HW8.2

Todd Weigel

March 6, 2019

# HW8.11.2

This problem is to plot the IBM closing stock price of IBM over a period of time, and examine it’s lack of “stationariness”.

First lets just plot the raw data. From the plot, it is obvious (as expected) that the data is not stationary. There is an upward trend in the first third, then an accelerating downward trend in the second third, while the last third doesn’t exhibit a clear trend. The variability seems more pronounced in the first half of the graph, but that is perhaps only subjective. There doesn’t seem to be obvious seasonality, or cyclicity.

library(fpp2)

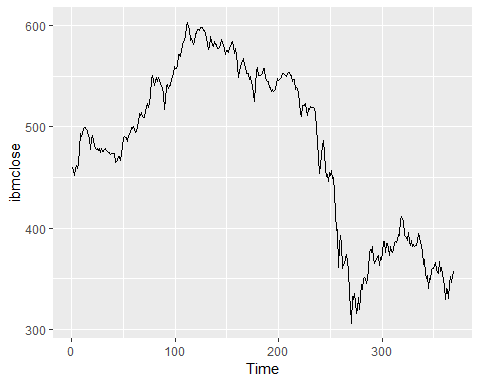
## Loading required package: ggplot2

## Loading required package: forecast

## Loading required package: fma

## Loading required package: expsmooth

autoplot(ibmclose)



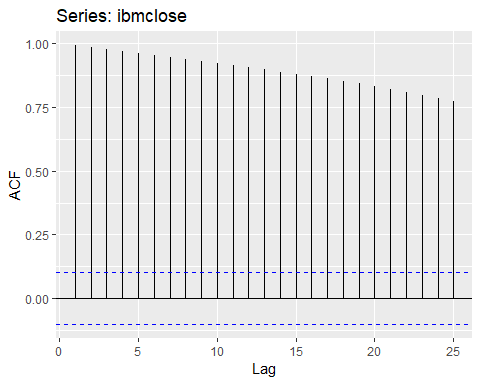
## ACF PACF Plots

Below we print the plots of the ACF (auto correlation function) and the PACF (partial ACF) to further examine the non-stationariness as well as to how we might need to difference the series to make it stationary.

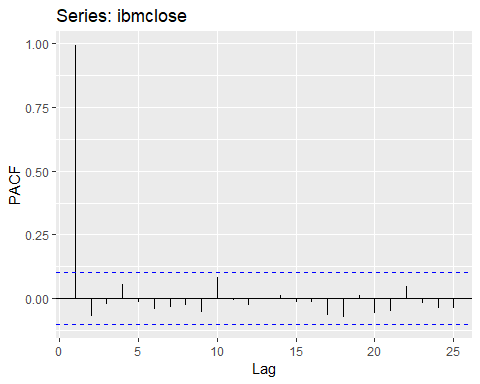
Looking at the ACF plot we see it is decaying slowly, a clear indication of non-stationary data. Also the value of R is quite large, further indicating the series isn’t stationary.

With the PACF model, we see 1 really large spike, this indicates differencing of 1 is probably all that is needed to make the data stationary.

ggAcf(ibmclose)

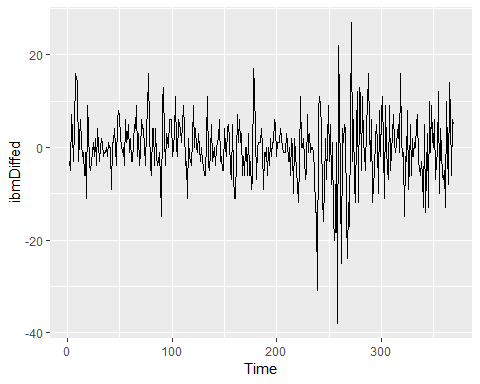


ggPacf(ibmclose)

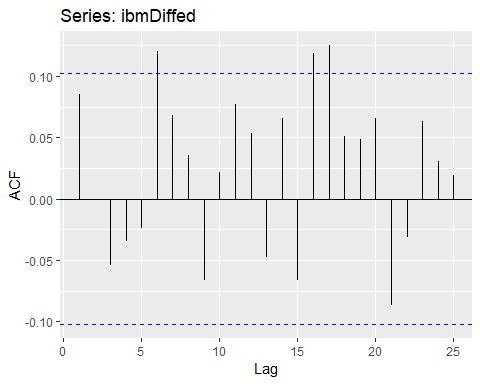
 ###Rerun After taking one difference.

