Time Series Analysis

Joan Dai

11/24/2017

```r
library(TSA)
library(astsa)
```

Introduction

When I was studying abroad in Japan last year, I had an opportunity to take a basic time series class. It was one of the more rewarding and technical classes the university had offered, and I strongly enjoyed learning the materials. Unfortunately, I did not have too much exposure working with datasets while I was there. I hope to use this assignment to both better acquaint myself with time series in a R environment, and provide a nice introduction to Time Series for readers like you. Enjoy!

Time series is incredibly relavent in the modern day. It's used in econometrics, statistics, and financial mathematics just to name a few major fields. It can even be used in astronomy to predict the number of sunspots on the sun in the future! How awesome and random is that? Maybe one day, someone will be able to predict when we will all meet our special someone. Wishful thinking? Perhaps. Anyways, now that I've got your attention, let's get started right away.

Time Series Basics

The goal of time series is to

1. find the underlying pattern and features of time data
2. reconcile past observations with both past and present values
3. forecast

If you have done regression before, it is extremely similar! The assumptions and requirements are of course going to be different, but the overall gist is the same. Instead of regressing on variables, we regress the variable of interest on its own past values.

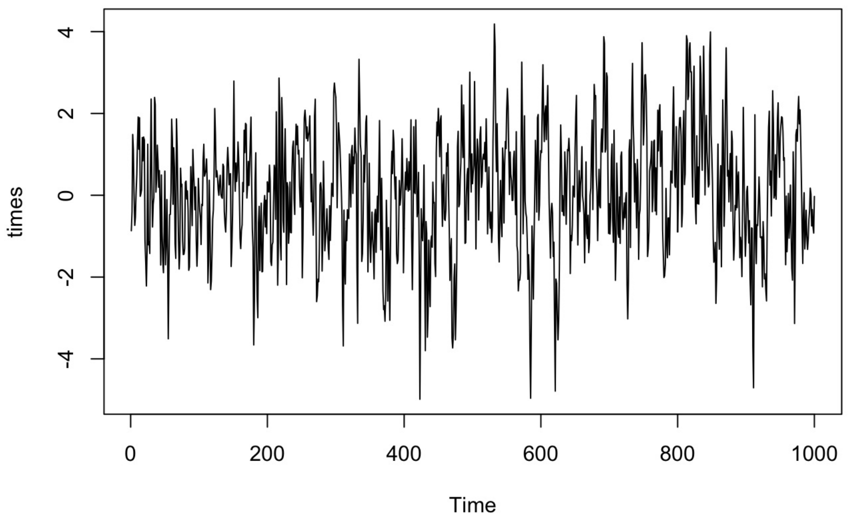
Let's visualize some data using time series package `TSA`, `astsa` and some data I will be generating called `times`.

We can use the function `plot()` normally, and it will give us a time series plot because the data itself is a time object. Can you see a pattern?

```r
set.seed(100)
times <- arima.sim(model = list(order = c(3,0,0), ar = c(.8897, -.4858, .32)), n = 1000)
class(times)
```

```
## [1] "ts"
```

```r
plot(times)
```



Now that I have shown you what the plot looks like (I hope it was not too anticlimactic), I would like to get into the nitty gritty of what the components are, and the steps one must take in order to analyze it. In order to do that, I must introduce some models.

The Autoregressive Model - AR(p)

A simple formula for an **autoregressive (AR)** time series model is

$x_t = \delta + \phi_1 x_{t-1} + w_t$

This is called an AR(1) model because we regress x on the observed value in the previous period. If we add one more explanatory variable, it would be called AR(2). Here the errors, w_t are iid N(0, σ_{2w}).

A p^{th} order AR model is

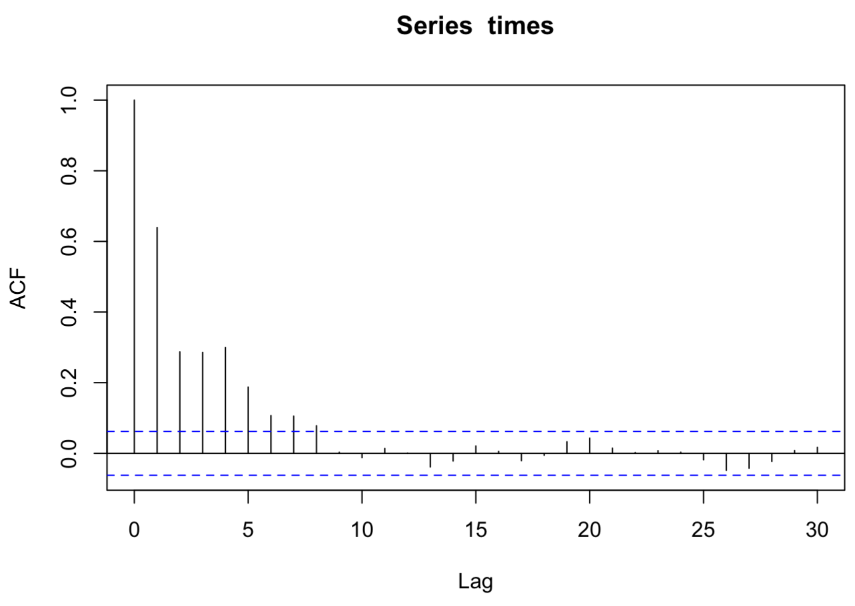
$x_t = \delta + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \dots + \phi_p x_{t-p} + w_t$

We *must* assume the errors follow a normal distribution with mean 0 and constant standard deviation. The errors must also not be correlated with the regressor, x_{t-1} nor x_t.

