Week 5: ARIMA (Part 1)

Xuan Pham

9/17/2019

You will need these packages: fpp2, urca, forecast, and stats

# Refresher: Transformation

We have talked about the **i.i.d** assumption:  
\* observations are independent of each other  
\* observations are drawn from the same (identical) probability distribution

If this assumption is met, we can calculate the joint probability of all the observations as follows: P(time series) = P(X1) \* P(X2) \* P(X3)…  
In machine learning parlance, the i.i.d assumption allows us to focus on modeling the relationships among variables (columns) and not have to worry about the relationships among the observations (rows).

The **i.i.d** assumption is questionable for time series data. We cannot guarantee that each observation is arising from the same processes. It’s possible that we are seeing many “mini” processes occurring over some length of time. We called this “autocorrelation.” Hence, we need to figure out ways to remove autocorrelation.

We have also looked at time series exploratory data analysis (EDA) where we focused on the four moments of a distribution:  
1. Mean  
2. Variance  
3. Skewness  
4. Kurtosis

We also know that a normal distribution has these characteristics:

1. Mean is stable throughout the series.
2. Variance is stable throghout the series (variance = sigma^square).
3. Skewness does not exist. The distribution should be symmetric.
4. Peakiness does not exist.

We have already discussed ways to stabilize the variance of a time series via transformations. We discussed logarithmic, power, and Box-Cox transformations. Here is an example:

library(fpp2)

## Loading required package: ggplot2

## Loading required package: forecast

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

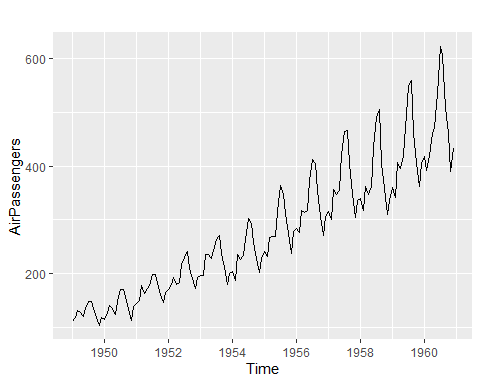
## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

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

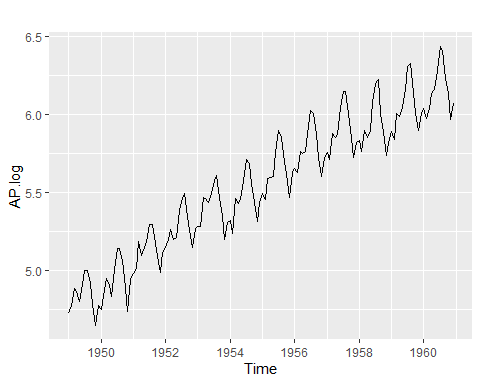
## Loading required package: fma

## Loading required package: expsmooth

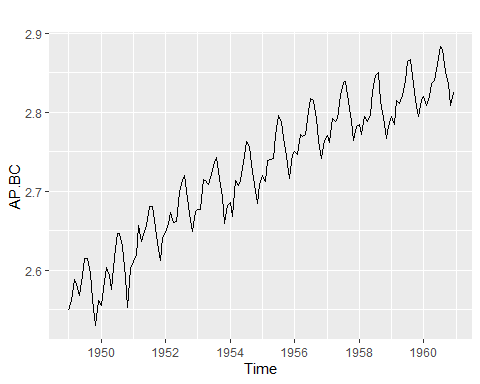
autoplot(AirPassengers)



AP.log <- log(AirPassengers)  
autoplot(AP.log)



AP.BC <- BoxCox(AirPassengers, lambda="auto")  
autoplot(AP.BC)



# Stationarity

We are going to tackle the issue of stabilizing the mean in this module. If a time series has trend and/or seasonality, the mean is not constant/stable. If we can remove the trend and seasonality, we are left with a series that has no predictable patterns. A time series with no discernable, predictable patterns is called a stationary series.

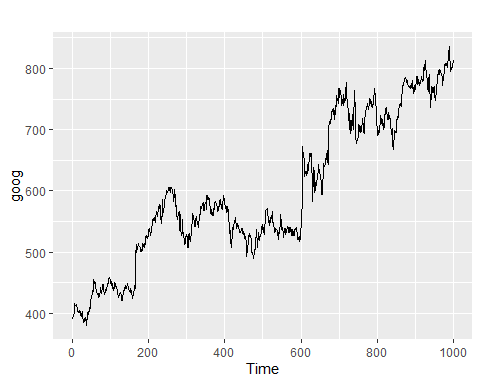
Let’s look at a new time series. The series has a trend.

