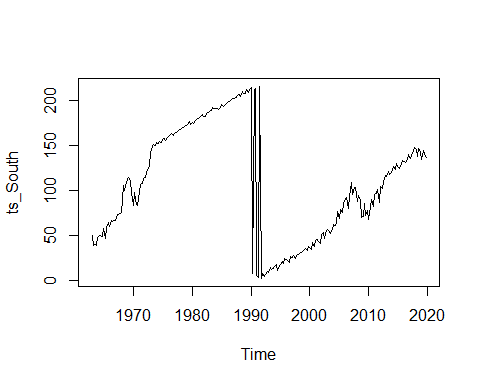
plot(histCons)

 The first step will be to create the timeseries, where we’re analyzing the south and our time frequency is 4 indicating quarterly:

ts\_South<-ts(histCons$South, start = c(1963,1), frequency = 4)

Now we will get a look at our time series plot to get an understanding of trending:

plot.ts(ts\_South)

 So were seeing a drastic shift from 1990 to rouglhy 1992; looking at the data frame it becomes apparant that as compared to prior trending with a continous upward swing,the early nineties had some backwards movements from year over year moving on average $10k or so. This is likely the culprit of the drastic drop off in the time series plot.

Upon further investigation of these results I found in an Forbes article the following snippet that helps us understand that movement:

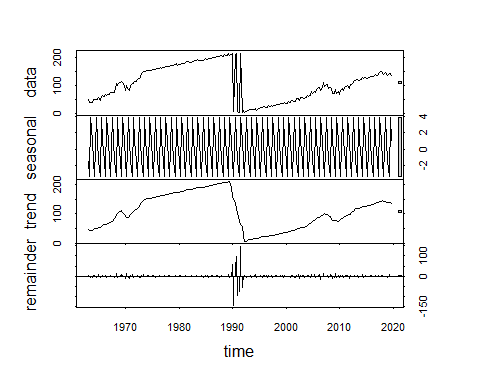
“Real home prices peaked in 1989, the recession hit in 1990, home prices fell 7% from the peak until the end of 1990, the recession ended in the spring of 1991 but real U.S. home prices continued to fade for years until they bottomed out in 1997, down 14% from the 1989 peak eight years earlier.” <https://www.forbes.com/sites/johnwake/2018/11/02/the-next-housing-bust/#1352e30918b7>

So this movement is not not an error.

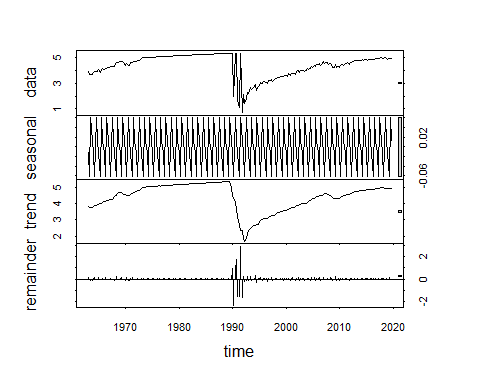
We can see an overall continuing upward sloping trend for pricing values which makes sense. We can also see the impact of the great financial crisis with the slump in prices from 2008-2010. I think we are observing some seasonality due to an increaes in demand over the summer months,leading to increased prices; followed by a decline over winter months when houses are harder to sell, and sellers are willing to drop prices as a result of the decline in demand by buyers.

Since we have some seaonality in our data we can use the stl() function to seasonally decomopose our data:

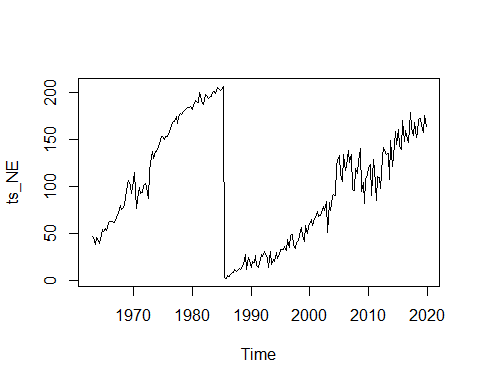
fit<-stl(ts\_South, s.window = "period")  
plot(fit)