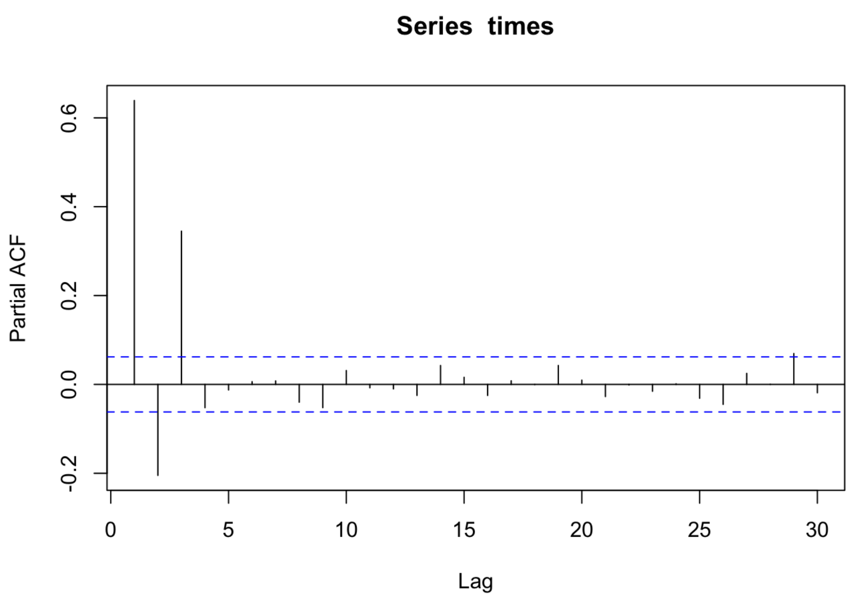
Partial autocorrelation functions (PACF) are used to find the order of AR models because it will cut off at the appropriate order. The plot of the autocorrelation function (ACF) function will taper off and decrease to zero slowly.

In general, we are interested where the verticle lines first fall under the blue significance-level horizontal line for both ACF and PACF functions. That will show us the correct order.

```
acf(times)
```



```
pacf(times) #we see the first 3 orders is significant
```



```
arima(times,c(3,0,0)) #intercept not significant, but the coefficients all are. If you
```

```
##
## Call:
## arima(x = times, order = c(3, 0, 0))
##
## Coefficients:
##          ar1      ar2     ar3  intercept
##       0.8398  -0.4699  0.3444     0.0616
## s.e.  0.0297   0.0370  0.0297     0.1134
##
## sigma^2 estimated as 1.056:  log likelihood = -1446.78,  aic = 2903.56
```

```
                      #look closely, the coefficients are actually really close to our
                      #model we made previously (.83, -.46, .34) vs (.88, -.48, .32).
arima(times,c(2,0,0)) # intercept not significant but other coefficients are
```

```
##
## Call:
## arima(x = times, order = c(2, 0, 0))
##
## Coefficients:
##          ar1      ar2  intercept
##       0.7694  -0.2045     0.0620
## s.e.  0.0309   0.0309     0.0795
##
## sigma^2 estimated as 1.199:  log likelihood = -1509.9,  aic = 3027.8
```

```
arima(times,c(1,0,0)) # intercept not significant
```

```
##
## Call:
## arima(x = times, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.6386     0.0610
## s.e.  0.0243     0.0977
##
## sigma^2 estimated as 1.251:  log likelihood = -1531.29,  aic = 3068.58
```

Usually, we evaluate models based on the following criteria:

*Coefficient of regressors (are they significant? This can be checked easily. Simply see if the given coefficient value is greater than 2SEs.)

*Log Likelihood (higher is better)

*AIC and BIC (lower is better)

*Residuals (are they homoskedastic?)

It is worthwhile to note that while I said intercepts are not significant, it is alright since we did not set an intercept in our model. We could change the argument within our earlier call of `arima(times, c(3,0,0))` to omit the mean (the intercept). I believe this would give us a more accurate model prediction.

```
fit1 <- arima(times,c(3,0,0), include.mean = F)
```

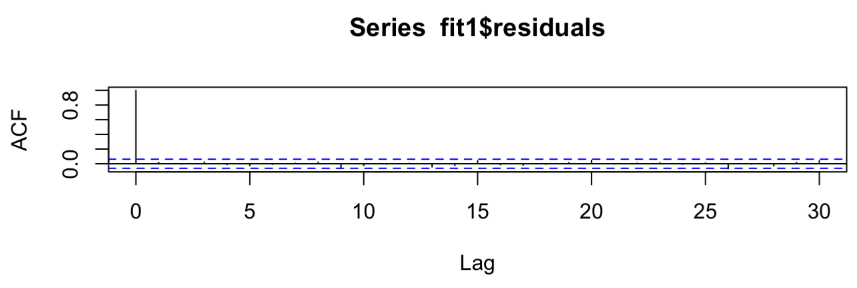
Compared to earlier, our likelihood is bigger, and our AIC is smaller! The coefficients have also become slightly more accurate. (.84, -.47, .34) vs (.88, -.48, .32).

