Time Series Forecasting

K. Lass

1/12/2022

Data

Series: Federal Government: Current Expenditures Web Address: (https://fred.stlouisfed.org/series/NA000283Q) (1) The federal government budget includes three types of spending: mandatory, discretionary, and interest on debts owed by the nation. The majority of the Government's expenditures go towards Social Security, Medicare, and Medicaid programs as mandatory spending. (a) Units: Millions of Dollars, Not Seasonally Adjusted (b) Low Frequency: Quarterly (c) Number Observations: 48 (i) Duration: Q4 2006- Q3 2018 (ii) Within-Sample: Area of focus is Q4 2006- Q3 2016 (40 Obs) (iii) Post-Sample: This leaves 8 quarters (Q4 2016 - Q3 2018) after the area of focus for post-sample testing

```r
#read/attach data
data=read.csv("/stfm/dev2/m1kal01/Personal/data.csv")
attach(data)

#generate time series
Expenditure_Quarterly=ts(Expenditure, frequency=4)
print(Expenditure_Quarterly)
```

```
##       Qtr1    Qtr2    Qtr3    Qtr4
## 1   693643  718612  734747  732366
## 2   748213  762963  852722  803094
## 3   793027  814978  894346  884482
## 4   894576  927941  941628  943152
## 5   956392  946858  977285  944835
## 6   945771  937575  953318  940273
## 7   947808  933661  954582  951184
## 8   937463  953938  979777  988781
## 9   971506  979738 1008980 1022502
## 10 1003962 1015165 1033669 1048953
## 11 1042766 1054016 1056024 1064175
## 12 1079954 1096440 1117081 1130728
```

```r
#Create size restrictions
N=length(Expenditure_Quarterly)
WithinSampleLength=N-8
WithinSample=ts(Expenditure_Quarterly[1:WithinSampleLength],frequency=4,start=c(2006,4))
print(WithinSample)
```

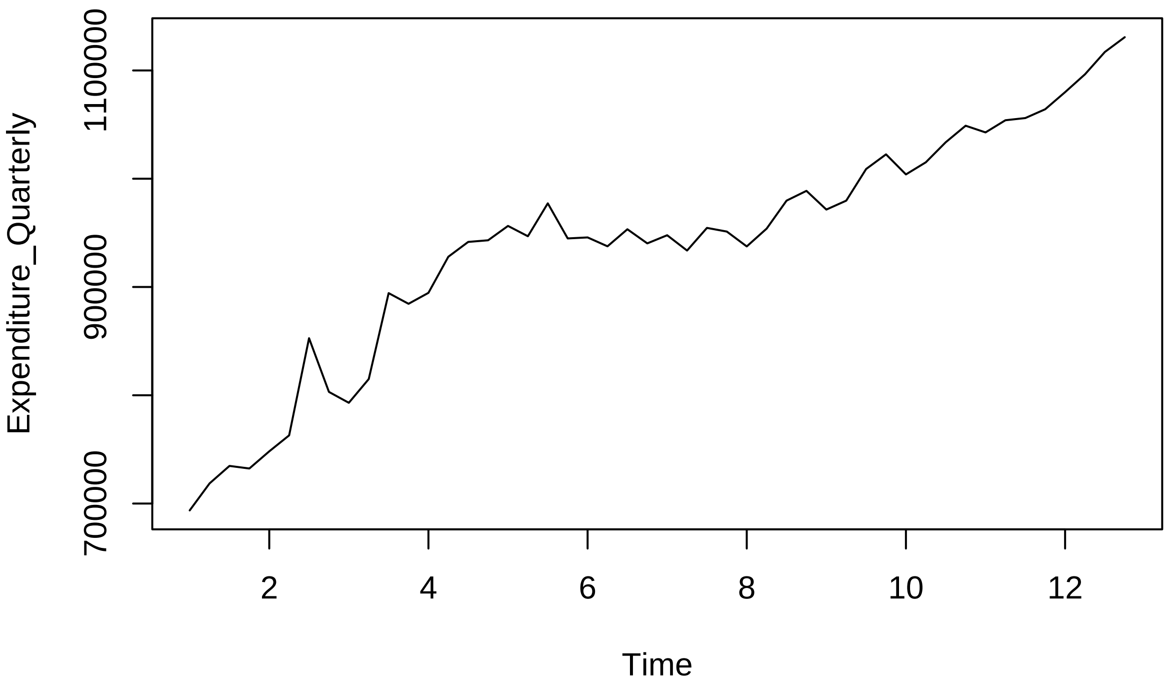
```
##         Qtr1    Qtr2    Qtr3    Qtr4
## 2006                          693643
## 2007  718612  734747  732366  748213
## 2008  762963  852722  803094  793027
## 2009  814978  894346  884482  894576
## 2010  927941  941628  943152  956392
```

```
## 2011  946858  977285  944835  945771
## 2012  937575  953318  940273  947808
## 2013  933661  954582  951184  937463
## 2014  953938  979777  988781  971506
## 2015  979738 1008980 1022502 1003962
## 2016 1015165 1033669 1048953
```

```r
PostTest=Expenditure_Quarterly[(WithinSampleLength+1):N]
print(PostTest)
```

```
## [1] 1042766 1054016 1056024 1064175 1079954 1096440 1117081 1130728
```