 We may also want to manage the multiplicative effects by transforming into a series with additive effects, this will be our new time series as shown below, and we will run through and look at the seasonal decomposition after this log transformation of the time series:

newTsSouth <- log(ts\_South)  
fit2<-stl(newTsSouth, s.window = "period")  
plot(fit2)

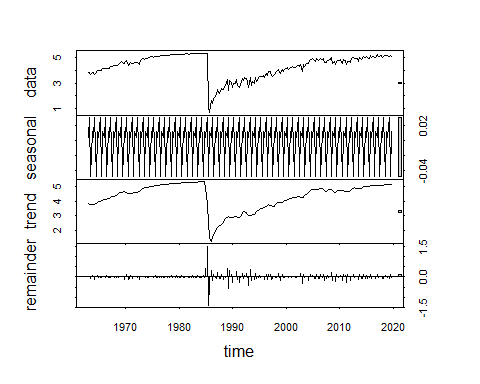
 This smoothed out our overall trending. It makes me wonder if the South’s housing market was more impacted than others as a result of the dot com burst of the nineties. So we can make the same investigations looking at the Northeast.

ts\_NE<-ts(histCons$Northeast, start = c(1963,1), frequency = 4)  
plot(ts\_NE)

 What’s interesting here is it looks like we’re seeing a drop off more correlated with the 1987 recession; we still see the same run up due to inflation and interest rates running rampant in the early eighties; but we see the steep drop off sooner.

Lets continue to decompose our data in the east for seaonality, and continue on with investigations of the North Easts housing data as I’m seeing some more interesting trending; deeper declines due to the great recession in 2008-2010.

ts\_NormNE <- log(ts\_NE)  
fit3<-stl(ts\_NormNE, s.window = "period")  
plot(fit3)

 Now we can conduct an Augmented Dickey Fuller test on the timeseries. We will set our lags to 0 and with a trend and intercept. The Augmented Dickey Fuller test: null hypothesis is that the data are non-stationary and non-seasonal.

library(tseries)  
adf.test(ts\_NormNE,  
 alternative = c("stationary","explosive"),  
 k = trunc((length(x)-1)^(1/3)))

Our North East housing data is stationary, and important assumption for our timeseries analysis.

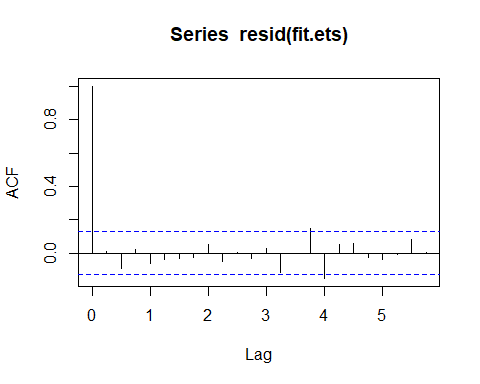
Moving on to our forecast, we will use the ETS package, standing for Error, Trend, Seasonal. The function automatically chooses a model by default using AIC, AICs or BIC. It can handle any combination of trend seasonality and damping. We will run our model and get a summary of our results below:

fit.ets <- ets(ts\_NormNE)  
summary(fit.ets)

## ETS(A,N,N)   
##   
## Call:  
## ets(y = ts\_NormNE)   
##   
## Smoothing parameters:  
## alpha = 0.8648   
##   
## Initial states:  
## l = 3.8441   
##   
## sigma: 0.3457  
##   
## AIC AICc BIC   
## 757.5257 757.6329 767.8138   
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 0.006408129 0.3441802 0.1456528 -1.493837 5.827662 0.5814084  
## ACF1  
## Training set 0.01214509

The acd(resid(fit.ets)) looks good, we’ve got white noise:

acf(resid(fit.ets))