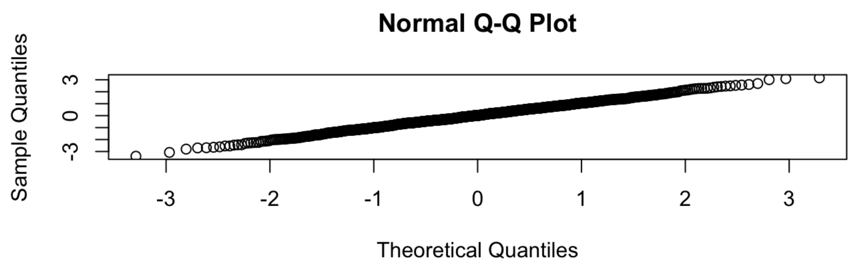
Keep in mind that we also want to make the model as parsimonious as possible; if a simple model can explain the data just as well as a complicated model can, we will often choose a simple model instead.

Since I made the `times` data, I know that it is actually an AR(3) model. We were able to correctly specify this model. In situations where we do not actually know the order of the model, we would have to check some diagnostics. These include *checking the acf of residuals and qqplot of residuals.*

To check if it is indeed a good model, we would have to see the residuals of the fitted model. We can do so by calling the following, since `residuals` is an attribute of an arima model.

```
par(mfrow = c(2,1))
acf(fit1$residuals) #we want all of the lines to be within the blue bands
qqnorm(fit1$residuals) #we want all the points to fall on a straight line
```





We fit this model really nicely!

The Moving Average Model - MA(q)

Another model that is used in time series analysis is **Moving Average (MA)**. The simple MA(1) equation is

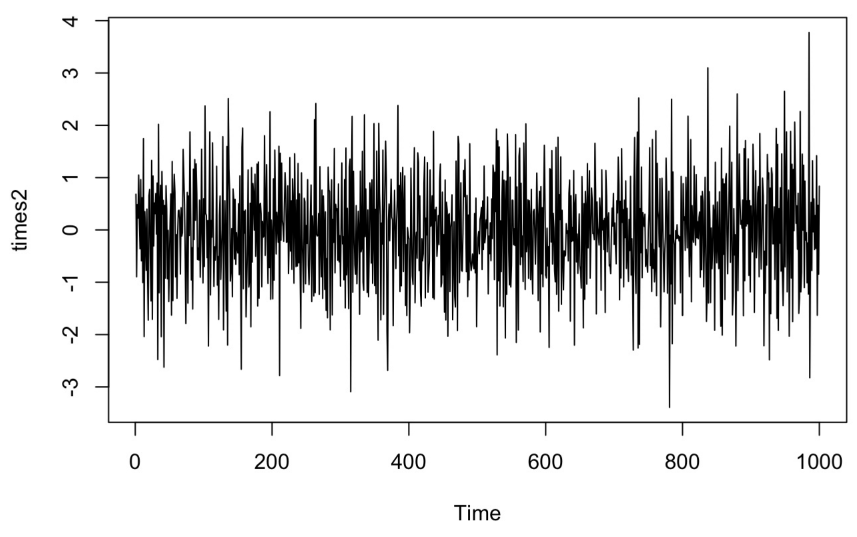
$x_t = \mu + w_t + \theta_1 w_{t-1}$

and the q^{th} order MA model is

$x_t = \mu + w_t + \theta_1 w_{t-1} + \theta_2 w_{t-2} + \dots + \theta_q w_{t-q}$

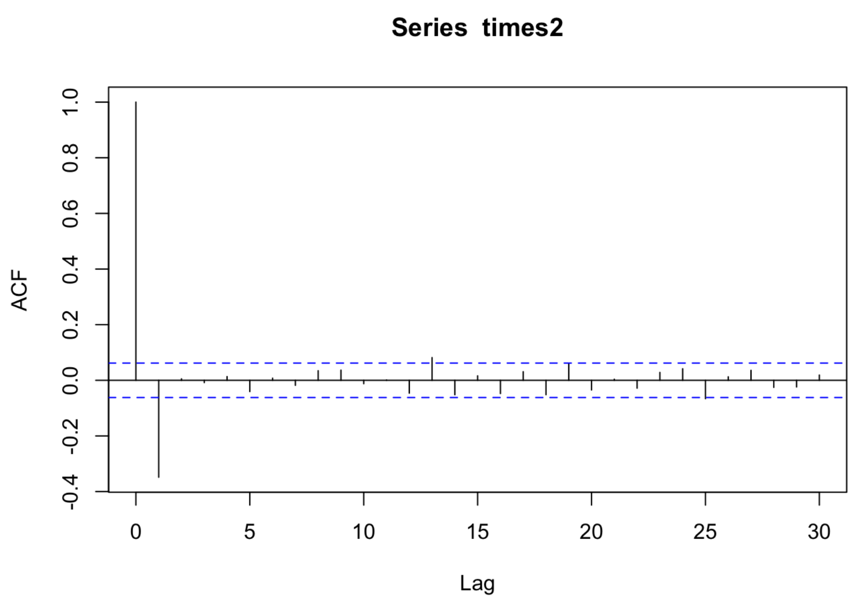
We use the autocorrelation function (ACF) to find the order of MA series. Similarly, we also look for where the vertical line disappears under the blue horizontal line. I will generate some more data to fit.

```
set.seed(101)
times2 <- arima.sim(n = 1000, model = list(order = c(0,0,1), ma = -.4))
plot(times2)
```