Running the plots after we use the “diff” (difference) command, we see that data now doesn’t show upward/downward trend, but the variance looks like it might be slightly larger, towards the end of the sequence. In any case the ACF and PACF plots while still containing some spikes above the upper signifcance line, are only very slighly above, so the series is much closer to being a stationary series.

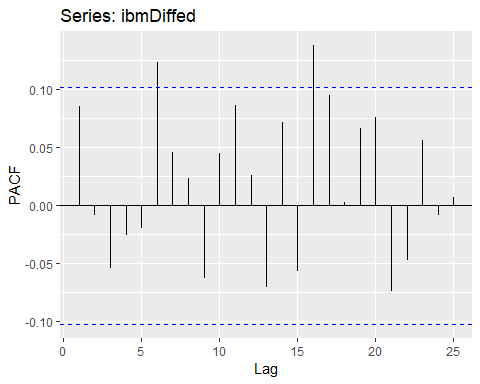
ibmDiffed <- diff(ibmclose)  
autoplot(ibmDiffed)



ggAcf(ibmDiffed)



ggPacf(ibmDiffed)

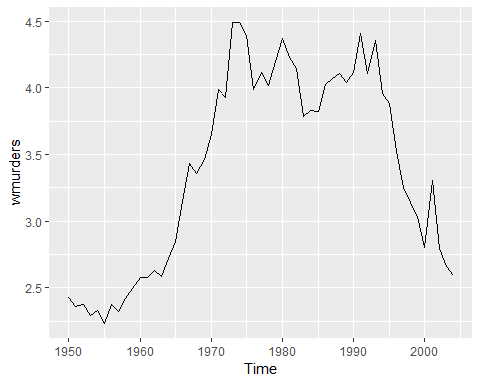


## 8.11.7

This problem deals with the dataset wmurders, which is a series containing the number of women murdered each year in the United States. First lets run some plots to examine the data.

From the raw plot of the data, we can definitely see obvious trend, with the murder rate climbing significantly through the 50’s and 60’s and then again with a strong decline from the early 90’s to the early 2000’s (until the end of the data). Seasonal differences would not apply as the data is yearly and it is unlikely there is seasonal cycles of multiple years.

autoplot(wmurders)



Unit Tests: Rather than just relying on subject “eyeballing” of the plotted data, let’s run some unit tests for more objectivity. We will run the KPSS (Kwiatkowski-Phillips-Schmidt-Shin) and the Augemented Dickey Fuller Tests. The results are below for both the intial raw data and the data differenced once.

library(urca)  
library(tseries)  
summary(ur.kpss(wmurders))

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 3 lags.   
##   
## Value of test-statistic is: 0.6331   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

summary(ur.kpss(diff(wmurders)))

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 3 lags.   
##   
## Value of test-statistic is: 0.4697   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

adf.test(wmurders)

##   
## Augmented Dickey-Fuller Test  
##   
## data: wmurders  
## Dickey-Fuller = -0.29243, Lag order = 3, p-value = 0.9878  
## alternative hypothesis: stationary

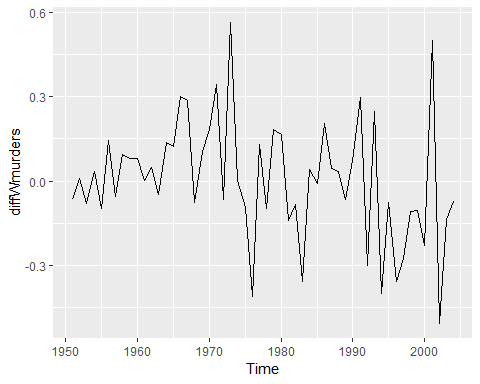
adf.test(diff(wmurders))

##   
## Augmented Dickey-Fuller Test  
##   
## data: diff(wmurders)  
## Dickey-Fuller = -3.7688, Lag order = 3, p-value = 0.02726  
## alternative hypothesis: stationary