 Lets get perform the box test, which will indicate if we have independent or no correlation between successive errors. If not rejected (p-value greater than alpha), no need to improve the model. (aka “portmanteau” test – or Box-Pierce test)

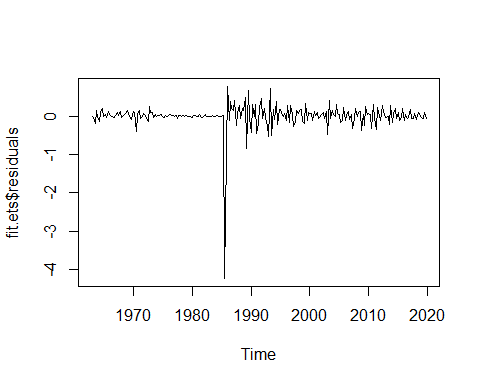
H0: the model does not exhibit lack of fit H1: the model exhibits lack of fit

BoxTest <- Box.test(fit.ets$residuals, type = "Ljung-Box")  
BoxTest

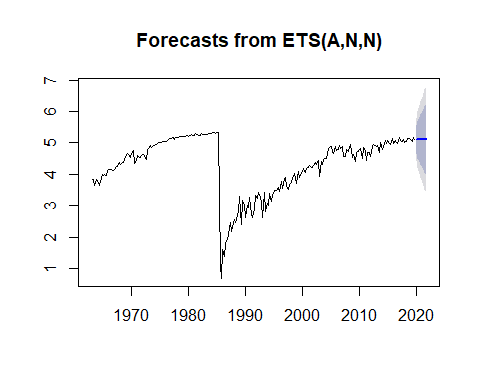
##   
## Box-Ljung test  
##   
## data: fit.ets$residuals  
## X-squared = 0.034075, df = 1, p-value = 0.8535

The p value is greater than 0.05 then the residuals are independent; this is what we want for the model to be correct

plot.ts(fit.ets$residuals)

 Alright, now we can move forward to our 8 period forecast:

pred.ets <- forecast(fit.ets, h=8)  
plot(pred.ets)

 We can get a look at the accuracy of the model; the mean error has a farily low value summarizing the average of the errors, so that would be a good indicator of the model. Our Root Mean Squared Error is .34

For context of these values I obtained this information from Stack to outline what each of the results indicate:

ME: Mean Error – The mean error is an informal term that usually refers to the average of – all the errors in a set. An “error” in this context is an uncertainty in a measurement, – or the difference between the measured value and true/correct value

RMSE: Root Mean Squared Error 2.1 MAE: Mean Absolute Error – The MAE measures the average magnitude of the errors in a set of forecasts, – without considering their direction. It measures accuracy for continuous variables. – The RMSE will always be larger or equal to the MAE; – the greater difference between them, the greater the variance in the individual errors – in the sample. If the RMSE=MAE, then all the errors are of the same magnitude – Both the MAE and RMSE can range from 0 to ∞. – They are negatively-oriented scores: Lower values are better.

MPE: Mean Percentage Error – the mean percentage error (MPE) is the computed average of – percentage errors by which forecasts of a model differ from actual values of the – quantity being forecast.

MAPE: Mean Absolute Percentage Error – The MAPE, as a percentage, only makes sense for values where divisions and – ratios make sense. It doesn’t make sense to calculate percentages of temperatures – MAPEs greater than 100% can occur. – then this may lead to negative accuracy, which people may have a hard time understanding – Error close to 0% => Increasing forecast accuracy – Around 2.2% MAPE implies the model is about 97.8% accurate in predicting the next 15 observations.

MASE: Mean Absolute Scaled Error – Scale invariance: The mean absolute scaled error is independent of the scale of the data, – so can be used to compare forecasts across data sets with different scales. – ok for scales that do not have a meaningful 0, – penalizes positive and negative forecast errors equally – Values greater than one indicate that in-sample one-step forecasts from the naïve method perform better than the forecast values under consideration. – When comparing forecasting methods, the method with the lowest MASE is the preferred method.