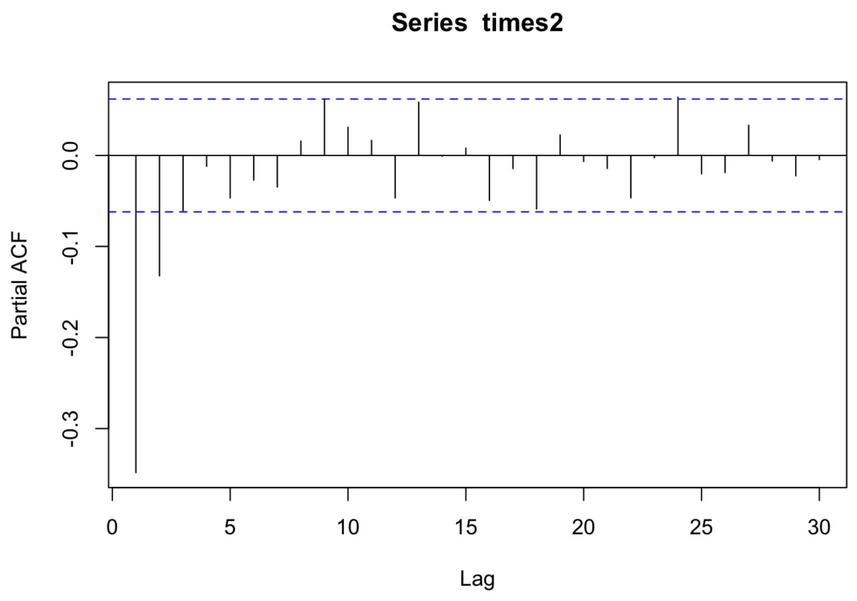


It probably looks like the same graph as the AR model doesn't it? It definitely does to me.

```
acf(times2) #the ACF component cuts off at 1, suggesting a MA(1) model.
```



```
pacf(times2)
```



```
arima(times2, c(0,0,2), include.mean = F)  #ma1 coefficient is significant. ma2 is not.
```

```
##
## Call:
## arima(x = times2, order = c(0, 0, 2), include.mean = F)
##
## Coefficients:
##           ma1      ma2
##       -0.4035  -0.0036
## s.e.   0.0314   0.0320
##
## sigma^2 estimated as 0.9204:  log likelihood = -1377.54,  aic = 2761.08
```

```
arima(times2, c(0,0,1), include.mean = F) #ma1 coefficient is significant.
```

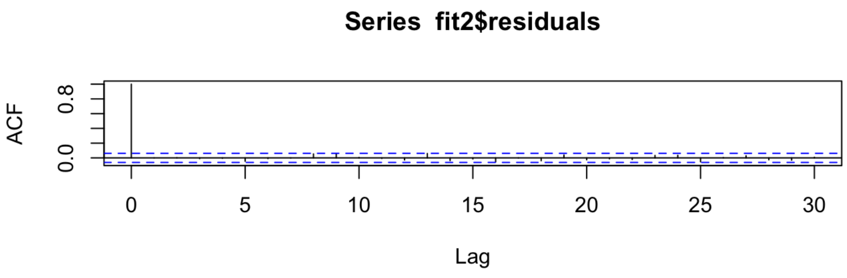
```
##
## Call:
## arima(x = times2, order = c(0, 0, 1), include.mean = F)
##
## Coefficients:
##           ma1
##       -0.4048
## s.e.   0.0293
##
## sigma^2 estimated as 0.9204:  log likelihood = -1377.55,  aic = 2759.1
```

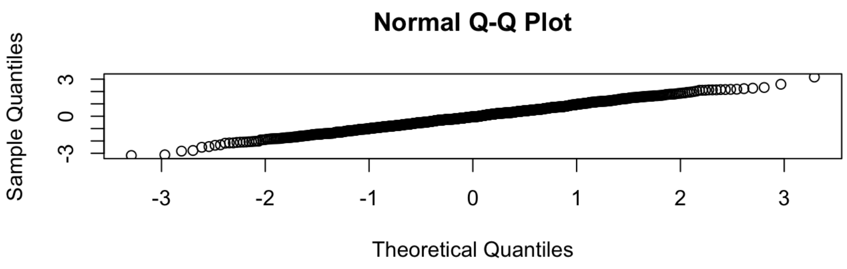
A closer look reveals log-likelihood and AIC are very similar between these two models. AIC is slightly lower for the MA(1) model. We end up choosing this because it is simpler and does a good job of explaining our observations! This is definitely not because we know it is a MA(1) model. It's not!