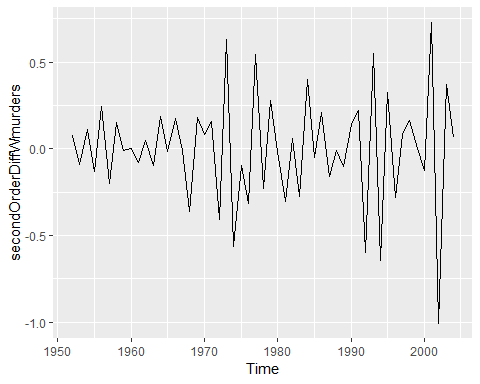
Interestingly, if we look at the KPSS test there doesn’t seem to be a huge need to difference at all, the test statistics is under 1; differencing once does make that number even smaller though. If we look at the ADF test though, the first result of 0.98 shows a strong need to difference, the result after differencing is much smaller at .0273, which indicates likely minimal trend, but we might wish to consider differencing twice to reduce further.

Anyhow, lets plot the differenced data (below). From the plot of the differenced data, we see that the trend has been removed, although there definitely still seems to be some increase in variation as time goes along, with small variation in the early stages and larger ones as we move forward in time. Plotting the 2nd order difference removes much of that variation, so perhaps 2 will be good for the “d” of the ARIMA model.

diffWmurders <- diff(wmurders)  
autoplot(diffWmurders)

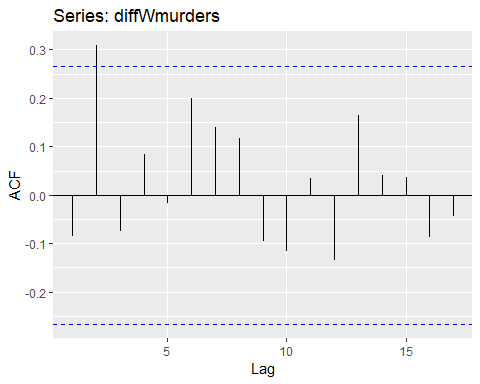


secondOrderDiffWmurders <- diff(diff(wmurders))  
autoplot(secondOrderDiffWmurders)

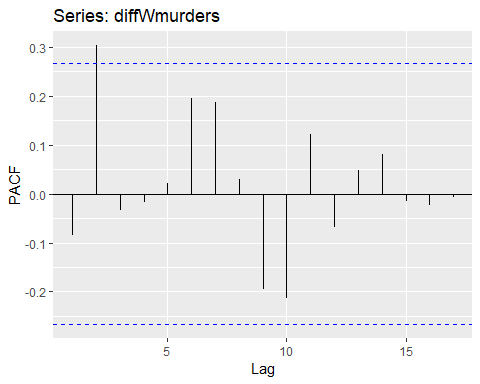


Now looking at the ACF and the PACF plots (below), we see one time containing a spike into the “significance” range. This was at time period 2 for both graphs. This suggests maybe having a “p” value of at least 1 for the ARIMA model.

ggAcf(diffWmurders)

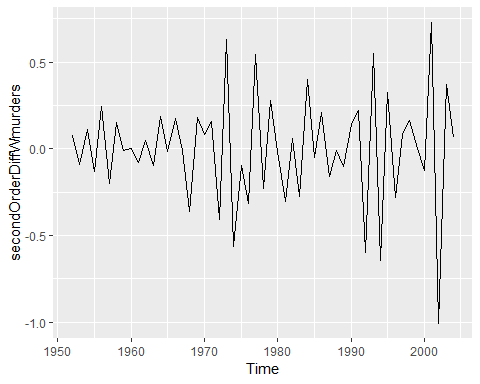


ggPacf(diffWmurders)