Plot



Purpose

When modeling a series, you must start by testing for stationarity, this is done by looking at the behavior of the error term. We are looking to see if the errors or series are autocorrelated, and thus the behavior of the variable today I affected by its behavior in the past. If the autocorrelation coefficient, rho, is equal to 1, the error term behaves as Et=Et-1+ut. This is a random walk model, thus the variance become undefined, as it goes to infinity. In terms of the difference equation, the time path and the general solution of this variable will not converge to equilibrium, the model is totally random. This is a unit root, which can detected by a Dickey-Fuller test, and it must be dealt with by differencing the equation. Once the series is stationary, it must be tested to see if the series is truly white noise, which can be tested by a Ljung-Box test. In the case that it is truly white noise, the model can not be used to forecast. If it is not, examining of the autocorrelation functions and partial autocorrelation functions will reveal if the model should be an AR,MA, or ARMA process.

#Test for Stationarity First, a Dickey-Fuller test for a unit root was ran on the original series.

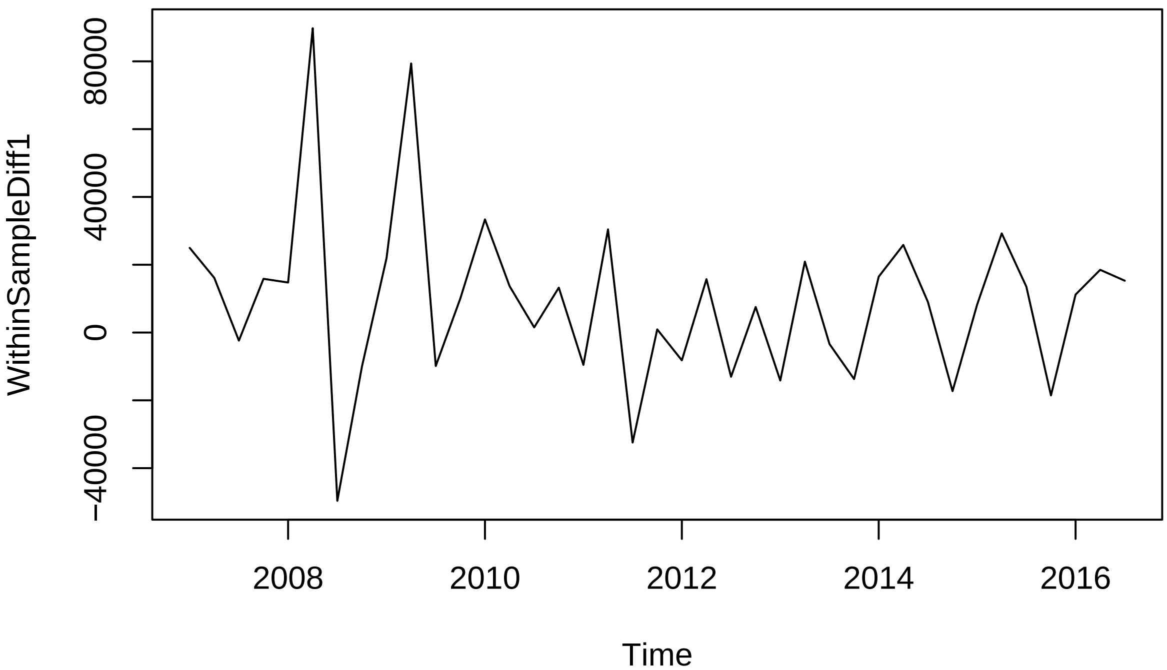
```r
#adf is augmented dickey fuller test
adf.test(WithinSample,alternative='stationary')
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  WithinSample
## Dickey-Fuller = -2.1713, Lag order = 3, p-value = 0.5063
## alternative hypothesis: stationary
```

The results of this test: A large p-value is indicative of a unit root process, that the series is not stationary. To deal with a not stationary series, we use differencing. Reject null hypothesis that series is stationary, must difference it.