At this stage, we would need to again check the residuals. What do you think?

```
fit2 <- arima(times2, c(0,0,1), include.mean = F)

par(mfrow = c(2,1))
acf(fit2$residuals)
qqnorm(fit2$residuals)
```



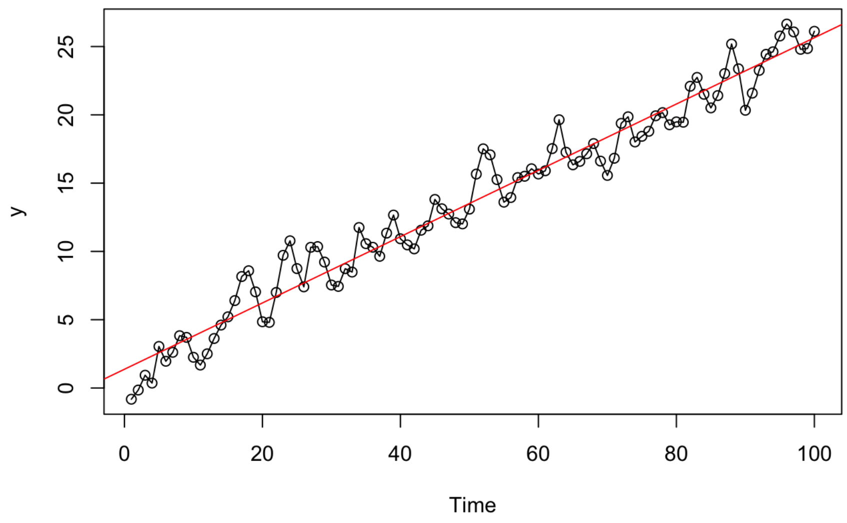


The Autoregressive Integrated Moving Average Model - ARMA(p,d,q)

Now, you may have been wondering why we have been using a function called `arima()`. Seeing AR and MA models from earlier, you may have been able to put two and two together. Yes! We can combine these two models to creat ARMA or ARIMA models. The "I" stands for integrated, and is often used to remove underlying trends from the data. You can think of trends as a persistent linear growth or decay within the data. I will show a graph below. The previous `times` and `times2` all had zero mean with no trend whereas this following graph will have a clear upward trend.

```
set.seed(102)
y.trs <- 1 + 0.25*seq(100) + arima.sim(n = 100, model = list(ar = c(.75,-.33)))
lm.y.trs <- lm(y.trs~time(y.trs))

plot(y.trs, type = "o", ylab = "y")
abline(lm.y.trs, col = "red") #the trend line
```



Without differencing `y.trs` because the data is *nonstationary*. Differencing the trend will stabilize the data give us something we can more readily work with because we are trying to explain the non-trend non-random portion of the data.

Anyways, combining these two together give the **Autoregressive Integrated Moving Average (ARIMA) model**. Commonly denoted ARIMA(p,d,q). It can be analyzed in a similar way we did earlier, but ARMA models are generally much more difficult to fit correctly. This is the general formula for an ARMA(1,1) or ARIMA(1,0,1) model.

$x_t = \phi_1 x_{t-1} + w_t + \theta_1 w_{t-1}$

Where the w_t are again errors that are iid N(0, σ_{2w}).

We can no longer use the `acf` and `pacf` functions since both will just tail off. Instead, I introduce the `eacf` function here. The numbers on the table represent the order of the AR/MA. The idea is to find a "o" that is as high in the upper left corner as possible (we want a simple model). Consequently, we also want the leftmost "o" that will give us "o"'s for the rest of the row on the right.

```
require(TSA) #Please excuse this mess. Due to technical difficulties, I must include this in the code chunk to mak
e eacf run.
```

```
## Loading required package: TSA
```

```
## Loading required package: leaps
```

```
## Loading required package: locfit
```

```
## locfit 1.5-9.1     2013-03-22
```

```
## Loading required package: mgcv
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.8-17. For overview type 'help("mgcv-package")'.
```

```
## Loading required package: tseries
```

```
##
## Attaching package: 'TSA'
```

```
## The following objects are masked from 'package:stats':
##
##     acf, arima
```

```
## The following object is masked from 'package:utils':
##
##     tar
```

```
dy.trs <- diff(y.trs)
eacf(dy.trs) #ARMA(2,2) looks proper.
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 o x x o x o o x o x o  o  x  o
## 1 x x x o x o o x o o x  o  o  o
## 2 x x o o o o o o o o o  o  o  o
## 3 x o o x o o o o o o o  o  o  o
## 4 x o x o o o o o o o o  o  o  o
## 5 o x o o o x o o o o o  o  o  o
## 6 x x x o x o o o o o o  o  o  o
## 7 x x o o x o o o o o o  o  o  o
```

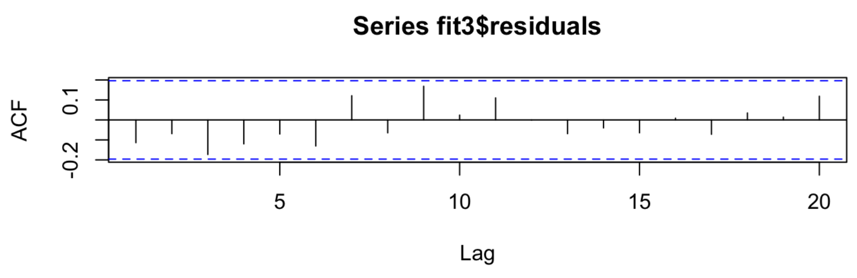
```
fit3 <- arima(y.trs, c(2,1,2)) #all significant coefficients. LL = -156.83, AIC = 321.65
arima(y.trs, c(2,1,0)) #significant coefficients. LL = -162.17, AIC = 328.35
```