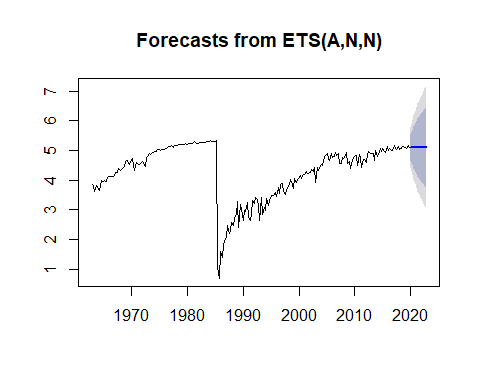
ACF1: Autocorrelation of errors at lag 1.’ – it is a measure of how much is the current value influenced by the previous values in a time series. – Specifically, the autocorrelation function tells you the correlation between points separated by various time lags – the ACF tells you how correlated points are with each other, – based on how many time steps they are separated by. That is the gist of autocorrelation, – it is how correlated past data points are to future data points, for different values of the time separation. – Typically, you’d expect the autocorrelation function – to fall towards 0 as points become more separated (i.e. n becomes large in the above notation) – because its generally harder to forecast further into the future from a given set of data. – This is not a rule, but is typical. – ACF(0)=1 (all data are perfectly correlated with themselves), – ACF(1)=.9 (the correlation between a point and the next point is 0.9), ACF(2)=.4 – (the correlati.

accuracy(fit.ets)

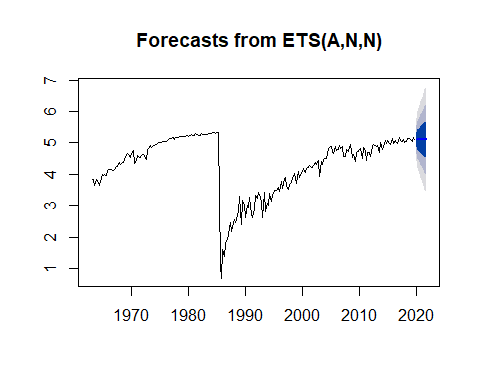
## ME RMSE MAE MPE MAPE MASE  
## Training set 0.006408129 0.3441802 0.1456528 -1.493837 5.827662 0.5814084  
## ACF1  
## Training set 0.01214509

Now we will forecast out for 12 periods as shown below:

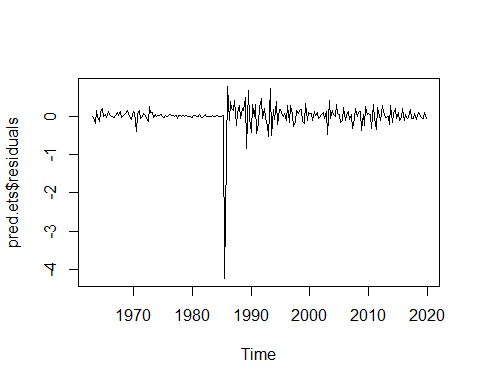
pred.ets12 <- forecast(fit.ets, h=12)  
plot(pred.ets12)

 Now we will plot forecast with forecast intervals as shown below:

plot (forecast(fit.ets, level=c(50,80,95)))

 Now we can get a look at our residuals:

plot.ts(pred.ets$residuals)

 We will now pick the ARIMA model (automatically) with auto.arima()

arimaTsNe <- auto.arima(ts\_NormNE)  
arimaTsNe

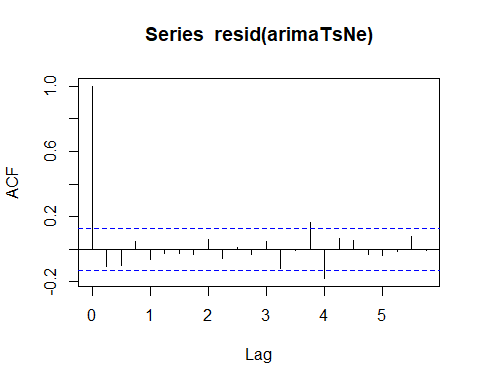
## Series: ts\_NormNE   
## ARIMA(0,1,0)   
##   
## sigma^2 estimated as 0.1208: log likelihood=-82.16  
## AIC=166.32 AICc=166.34 BIC=169.75