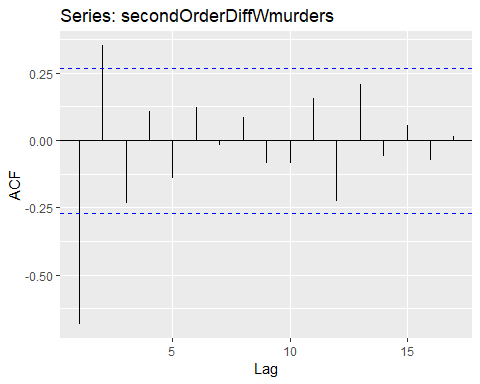


If we want to consider differencing a second time, let’s look at the two ACF plots. We see that the first lag in both plots is greater than negative 0.5, which may mean the plot is overdifferenced. But leaving that aside for now, we see with both plots that they are sinusoidal, indicating that both a p and q parameter would make sense. For the PACF, the first lag is significant so we should adding a parameter for the “q” or Moving Average (MA) part of the model. In the ACF the first two lags are significant, indicating when we difference twice we should add 2 to the “p” or Autoregressive ARIMA parameter, but since the second is so much smaller and guidance from <https://people.duke.edu/~rnau/arimrule.htm> suggests that if we have both MA and AR terms, we may want to reduce one or both of those by 1, since they may cancel each other out.

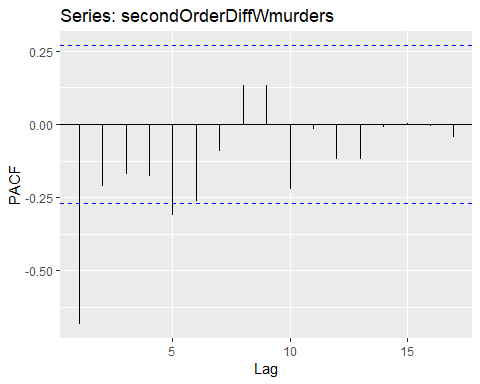
secondOrderDiffWmurders <- diff(diff(wmurders))  
autoplot(secondOrderDiffWmurders)



ggAcf(secondOrderDiffWmurders)



ggPacf(secondOrderDiffWmurders)



So, we are left with a few choices for models, if we go with 1 order of differencing as we saw potential of over differencing with 2 orders, our model is (1,1,1). If we choose 2 orders of differencing, we are left with either (1,2,2) as the likely choice, with the (1,2,1) also possible, if we lower the higher order MA term based on the comment from the Duke website. Let’s look at the 3 results:

Arima(wmurders, c(1,1,1))

## Series: wmurders   
## ARIMA(1,1,1)   
##   
## Coefficients:  
## ar1 ma1  
## -0.8255 0.6916  
## s.e. 0.1941 0.2333  
##   
## sigma^2 estimated as 0.04483: log likelihood=8.18  
## AIC=-10.36 AICc=-9.88 BIC=-4.39

Arima(wmurders, c(1,2,2))

## Series: wmurders   
## ARIMA(1,2,2)   
##   
## Coefficients:  
## ar1 ma1 ma2  
## -0.7677 -0.2812 -0.4977  
## s.e. 0.2350 0.2879 0.2762  
##   
## sigma^2 estimated as 0.04552: log likelihood=7.38  
## AIC=-6.75 AICc=-5.92 BIC=1.13

Arima(wmurders, c(1,2,1))

## Series: wmurders   
## ARIMA(1,2,1)   
##   
## Coefficients:  
## ar1 ma1  
## -0.2434 -0.8261  
## s.e. 0.1553 0.1143  
##   
## sigma^2 estimated as 0.04632: log likelihood=6.44  
## AIC=-6.88 AICc=-6.39 BIC=-0.97

Looking at these results, the model with the lowest AIC values is actually (1,1,1). So, the fact that has the lowest AIC, the fewest params (simpler) and the lowest variation and the least differencing, then this perhaps is the best model. But (not shown), in checking the residuals for the (1,1,1) they were not as satisfactory. So that leaves the other models, of the two, the (1,2,2) has very marginally better results for AIC and the BIC is noticably worse, and since the it is slighly more complex, I will suggest the best model is ARIMA(1,2,1),

## 8.11.7.B

Should a constant be part of the model? Normally if the model is differenced (which in our case is yes) and especially differenced more than once, then a constant is not included, so we shall not include a constant in our model.