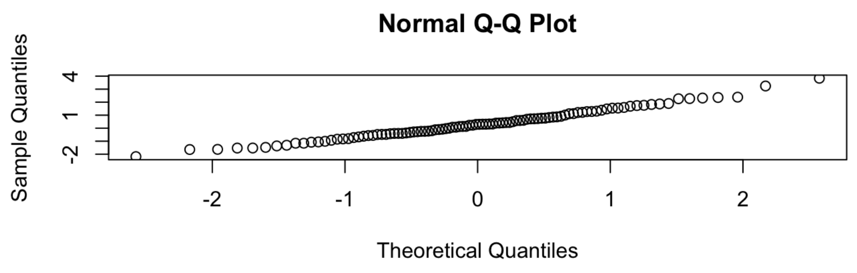
```
##
## Call:
## arima(x = y.trs, order = c(2, 1, 0))
##
## Coefficients:
##          ar1      ar2
##       0.2326  -0.3756
## s.e.  0.0932   0.0925
##
## sigma^2 estimated as 1.545:  log likelihood = -162.17,  aic = 328.35
```

```
arima(y.trs, c(3,1,1)) #not significant AR1, MA1 coefficient. LL = -156.9, AIC = 321.79
```

```
##
## Call:
## arima(x = y.trs, order = c(3, 1, 1))
##
## Coefficients:
##          ar1      ar2      ar3     ma1
##       0.0505  -0.2860  -0.3399  0.0704
## s.e.  0.2409   0.1038   0.1236  0.2499
##
## sigma^2 estimated as 1.384:  log likelihood = -156.9,  aic = 321.79
```

```
par(mfrow = c(2,1))
acf(fit3$residuals)
qqnorm(fit3$residuals) #not too linear, but can pass.
```





So we see here that even though we know the model is a AR(2) with a trend, actual analyses suggest ARIMA(2,1,2). I encourage you to try out other combinations of the ARIMA(p,d,q) models and check for yourself which one is the best.

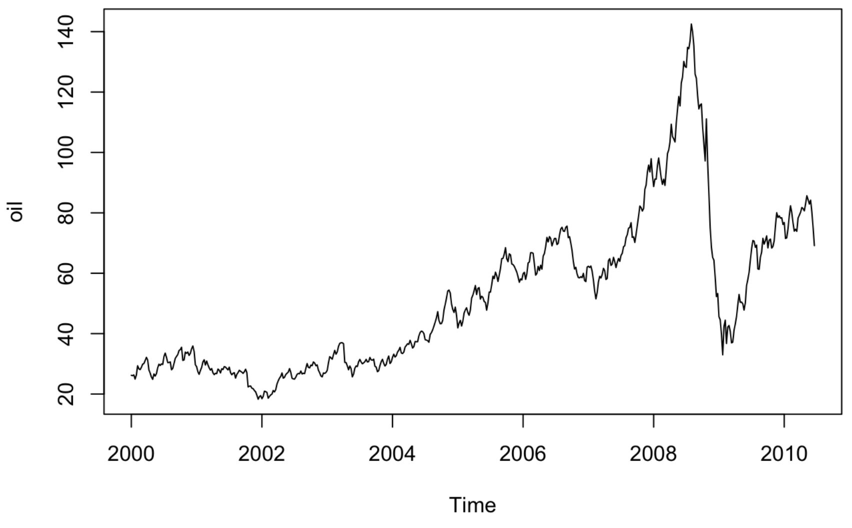
Application - Oil Prices

Now that we got through the dry stuff, we can try working with real data. I will use the `oil` data provided by Base R. It contains the crude oil spot price from 2000-mid2010 in weekly intervals. Use `?oil` to read the description.

```
require(astsa)
```

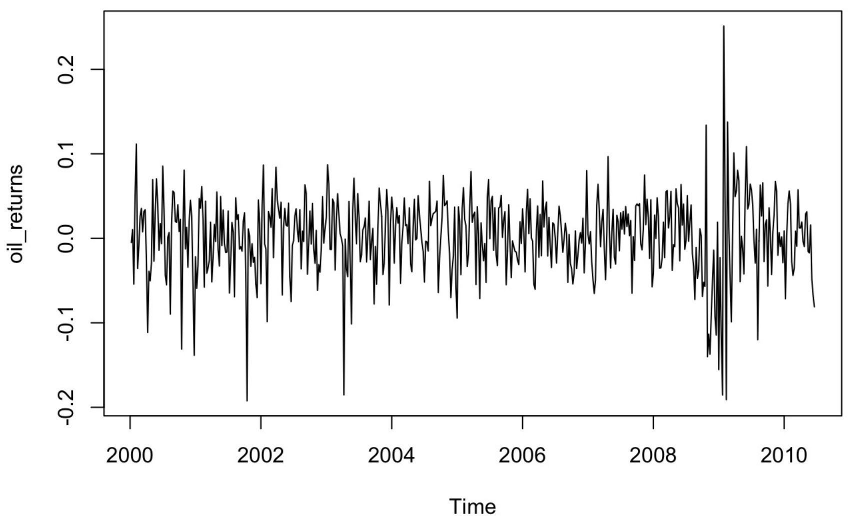
```
## Loading required package: astsa
```

```
plot(oil) #non stationary
diff(plot(oil)) #a strong trend still remains.
```