Differencing the equation:

```
WithinSampleDiff1=diff(WithinSample, differences=1)
plot(WithinSampleDiff1)
```



Dickey-Fuller Test for Stationarity on First-Differenced Series:

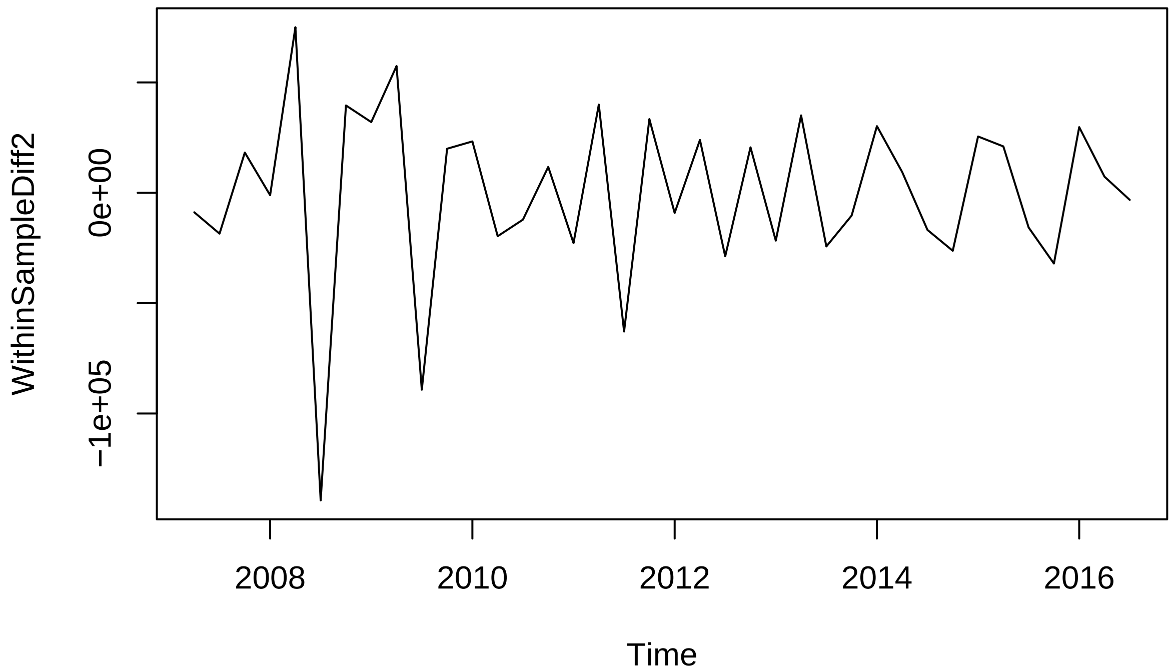
```
adf.test(WithinSampleDiff1,alternative='stationary')
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  WithinSampleDiff1
## Dickey-Fuller = -2.5403, Lag order = 3, p-value = 0.3618
## alternative hypothesis: stationary
```

The results of this test: Still reject null, difference again.

Differencing the equation again

```
WithinSampleDiff2=diff(WithinSample, differences=2)
plot(WithinSampleDiff2)
```



Dickey-Fuller Test for Stationarity on Second-Differenced Series

```
adf.test(WithinSampleDiff2,alternative='stationary')
```