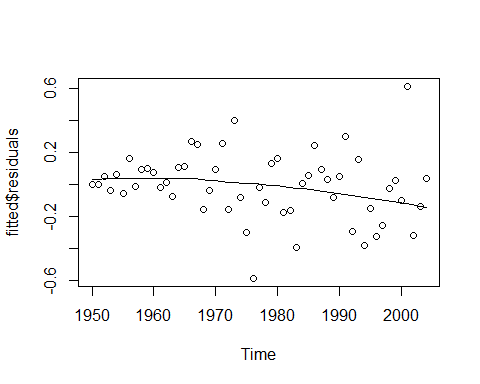
## 8.11.7.C

Using backshift notation, describe the differencing for our model. Since we are differencing twice, we need to compute . The full equation is:

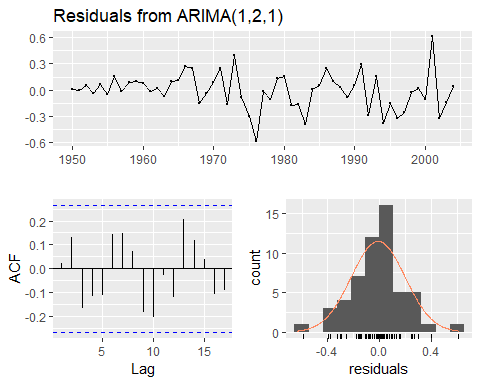
## 8.11.7.D

Fitting our ARIMA model in R and examing the residuals, see the plots below:

fitted <- Arima(wmurders, c(1,2,1))  
scatter.smooth(fitted$residuals)



checkresiduals(fitted)



##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(1,2,1)  
## Q\* = 12.419, df = 8, p-value = 0.1335  
##   
## Model df: 2. Total lags used: 10

Box.test(fitted$residuals, type="Ljung-Box")

##   
## Box-Ljung test  
##   
## data: fitted$residuals  
## X-squared = 0.027498, df = 1, p-value = 0.8683

The residuals look reasonable. They are evenly balanced around the mean. There are some outliers at 1977 and 2001, this is where the actual data had a sharp change in direction, i.e., there were a greatly fewer murders in 1977 and greatly more in 2001 than in the prior year. Not too suprisingly the model isn’t handling those well. The residuals do have a downward slope, so the model isn’t perfect. Also the ACF plot shows that the residuals are in tolerance, so basically these residuals are acceptable.

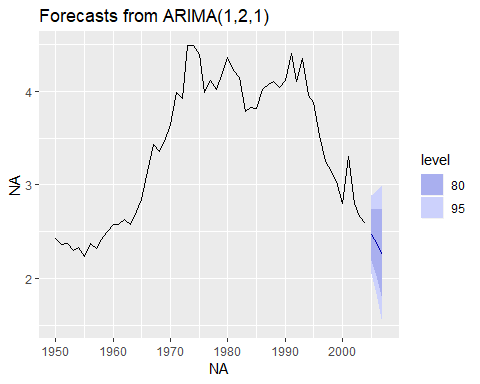
## 8.11.7.E and 8.11.7.F

Forecasting using our model.

forecast(fitted, h=3)

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 2005 2.470660 2.194836 2.746484 2.048824 2.892496  
## 2006 2.363106 1.986351 2.739862 1.786908 2.939304  
## 2007 2.252833 1.765391 2.740276 1.507354 2.998313

autoplot(forecast(fitted, h=3))



## 8.7.G

Comparing our model to what auto.ARIMA has generated.

Below we see that the auto.ARIMA function also chose a (1,2,1) model. Of course as noted way above, the (1,2,2) model could also be considered as an acceptable choice, and if one plots this (not done here for brevity, we see similar results to the above forecast plots).

auto.arima(wmurders)

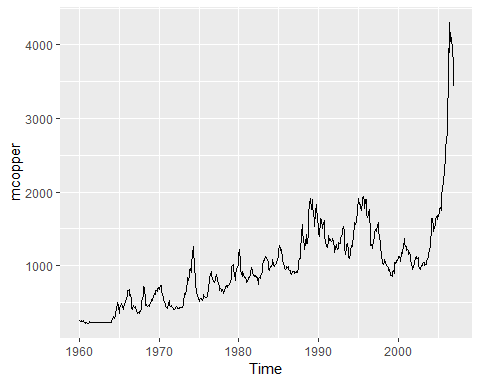
## Series: wmurders   
## ARIMA(1,2,1)   
##   
## Coefficients:  
## ar1 ma1  
## -0.2434 -0.8261  
## s.e. 0.1553 0.1143  
##   
## sigma^2 estimated as 0.04632: log likelihood=6.44  
## AIC=-6.88 AICc=-6.39 BIC=-0.97