To stabilize the mean (and remove the trend), we are going to take the difference between consecutive observations.

Y\_t = Y\_t - Y\_(t-1)

This is called taking the “first difference” or “first order difference.”

autoplot(goog)



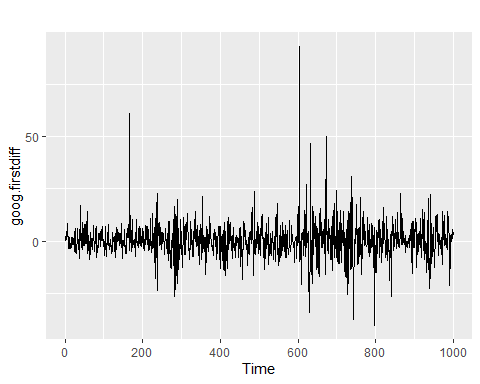
goog.lag <- lag(goog, k=1)  
  
head(goog)

## Time Series:  
## Start = 1   
## End = 6   
## Frequency = 1   
## [1] 392.8300 392.5121 397.3059 398.0113 400.4902 408.0957

head(goog.lag)

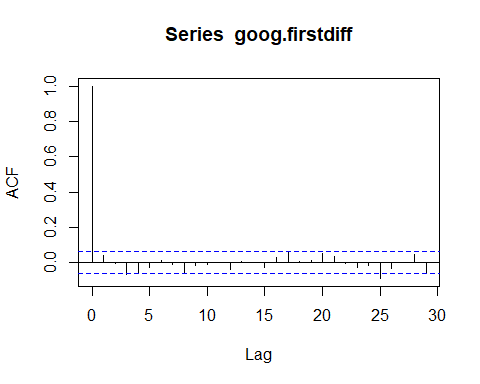
## Time Series:  
## Start = 0   
## End = 5   
## Frequency = 1   
## [1] 392.8300 392.5121 397.3059 398.0113 400.4902 408.0957

goog.firstdiff <- diff(goog)  
autoplot(goog.firstdiff)



Notice the trend is gone. Let’s do some further digging.

acf(goog.firstdiff)



# COMPUTE LJUNG-BOX TEST FOR WHITE NOISE (NO AUTOCORRELATION)  
#H0: p(1) = p(2) = p(k) = 0  
#H1: p(k) is not equal to 0  
  
Box.test(goog.firstdiff,lag=10,type='Ljung')

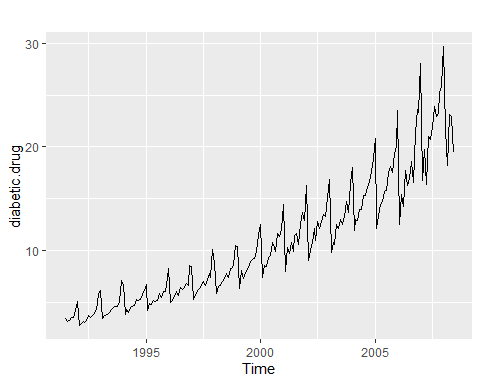
##   
## Box-Ljung test  
##   
## data: goog.firstdiff  
## X-squared = 13.123, df = 10, p-value = 0.2169

We are left with a series without discernable pattern. This is a stationary time series.

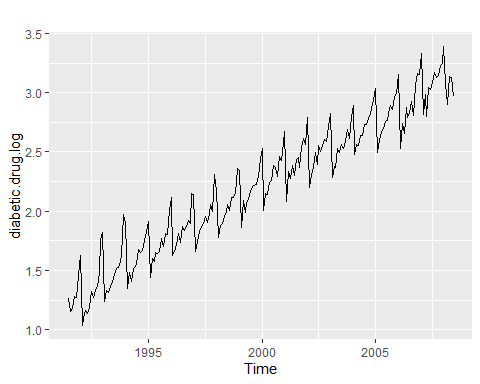
It is also possible to take a second-order difference if we do not achieve stationarity with the first-order difference. Typically, we do not go beyond second-order difference.