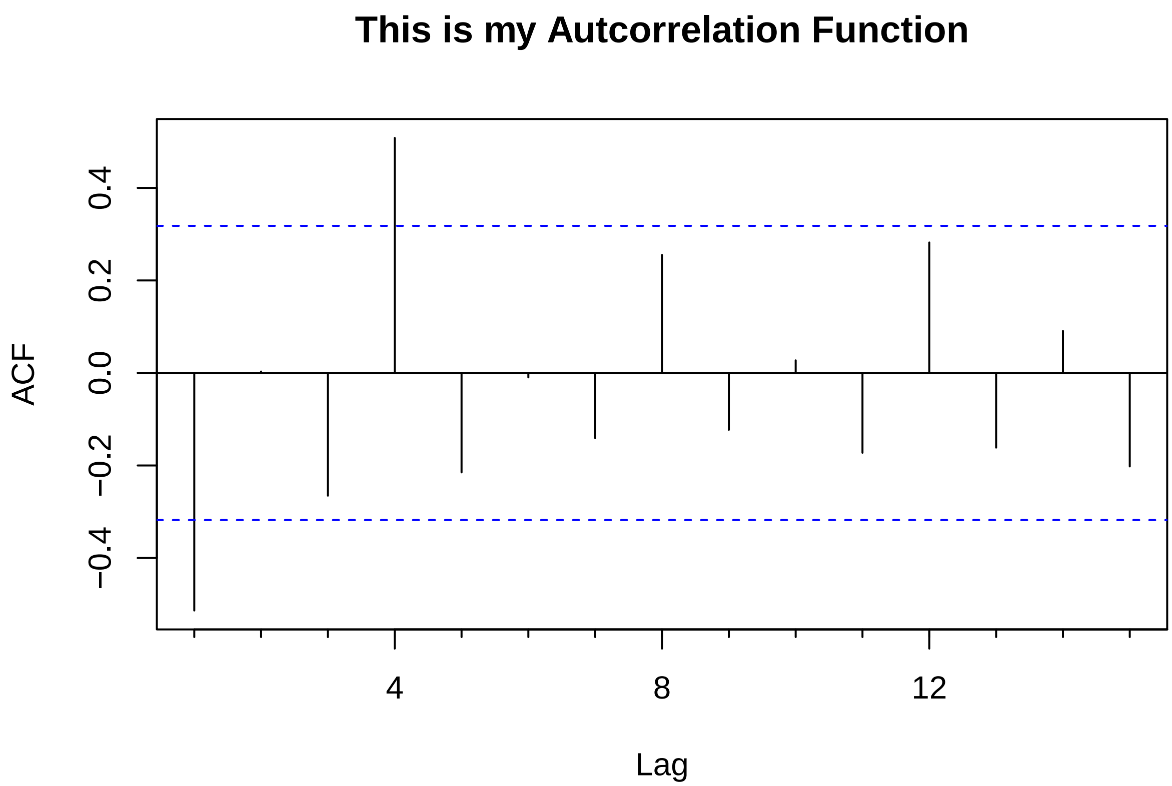
```
## Warning in adf.test(WithinSampleDiff2, alternative = "stationary"): p-value
## smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  WithinSampleDiff2
## Dickey-Fuller = -6.5834, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

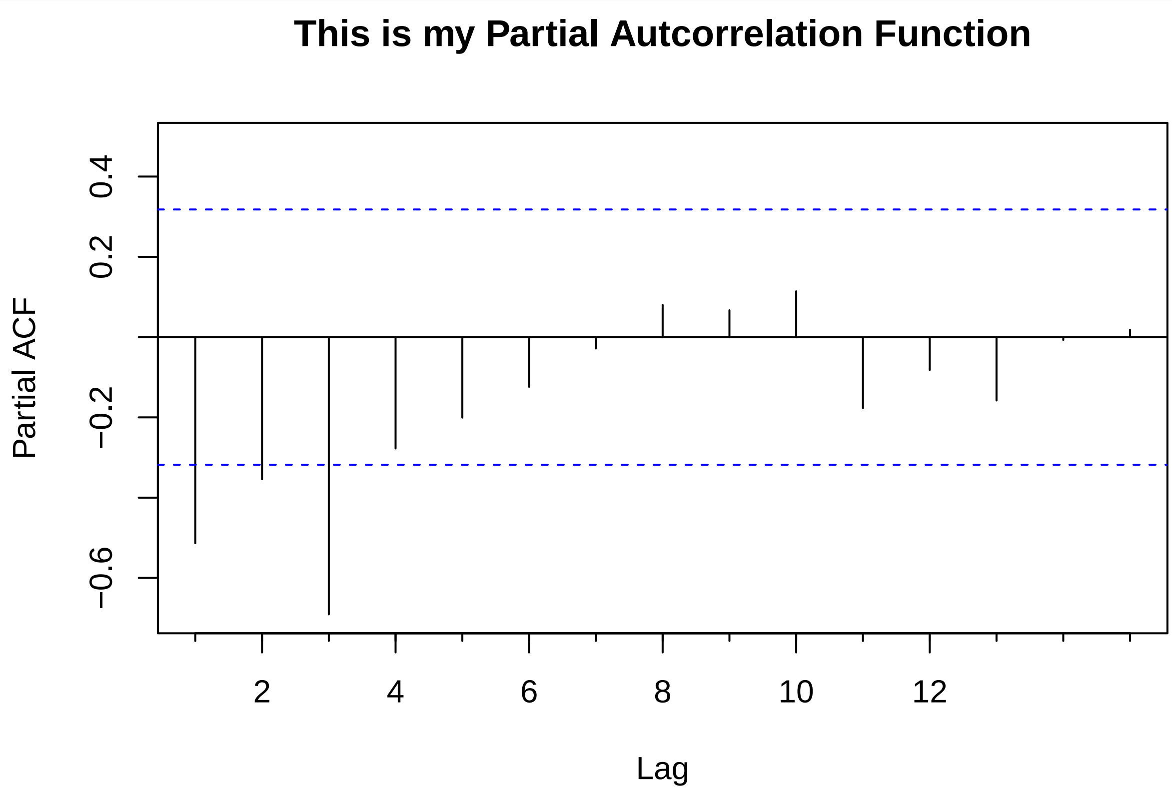
The results of this test: Now fail to reject null, there is now not a unit root process and the series is stationary.

#Partial Autocorrelation Function (PACF) and Autocorrelation Function(ACF)