# 8.11.12

## 8.11.12.A

This problem works on the mcopper dataset. First lets look at the data.

autoplot(mcopper)

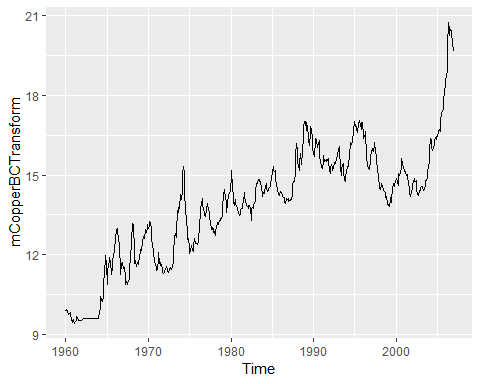


This looks like it needs a transformation, as the data shows variation over time. So let’s find an appropriate lambda value (0.191) for the transformation and plot that. We see doing this, that the variation is much less now from the early (left) portion of the time series to later entries.

lambda <- BoxCox.lambda((mcopper))  
lambda

## [1] 0.1919047

mCopperBCTransform <- BoxCox(mcopper, lambda)  
autoplot(mCopperBCTransform)



## 8.12.B

Now, let’s find an ARIMA function that will allow us to forecast. Using the R auto.Arima function we see that it predicts that an (0,1,1) model is appropriate.

auto.arima(mcopper, lambda = lambda)

## Series: mcopper   
## ARIMA(0,1,1)   
## Box Cox transformation: lambda= 0.1919047   
##   
## Coefficients:  
## ma1  
## 0.3720  
## s.e. 0.0388  
##   
## sigma^2 estimated as 0.04997: log likelihood=45.05  
## AIC=-86.1 AICc=-86.08 BIC=-77.43

## 8.12.C

Let’s experiment and see if other models might work better. First let’s adjust to the auto.Arima’s parameters to allow a deeper search of models.

auto.arima(mcopper, lambda = lambda, stepwise = FALSE, approximation = FALSE)

## Series: mcopper   
## ARIMA(0,1,1)   
## Box Cox transformation: lambda= 0.1919047   
##   
## Coefficients:  
## ma1  
## 0.3720  
## s.e. 0.0388  
##   
## sigma^2 estimated as 0.04997: log likelihood=45.05  
## AIC=-86.1 AICc=-86.08 BIC=-77.43

We see that the model returned was the same (0,1,1). So the autofunction even with deeper analysis, still feels (0,1,1) is appropriate.

We can do our own analysis (like in 8.11.7 above). Doing the unit tests, we see that both the KPSS and Augmented Dickey fuller show lots of trend without differencing as KPSS test statistic was well over 1 and the ADF results was substantially above 0.05 p-value, indicating trend. Applying an order of differencing of 1, removed any signficant trend from the models as shown by these two tests. So the having a “d” of one as indicated by auto.Arima seems correct.

summary(ur.kpss(mCopperBCTransform))

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 6 lags.   
##   
## Value of test-statistic is: 6.2659   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

summary(ur.kpss(diff(mCopperBCTransform)))

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 6 lags.   
##   
## Value of test-statistic is: 0.0573   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

adf.test(mCopperBCTransform)

##   
## Augmented Dickey-Fuller Test  
##   
## data: mCopperBCTransform  
## Dickey-Fuller = -2.9549, Lag order = 8, p-value = 0.1741  
## alternative hypothesis: stationary

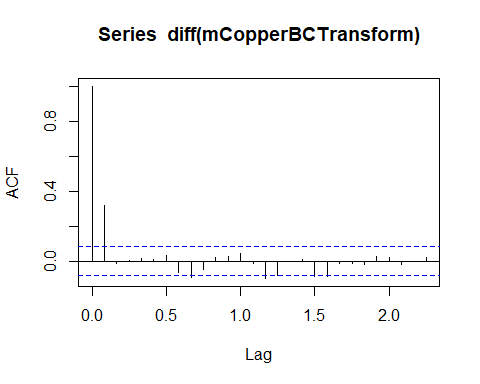
adf.test(diff(mCopperBCTransform))