Sometimes, it is necessary to take the seasonal difference to achieve stationarity. Seasonal difference is the change between one year to the next. Remember that seasonal difference is NOT ordinary difference (i.e. first-order, second-order)

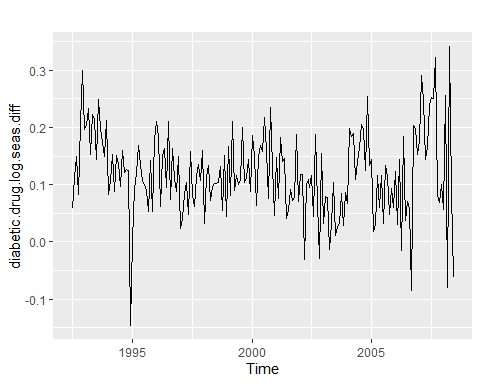
diabetic.drug <- a10  
autoplot(diabetic.drug)



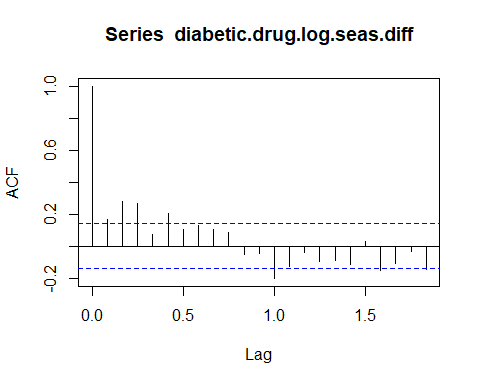
diabetic.drug.log <- log(diabetic.drug)  
autoplot(diabetic.drug.log)



diabetic.drug.log.seas.diff <- diff(diabetic.drug.log, 12)  
autoplot(diabetic.drug.log.seas.diff)



acf(diabetic.drug.log.seas.diff)

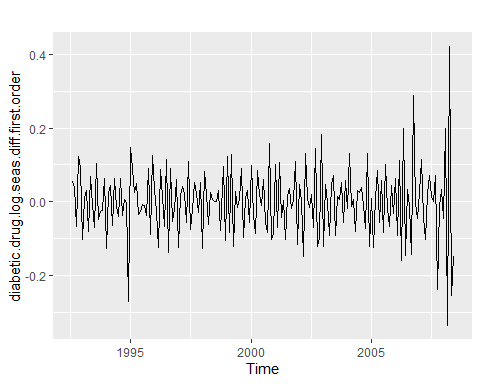


Box.test(diabetic.drug.log.seas.diff,lag=12,type='Ljung')

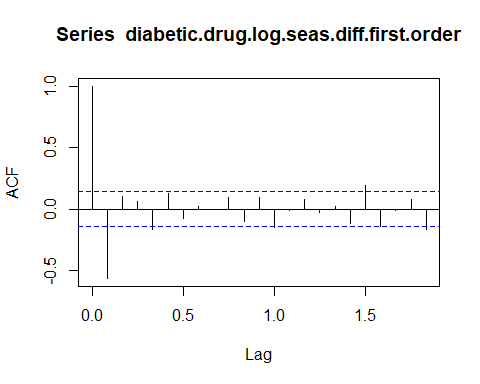
##   
## Box-Ljung test  
##   
## data: diabetic.drug.log.seas.diff  
## X-squared = 63.431, df = 12, p-value = 5.31e-09

Still not quite stationary yet. Perhaps we need to take the first difference to remove the trend?

diabetic.drug.log.seas.diff.first.order <- diff(diabetic.drug.log.seas.diff)  
autoplot(diabetic.drug.log.seas.diff.first.order)



acf(diabetic.drug.log.seas.diff.first.order)



Box.test(diabetic.drug.log.seas.diff.first.order,lag=12,type='Ljung')

##   
## Box-Ljung test  
##   
## data: diabetic.drug.log.seas.diff.first.order  
## X-squared = 83.932, df = 12, p-value = 7.301e-13

What do you think about the different information in the ACF plot and the JB test?

## Unit Root Test

The unit root test allows us to determine whether a series is stationary or not. The textbook used the KPSS test. We will follow the same fashion.

#H0: Data is stationary.   
#H1: Data is not stationary.   
library (urca) #KPSS test is in the urca package   
summary(ur.kpss(diabetic.drug))

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 4 lags.   
##   
## Value of test-statistic is: 3.827   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

Since the test statistic is greater than all the critical values, we reject H0. The diabetic drug time series is not stationary. But what about after we have done the seasonal difference and then the first order difference? Is it stationary now?