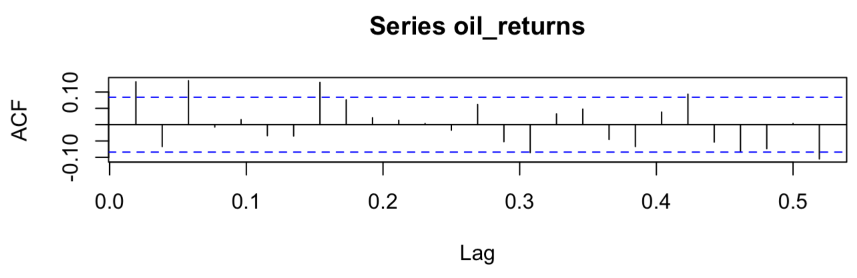
```
## NULL
```

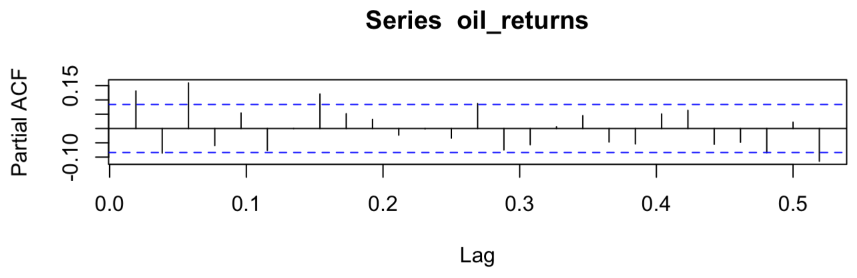
```
                #we can flatten the data by log-transforming it
oil_returns <- diff(log(oil))
plot(oil_returns) #better, but notice there are still sharp spikes
```



```
                  #be wary of outliers
```

```
par(mfrow = c(2,1))
acf(oil_returns) #oscillating, suggesting ARMA
pacf(oil_returns) #oscillating, suggests ARMA
```

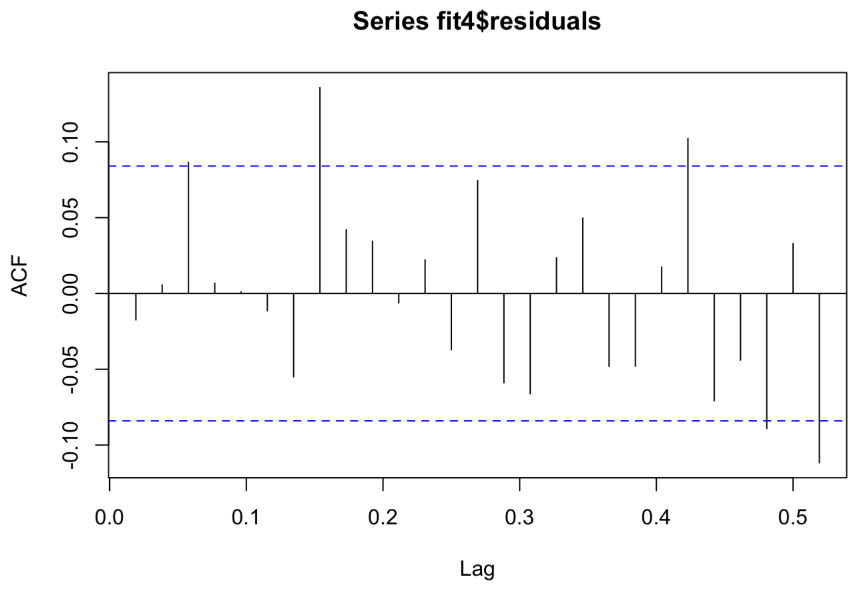




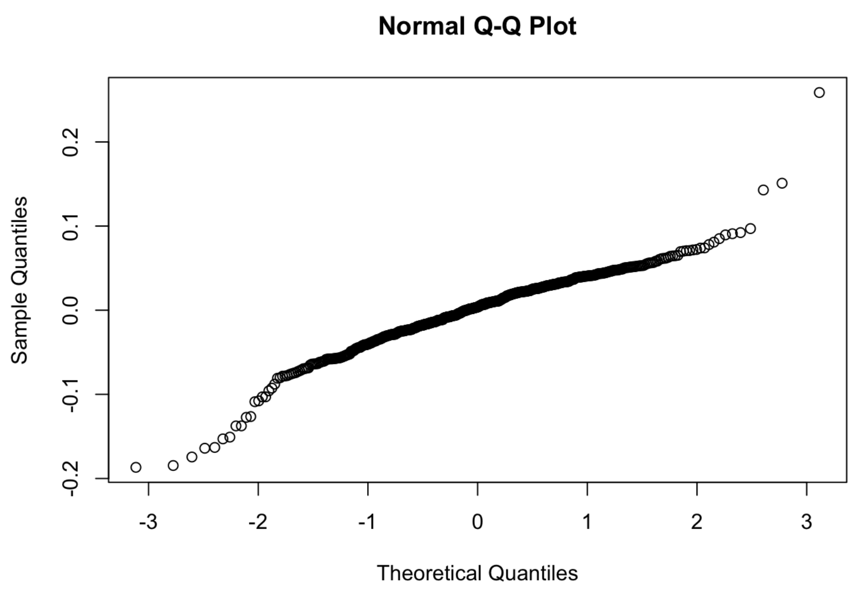
```
eacf(oil_returns) #ARMA(1,1)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x o x o o o o x o o o  o  o  o
## 1 x o x o o o o x o o o  o  o  o
## 2 x x x o o o o x o o o  o  o  o
## 3 x x x o o o o x o o o  o  o  o
## 4 x o x o o o o x o o o  o  o  o
## 5 x x x o x o o x o o o  o  o  o
## 6 o x x o x x o x o o o  o  o  x
## 7 o x x x x x x x o x o  o  o  o
```

```
fit4 <- arima(oil_returns, c(1,0,1))
acf(fit4$residuals) #some outliers
```