## Warning in adf.test(diff(mCopperBCTransform)): p-value smaller than printed  
## p-value

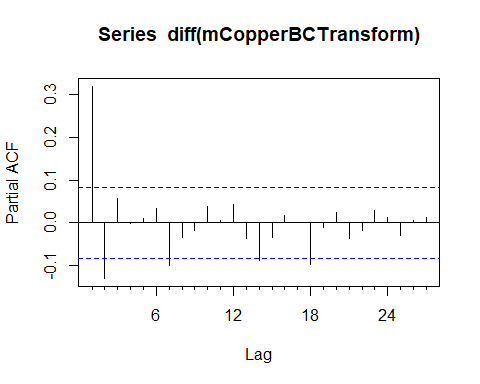
##   
## Augmented Dickey-Fuller Test  
##   
## data: diff(mCopperBCTransform)  
## Dickey-Fuller = -8.2555, Lag order = 8, p-value = 0.01  
## alternative hypothesis: stationary

Lets run ACF and PACF tests to see if the “p” and “q” values chosen by the auto.Arima seem accurate as well. Note due to substantial trend, we will only do the ACF tests on the data we already differenced.

acf(diff(mCopperBCTransform))



Pacf(diff(mCopperBCTransform))



We see on both the ACF and PACF two lags substantially above the significance threshold, which indicates trying a (2,1,2) model. Note there are a few other lags on each chart that are borderline significant, but making a model (5,1,5) is probably not wise, as it is likely we are overfitting at that point. So let’s try an ARIMA model of (2,1,2) and look at the results:

Arima(mcopper, c(2,1,2), lambda = lambda)

## Series: mcopper   
## ARIMA(2,1,2)   
## Box Cox transformation: lambda= 0.1919047   
##   
## Coefficients:  
## ar1 ar2 ma1 ma2  
## -0.9606 -0.1275 1.3337 0.4592  
## s.e. 0.1601 0.1194 0.1513 0.1113  
##   
## sigma^2 estimated as 0.05002: log likelihood=46.25  
## AIC=-82.5 AICc=-82.4 BIC=-60.84

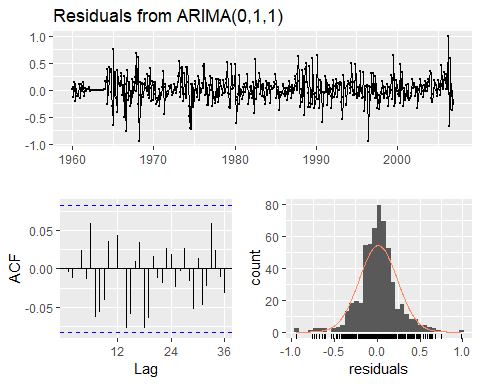
So the AIC is worse with this model. The log likelihood is pretty close, but once you add in the fact that we have 5 parameters vs 2 for the one the auto.arima found, then we see that AIC is noticeably lower for our choice. (Note AIC is a function of log likelihood and the paramters).

Therefore we will chose the (0,1,1) as the best model.

## 8.11.12.D

Using the (0,1,1) model let’s examine the residuals. The plots are below. We see that the residuals look like white noise. They are even around the 0 line with few outliers. The ACF plot shows no residuals in the “significant” category and the Ljung-Box test shows a very high p value, again indicating residuals are essentially “white noise”.

fitted <- Arima(mcopper, c(0,1,1), lambda = lambda)  
#fitted$residuals  
checkresiduals(fitted)



##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(0,1,1)  
## Q\* = 22.913, df = 23, p-value = 0.4659  
##   
## Model df: 1. Total lags used: 24

Box.test(fitted$residuals, type="Ljung-Box")

##   
## Box-Ljung test  
##   
## data: fitted$residuals  
## X-squared = 0.0099289, df = 1, p-value = 0.9206

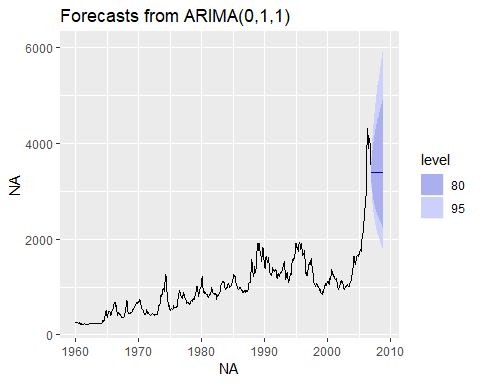
## 8.11.12.E

Forecasting the time series copper data. we see that the forecast is essentially a straight line. But the confidence levels are quite wide, meaning our predictions of a straight line is not that strong a prediction, the data could vary signifcantly.