#H0: Data is stationary.   
#H1: Data is not stationary.   
library (urca) #KPSS test is in the urca package   
summary(ur.kpss(diabetic.drug.log.seas.diff.first.order))

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 4 lags.   
##   
## Value of test-statistic is: 0.0576   
##   
## 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 smaller than all the critical values. We cannot reject H0. We do have a stationary series now.

There is a nifty function that used the KPSS test to determine the number of ordinary difference(s) to do to make a series stationary. There’s also a similar function for seasonal difference.

library(forecast)  
ndiffs(goog, alpha = 0.05)

## [1] 1

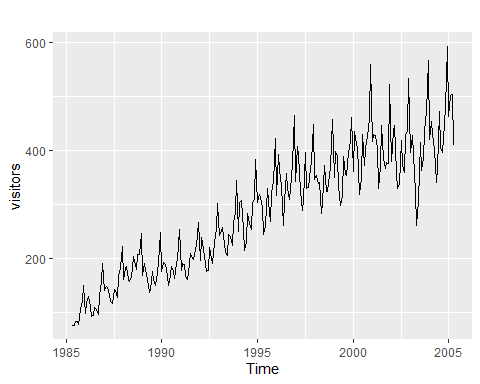
nsdiffs(diabetic.drug, alpha = 0.05)

## [1] 1

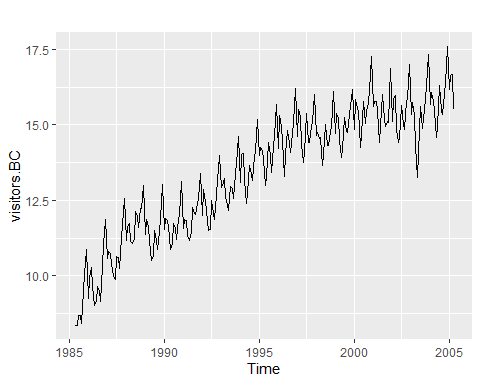
ndiffs(diabetic.drug.log.seas.diff, alpha=0.05) #notice the test gives a different answer than what we did above.

## [1] 0

autoplot(visitors)



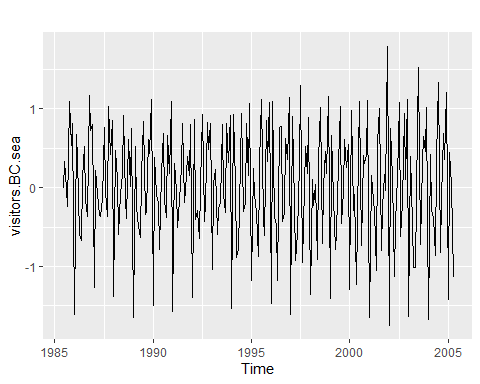
visitors.BC <- BoxCox(visitors, lambda="auto")  
autoplot(visitors.BC)



nsdiffs(visitors.BC)

## [1] 1

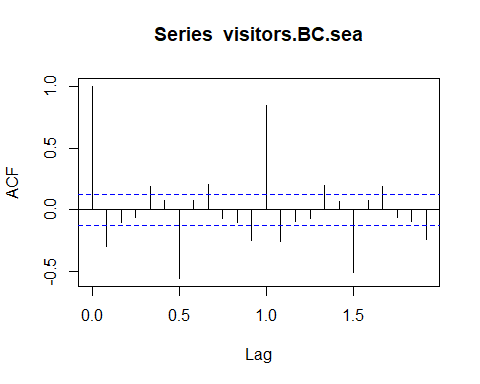
visitors.BC.sea <- diff(visitors.BC, k=12)  
autoplot(visitors.BC.sea)



ndiffs(visitors.BC.sea)

## [1] 0

acf(visitors.BC.sea)



Box.test(visitors.BC.sea, type = "Ljung-Box")

##   
## Box-Ljung test  
##   
## data: visitors.BC.sea  
## X-squared = 21.922, df = 1, p-value = 2.839e-06

summary(ur.kpss(visitors.BC.sea))

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 4 lags.   
##   
## Value of test-statistic is: 0.0519   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

So what do we do once we have a stationary time series? How do we model the underlying processes in a stationary time series?