Now getting a look at our Box Test on the Arima model

BoxTest <- Box.test(arimaTsNe$residuals, type = "Ljung-Box")  
BoxTest

##   
## Box-Ljung test  
##   
## data: arimaTsNe$residuals  
## X-squared = 2.7115, df = 1, p-value = 0.09963

acf(resid(arimaTsNe))

 Looking at the results of our Auto Arima model using the box test we can see that our p-values are low; because x-squared values increase as the sample autocorrelations of the residuals get larger; we’re seeing a lack of independance. Given that it is still above 0.05; the model is still useful; but not as strong as our ETS model.

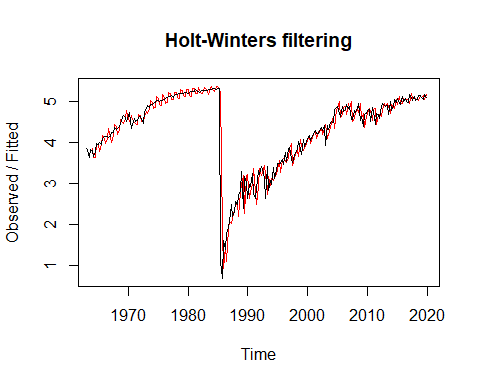
Last but not least we will get a look at our Holt-Winters model and see how the results stack up against Arima and our ETS:

predHW <- HoltWinters(ts\_NormNE)  
predHW

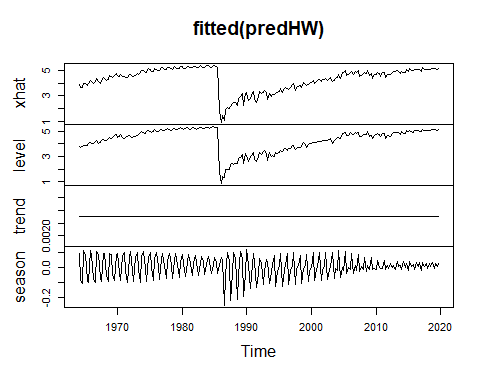
## Holt-Winters exponential smoothing with trend and additive seasonal component.  
##   
## Call:  
## HoltWinters(x = ts\_NormNE)  
##   
## Smoothing parameters:  
## alpha: 0.8401353  
## beta : 0  
## gamma: 0.3309222  
##   
## Coefficients:  
## [,1]  
## a 5.083718743  
## b 0.002765484  
## s1 -0.012059640  
## s2 0.018461525  
## s3 0.013374821  
## s4 0.024393860

Our alpha, beta and gamma values are 0.84, 0, and 0.33. The alpha our base value is farily high indicating that that a higher value is being placed on the most recent observations, which makes sense. We wouldn’t want to weight the 1980s uniformly with the 2020’s when it comes to housing values. Beta, our trend value is zero, telling us that we do not want to weight recent trending with older trends, which also makes sense. Lastly our Gamma is our seasonal component, a value of 0.33 tells us that our model is taking seasonality into account but not weighting it very heavily.

plot(predHW)



plot(fitted(predHW))

 Now we can review the coefficients

coefficients(predHW)

## a b s1 s2 s3 s4   
## 5.083718743 0.002765484 -0.012059640 0.018461525 0.013374821 0.024393860

From your analysis, what year do home prices reach its peak? Housing prices reach their peak at the hight of interest rates during the late eighties. We subsequently see the same level of decline following black friday 1987 market crash; where we see ebs and flows, including the second wave of declines observed in the great financial crisis of 2008.

What are your insights and/or suggested actionable decision? for continued modeling of housing prices and forecasting; I would reccomend leveraging the ETS model as it has provided the best fit in our analysis.