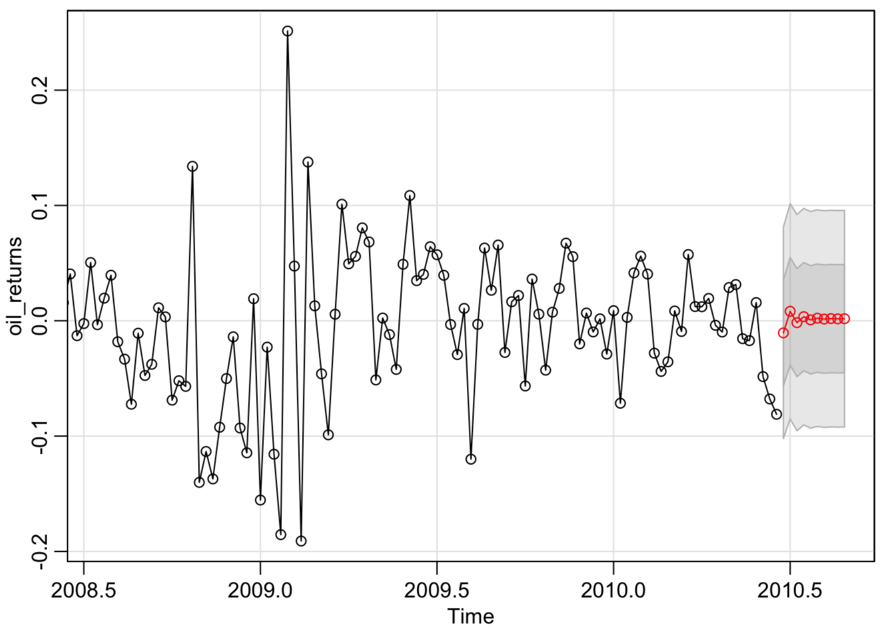
```
qqnorm(fit4$residuals) #not bad, but not good either
```



I have not introduced earlier, but we can forecast the differenced and logged oil returns through the `sarima.for` function. We would need an additional argument, `n.ahead` for the number of periods in the future you want predictions for. Often times, our best prediction for tomorrow is what we see today. In that sense, our predictions regress to the mean.

We need to be particularly careful of the standard errors. Below, a grey shadow is shown above and below the red predictions. The larger it is, the more likely it is that your prediction is incorrect. As a general rule of thumb, the farther into the future you want to see, the more erroneous your prediction is likely to be.

```
sarima.for(oil_returns, n.ahead = 10, 1,0,1)
```



```
## $pred
## Time Series:
## Start = c(2010, 26)
## End = c(2010, 35)
## Frequency = 52
##  [1] -0.0104962287  0.0082158706 -0.0016350537  0.0035509336  0.0008207873
##  [6]  0.0022580640  0.0015014143  0.0018997501  0.0016900475  0.0018004448
##
## $se
## Time Series:
## Start = c(2010, 26)
## End = c(2010, 35)
## Frequency = 52
##  [1] 0.04584754 0.04665243 0.04687306 0.04693402 0.04695090 0.04695558
##  [7] 0.04695688 0.04695724 0.04695734 0.04695736
```

Discussion and Conclusion

I hope I did a good job of running through some basic time series concepts with you. I myself am still quite rusty, but I had a good time relearning! There are so many more parts to Time Series Analysis - more complex modeling and forecasting - that I did not have time nor space to include. Particularly seasonality (taking into account seasonal differences in the data) is extremely relevant. An easy example to think about is the sales and expenditures right before the holidays - they are bound to increase especially before Christmas. This also means that sales and expenditures may be low for the first quarter of the next year.

To tie it all together in a short message, our every day choices may be captured in many time series models (like the scenario I mentioned above). The models itself try to pick out the differences that are attributed to randomness and the differences that are attributed to something innate that can be further explained. With the oil prices, we have shown that it can be modeled through ARMA(1,1) after differencing and logging. By reversing the transformations, we can get nominal predictions too! If you are interested in further readings, I strongy recommended *Time Series Analysis with Applications in R* by Jonathan Cryer and Kung-Sik Chan!

References

1. Time Series Wiki
2. Datacamp, Intro to Time Series
3. Datacamp, ARIMA Modeling in R
4. Penn State, Applied Time Series Analyses
5. Data, Base R - oil 1700-1988
6. Time Series Analysis with Applications in R - Cryer, Jonathan and Chan, Kung-Sik

Processing math: 100%

Some of these sources I’ve used more than once. I hope to use that toward the 7-source count!