# Autoregressive Models

Since observations in a time series are correlated, can we use autocorrelation to our advantage? Can we use past observations to forecast future observations? In an autoregressive model, we use a multiple linear regression model where the lagged values of Y(t) are the predictors.

library(stats)  
  
diabetic.lag1 <- lag(diabetic.drug.log.seas.diff.first.order, k=1)  
head(diabetic.drug, 12)

## Jan Feb Mar Apr May Jun Jul  
## 1991 3.526591  
## 1992 5.088335 2.814520 2.985811 3.204780 3.127578 3.270523   
## Aug Sep Oct Nov Dec  
## 1991 3.180891 3.252221 3.611003 3.565869 4.306371  
## 1992

head(diabetic.lag1,12)

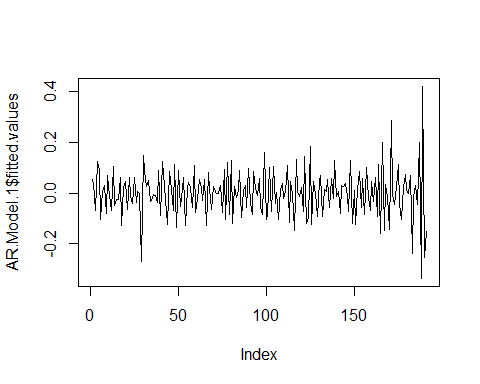
## Jan Feb Mar Apr May  
## 1992   
## 1993 0.007512468 0.029984441 -0.080897596 0.069183315 -0.008411313  
## Jun Jul Aug Sep Oct  
## 1992 0.054076227 0.037389960 -0.066394364 0.123879968  
## 1993 -0.069972142   
## Nov Dec  
## 1992 0.092448625 -0.103261131  
## 1993

AR.Model.1 <- lm(diabetic.drug.log.seas.diff.first.order~diabetic.lag1)  
summary(AR.Model.1)

## Warning in summary.lm(AR.Model.1): essentially perfect fit: summary may be  
## unreliable

##   
## Call:  
## lm(formula = diabetic.drug.log.seas.diff.first.order ~ diabetic.lag1)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -5.739e-17 -5.750e-18 -3.650e-18 -1.380e-18 7.154e-16   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 2.510e-19 3.812e-18 6.600e-02 0.948   
## diabetic.lag1 1.000e+00 4.055e-17 2.466e+16 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 5.268e-17 on 189 degrees of freedom  
## Multiple R-squared: 1, Adjusted R-squared: 1   
## F-statistic: 6.081e+32 on 1 and 189 DF, p-value: < 2.2e-16

plot(AR.Model.1$fitted.values, type="l")



We can make the AR model above to have two lagged Y(t) as predictors too. This can continue to include as many lagged predictors as we have. Hence, autoregressive models are denoted as **AR(p)** to show the number of lagged periods used as predictors.

# Moving Average Models

Instead of using lagged values as predictors, we can also use the past (lagged) forecast errors as predictors too.

forecast.errors <- AR.Model.1$residuals  
  
MA.Model.1 <- lm(diabetic.drug.log.seas.diff.first.order~forecast.errors)  
summary(MA.Model.1)

##   
## Call:  
## lm(formula = diabetic.drug.log.seas.diff.first.order ~ forecast.errors)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.33568 -0.06322 0.00571 0.05409 0.42218   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -6.308e-04 6.838e-03 -0.092 0.927  
## forecast.errors -8.622e-01 1.305e+14 0.000 1.000  
##   
## Residual standard error: 0.0945 on 189 degrees of freedom  
## Multiple R-squared: 2.394e-31, Adjusted R-squared: -0.005291   
## F-statistic: 4.524e-29 on 1 and 189 DF, p-value: 1

We can also include more than one lagged forecast error. Moving Average models are denoted as **MA(q)**.

# ARIMA (AutoRegressive Integrated Moving Average) Models

ARIMA is the combination of AR, reverse of Differencing (Integrated), and MA models. An ARIMA model is denoted as **ARIMA(p,d,q)**.  
p = number of lagged observations used in the autoregressive model  
d = number of differences performed to make a time series stationary  
q = number of lagged forecast errors used in a moving average model

ARIMA is considered a black-box model because it is not always easy to tell what p, d, and q values to use.

Let’s start with d. We know that we can use the tools above to find the d value.