forecast(fitted)

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## Jan 2007 3387.143 3188.095 3596.115 3086.619 3710.882  
## Feb 2007 3387.143 3054.898 3747.998 2889.994 3951.229  
## Mar 2007 3387.143 2964.977 3856.608 2759.364 4125.610  
## Apr 2007 3387.143 2893.279 3947.003 2656.458 4272.299  
## May 2007 3387.143 2832.337 4026.647 2569.883 4402.686  
## Jun 2007 3387.143 2778.707 4098.983 2494.387 4522.024  
## Jul 2007 3387.143 2730.462 4165.940 2427.034 4633.250  
## Aug 2007 3387.143 2686.396 4228.725 2365.987 4738.201  
## Sep 2007 3387.143 2645.693 4288.152 2310.004 4838.118  
## Oct 2007 3387.143 2607.770 4344.804 2258.202 4933.884  
## Nov 2007 3387.143 2572.198 4399.111 2209.926 5026.155  
## Dec 2007 3387.143 2538.644 4451.405 2164.672 5115.434  
## Jan 2008 3387.143 2506.849 4501.947 2122.046 5202.116  
## Feb 2008 3387.143 2476.603 4550.944 2081.731 5286.517  
## Mar 2008 3387.143 2447.735 4598.569 2043.468 5368.896  
## Apr 2008 3387.143 2420.103 4644.964 2007.042 5449.469  
## May 2008 3387.143 2393.588 4690.247 1972.273 5528.415  
## Jun 2008 3387.143 2368.088 4734.520 1939.008 5605.888  
## Jul 2008 3387.143 2343.517 4777.871 1907.114 5682.019  
## Aug 2008 3387.143 2319.799 4820.374 1876.478 5756.923  
## Sep 2008 3387.143 2296.867 4862.096 1847.001 5830.699  
## Oct 2008 3387.143 2274.664 4903.095 1818.596 5903.434  
## Nov 2008 3387.143 2253.139 4943.421 1791.184 5975.204  
## Dec 2008 3387.143 2232.247 4983.121 1764.699 6046.080

autoplot(forecast(fitted))

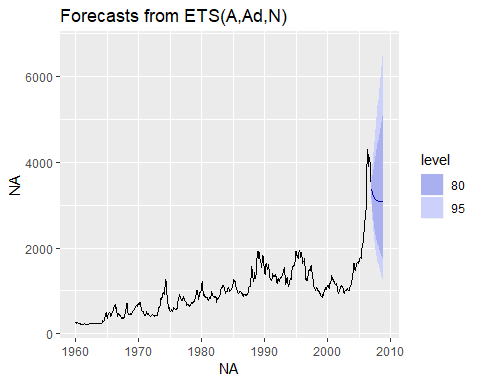


## 8.11.12.F

Compare the results of the ARIMA models to that achieved using ETS (error, trend, smoothing) model. From initial plotting, we see the forecast is quite noticeably lower ( 3100 vs 3400), and the 80% and 95% tolerance levels are even slightly wider that the ARIMA’s already wide intervals.

Comparing MSE’s from the two tests, we see that the ARIMA models errors are smaller, indicating that it is likely the ARIMA model is performing better.

autoplot(forecast(ets(mcopper, lambda = lambda)))



fets <- function(x,h, lambda)  
{  
 forecast(ets(x, lambda = lambda), h = h)  
}  
  
farima <- function(x,h, lambda)  
{  
 forecast(auto.arima(x, lambda = lambda), h = h)  
}  
  
etsCopper <- tsCV(mcopper, fets, h=1, lambda = lambda)  
arimaCopper <- tsCV(mcopper, farima, h=1, lambda = lambda)  
  
mean(etsCopper^2, na.rm=TRUE)

## [1] 6953.97

mean(arimaCopper^2, na.rm=TRUE)

## [1] 6198.188