```
Acf(WithinSampleDiff2,main='This is my Autcorrelation Function')
```




```
Pacf(WithinSampleDiff2,main='This is my Partial Autcorrelation Function')
```



From visual inspection, it appears to tail off after 3 My Guess: IMA (2,4)

#Doing Auto-ARIMA Auto-ARIMA is selected by running tons of possible combinations of a model and choosing the one with the smallest AIC (Akaike Information Criteria)

```r
auto.arima(WithinSample,seasonal=TRUE)
```

```
## Series: WithinSample
## ARIMA(0,1,1)(0,0,1)[4] with drift
##
## Coefficients:
##           ma1    sma1     drift
##       -0.3688  0.6272  9247.707
## s.e.   0.1535  0.1528  3150.005
##
## sigma^2 estimated as 415737851:  log likelihood=-441.83
## AIC=891.65   AICc=892.83   BIC=898.31
```

This Auto Arima I disagree with, it doesn't make sense it only differenced it once when it is still not stationary yet. In response I forced it to allow me to difference twice.

```r
ARIMAWithin=auto.arima(WithinSample,d=2,seasonal=TRUE)
print(ARIMAWithin)
```

```
## Series: WithinSample
## ARIMA(0,2,2)(0,0,1)[4]
##
## Coefficients:
##           ma1     ma2    sma1
##       -1.3338  0.3973  0.6281
## s.e.   0.1684  0.1884  0.1542
##
## sigma^2 estimated as 441882995:  log likelihood=-432.62
## AIC=873.23   AICc=874.44   BIC=879.78
```

#Forecasting with Auto-Arima

```r
#h means steps ahead
ARIMAWithinYhats=forecast(ARIMAWithin,h=8)
print(ARIMAWithinYhats)
```

```
##         Point Forecast   Lo 80   Hi 80     Lo 95   Hi 95
## 2016 Q4        1040532 1013592 1067472  999330.7 1081733
## 2017 Q1        1055804 1023432 1088175 1006296.0 1105312
## 2017 Q2        1063111 1025237 1100985 1005187.8 1121035
## 2017 Q3        1075116 1031627 1118604 1008605.8 1141626
## 2017 Q4        1080337 1021246 1139429  989965.2 1170710
## 2018 Q1        1087824 1018593 1157055  981945.0 1193703
## 2018 Q2        1095311 1015920 1174702  973893.2 1216729
## 2018 Q3        1102798 1013146 1192449  965688.0 1239907
```

```r
ActualForecastValues=ARIMAWithinYhats[4]
print(PostTest[1:8])
```

```
## [1] 1042766 1054016 1056024 1064175 1079954 1096440 1117081 1130728
```

```r
PostValues=PostTest[1:8]
print(PostValues)
```

```
## [1] 1042766 1054016 1056024 1064175 1079954 1096440 1117081 1130728
```

Getting post-sample residuals:

```
ARIMAResidualsTest=ARIMAWithinYhats$mean-PostTest
print(ARIMAResidualsTest)
```

```
##             Qtr1        Qtr2        Qtr3        Qtr4
## 2016                                       -2233.8898
## 2017   1787.8849   7087.3125  10940.8186    383.4696
## 2018  -8615.7965 -21770.0625 -27930.3286
```

Getting RMSE

```
sqrt(mean(ARIMAResidualsTest^2))
```

```
## [1] 13722.86
```