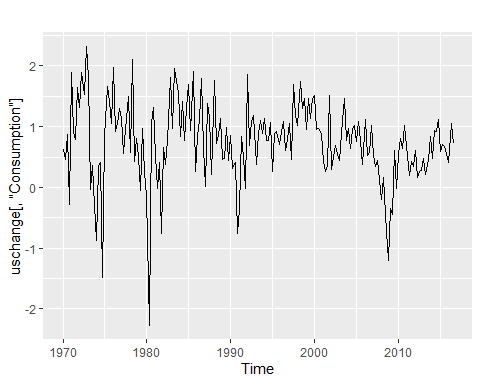
## ARIMA(p,d,0)

If we have a time series that is ARIMA(p,d,0) then we can use the ACF plot and an associated plot called the PACF (partial autocorrelation function) to help us find the p value.

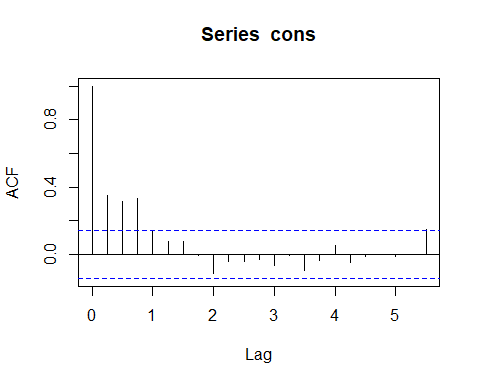
The PACF measures the relationship between Y(t) and Y(t-k) after removing the effects of all the previous lags.

IF the ACF is exponentially decreasing or sinusoidal (oscillating pattern) AND PACF shows a significant spike up to lag k but not after that  
THEN use the number of lags in the PACF to determine the p value.

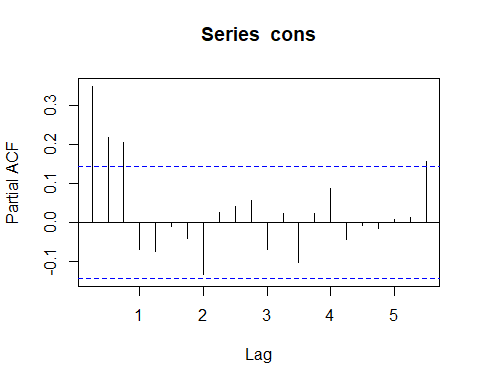
autoplot(uschange[,"Consumption"])



cons <- uschange[,"Consumption"]  
acf(cons)



pacf(cons) #3 significant lags



Here’s the ARIMA(3,0,0) model:

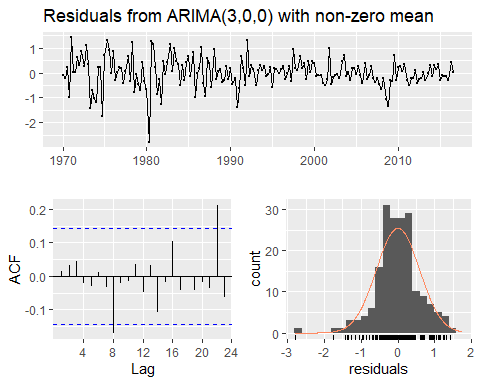
fit <- Arima(cons, order=c(3,0,0))  
fit

## Series: cons   
## ARIMA(3,0,0) with non-zero mean   
##   
## Coefficients:  
## ar1 ar2 ar3 mean  
## 0.2274 0.1604 0.2027 0.7449  
## s.e. 0.0713 0.0723 0.0712 0.1029  
##   
## sigma^2 estimated as 0.3494: log likelihood=-165.17  
## AIC=340.34 AICc=340.67 BIC=356.5

This model can be written as: Cons(t) = 0.227Cons(t-1) + 0.160Cons(t-2) + 0.745Cons(t-3)

Now we check the residuals to make sure they are white noise. The residuals are indeed white noise!

checkresiduals(fit)



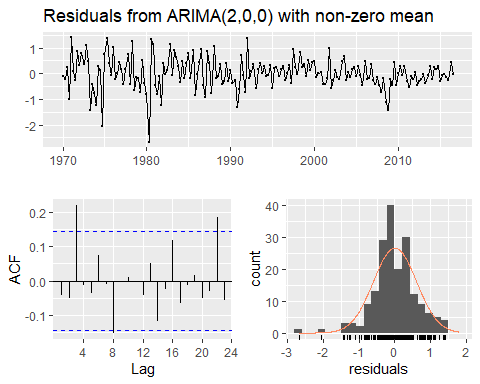
##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(3,0,0) with non-zero mean  
## Q\* = 6.7407, df = 4, p-value = 0.1502  
##   
## Model df: 4. Total lags used: 8

What if I were to pick ARIMA(2,0,0) or ARIMA(3,0,0)?

fit2 <- Arima(cons, order=c(2,0,0))  
fit2

## Series: cons   
## ARIMA(2,0,0) with non-zero mean   
##   
## Coefficients:  
## ar1 ar2 mean  
## 0.272 0.2166 0.7458  
## s.e. 0.071 0.0710 0.0848  
##   
## sigma^2 estimated as 0.3628: log likelihood=-169.13  
## AIC=346.27 AICc=346.49 BIC=359.19

checkresiduals(fit2)

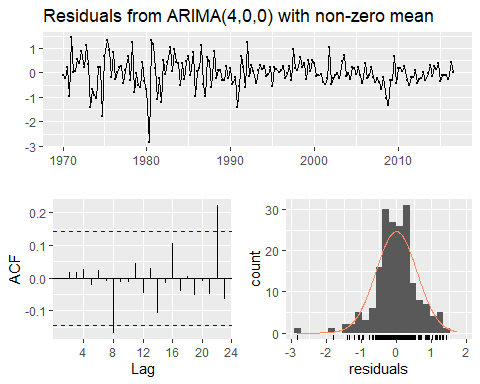


##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(2,0,0) with non-zero mean  
## Q\* = 15.983, df = 5, p-value = 0.006892  
##   
## Model df: 3. Total lags used: 8

fit3 <- Arima(cons, order=c(4,0,0))  
fit3

## Series: cons   
## ARIMA(4,0,0) with non-zero mean   
##   
## Coefficients:  
## ar1 ar2 ar3 ar4 mean  
## 0.2416 0.1714 0.2188 -0.0685 0.7461  
## s.e. 0.0727 0.0731 0.0730 0.0730 0.0965  
##   
## sigma^2 estimated as 0.3496: log likelihood=-164.73  
## AIC=341.46 AICc=341.93 BIC=360.85

checkresiduals(fit3)



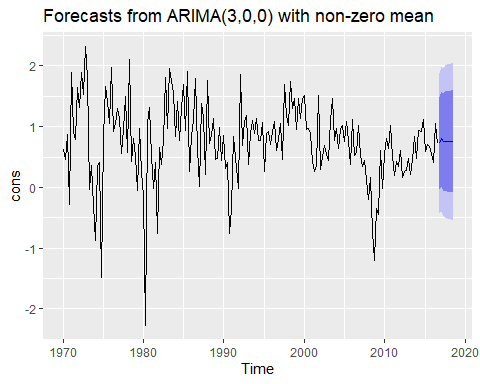
##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(4,0,0) with non-zero mean  
## Q\* = 6.0206, df = 3, p-value = 0.1106  
##   
## Model df: 5. Total lags used: 8

We would use the AICc to select the best model among the three candidates. ARIMA(2,0,0): 346.49 ARIMA(3,0,0): 340.67 ARIMA(4,0,0): 341.93

The ARIMA(3,0,0) model has the lowest AICc so it is the best fit.

How about using the ARIMA(3,0,0) model to forecast?

#run this to see the actual point & interval forecasts  
#forecast(fit,h=12)  
#or you can plot them  
autoplot(forecast(fit), h=12)



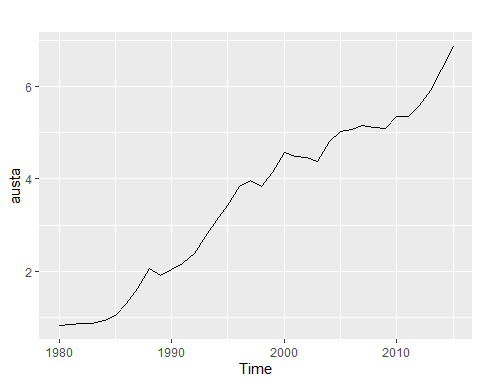
## ARIMA(0,d,q)

If we have a time series that is ARIMA(0,d,q) then we can use the ACF plot and an associated plot called the PACF (partial autocorrelation function) to help us find the p value.

The PACF measures the relationship between Y(t) and Y(t-k) after removing the effects of all the previous lags.

IF the PACF is exponentially decreasing or sinusoidal (periodic oscillation) AND the ACF shows a significant spike up to lag k but not after that  
THEN use the number of lags in the ACF to help you determine the q value.

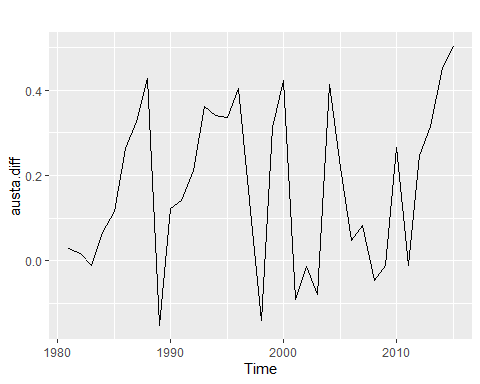
autoplot(austa)



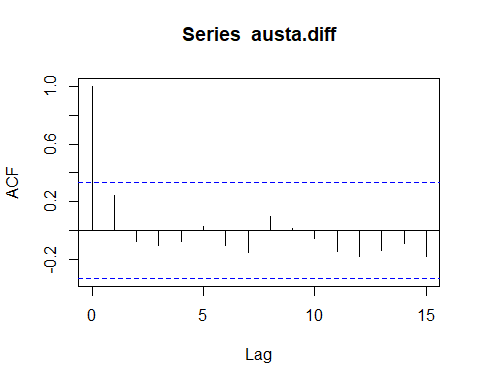
ndiffs(austa, alpha = 0.05)

## [1] 1

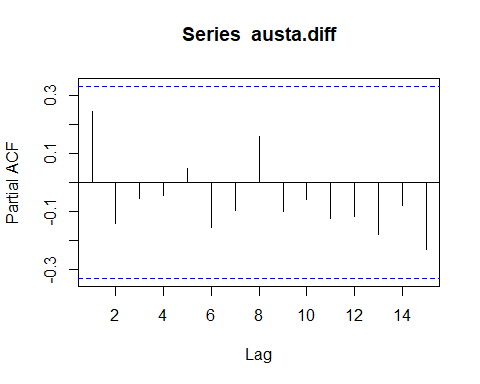
austa.diff <- diff(austa)  
autoplot(austa.diff)



acf(austa.diff) #one significant lag



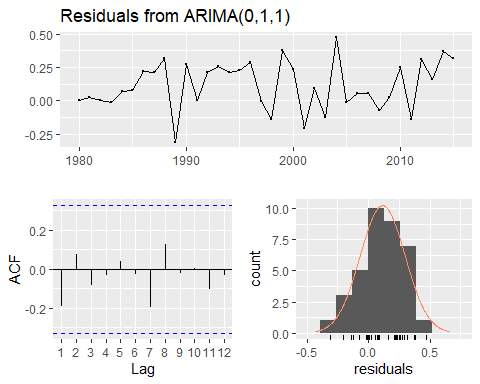
pacf(austa.diff) #no significant lag



fit <- Arima(austa, order=c(0,1,1))  
fit

## Series: austa   
## ARIMA(0,1,1)   
##   
## Coefficients:  
## ma1  
## 0.4936  
## s.e. 0.1265  
##   
## sigma^2 estimated as 0.04833: log likelihood=3.73  
## AIC=-3.45 AICc=-3.08 BIC=-0.34

checkresiduals(fit)

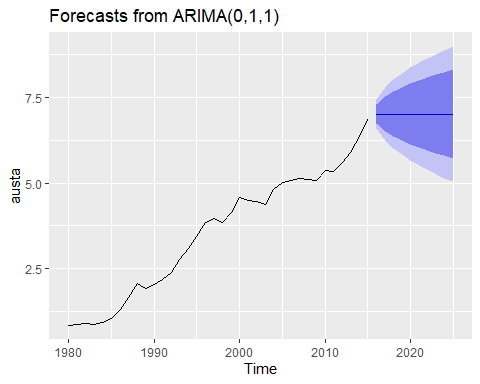


##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(0,1,1)  
## Q\* = 3.8403, df = 6, p-value = 0.6983  
##   
## Model df: 1. Total lags used: 7

forecast(fit, h=2)

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 2016 7.015837 6.734107 7.297567 6.584969 7.446705  
## 2017 7.015837 6.509440 7.522234 6.241369 7.790305

autoplot(forecast(fit),h=2) #two year forecast



If you have both AR and MA processes in a time series, then the ACF and PACF will not be able to help you. You will need to find right out a combination of both to get a model. More on this next week.

# Decision Tree for ARIMA Model Selection

From Hyndman’s text in Section 8.7 