Time Series Analysis of S&P500

A Time Series Modelling of the S&P500

Load Packages

```r
library("tseries")
library("forecast")
```

Collecting the Data

We will collect the S&P500 Stock Index monthly data starting from January 1, 1996 to April 1, 2019. This is an arbitrary range, you may choose a larger range if you prefer. We will get our data from Yahoo Finance, as it is a reliable source. From Yahoo Finance to get a csv file containing the Open, High, Low and Close price of S&P500.

Reading the Data

We will simply read the csv file we got from Yahoo Finance into R.

```r
sp500 = read.csv(file="/Users/aditya/Documents/SP500/SP500.csv", header=TRUE)
```

Data Overview

I've presented the first 12 rows as a visual.

```r
head(sp500,12)
```

```
##          Date   Open   High    Low  Close Adj.Close     Volume
## 1  1996-01-01 615.93 636.18 597.29 636.02    636.02 9188050000
## 2  1996-02-01 636.02 664.23 633.71 640.43    640.43 8749960000
## 3  1996-03-01 640.43 656.97 627.63 645.50    645.50 8984200000
## 4  1996-04-01 645.50 656.68 624.14 654.17    654.17 8875580000
## 5  1996-05-01 654.17 681.10 630.07 669.12    669.12 8921140000
## 6  1996-06-01 669.12 680.32 658.75 670.63    670.63 7930840000
## 7  1996-07-01 670.63 675.88 605.88 639.95    639.95 8849860000
## 8  1996-08-01 639.95 670.68 639.49 651.99    651.99 7380320000
## 9  1996-09-01 651.99 690.88 643.97 687.33    687.33 8064070000
## 10 1996-10-01 687.31 714.10 684.44 705.27    705.27 9703670000
## 11 1996-11-01 705.27 762.12 701.30 757.02    757.02 8763850000
## 12 1996-12-01 757.02 761.75 716.69 740.74    740.74 9089170000
```

Here is the summary of the data.

```r
summary(sp500)
```

```
##          Date          Open             High             Low
##  1996-01-01:  1   Min.   : 615.9   Min.   : 636.2   Min.   : 597.3
##  1996-02-01:  1   1st Qu.:1098.8   1st Qu.:1130.3   1st Qu.:1049.5
##  1996-03-01:  1   Median :1287.1   Median :1325.3   Median :1253.8
##  1996-04-01:  1   Mean   :1429.1   Mean   :1474.8   Mean   :1380.9
##  1996-05-01:  1   3rd Qu.:1615.3   3rd Qu.:1690.1   3rd Qu.:1587.1
##  1996-06-01:  1   Max.   :2926.3   Max.   :2949.5   Max.   :2864.1
##  (Other)   :274
##      Close          Adj.Close         Volume
##  Min.   : 636   Min.   : 636   Min.   :7.380e+09
##  1st Qu.:1101   1st Qu.:1101   1st Qu.:2.598e+10
##  Median :1286   Median :1286   Median :6.323e+10
##  Mean   :1437   Mean   :1437   Mean   :5.569e+10
##  3rd Qu.:1631   3rd Qu.:1631   3rd Qu.:7.957e+10
##  Max.   :2946   Max.   :2946   Max.   :1.618e+11
##
```

Exploratory Analysis

We want to understand how the index is behaving overtime in order to model and forecast the S&P500. We want to ask questions like:

- What's the trend?
- Is there cyclical/seasonal behaviour?
- Is our seires stationary?
- How does the ACF and PACF look?
- Will our series be best represented by an autogressive (AR) model, a moving average (MA) model or an ARIMA model (cominbation of both AR and MA models)?

Some Time Series Terminology

Before we continue, I'd like to review some time series terminology.

What does it mean for a series to be stationary? We say a stochastic process $\{Y_t\}$ is stationary if

- the expected value of Y_t is constant overtime and
- the autocovarice of $Y_{t,t-h}$ is equal to the autocovariance of $Y_{0,h}$.

What is ACF? ACF stands for autocorrelation function. Autocorrelation occurs when the error terms of a regression forecasting model are correlated. It's used to detect the order of a moving average process.

What is PACF? PACF stands for partial autocorrelation function. It's the correlation between Y_t and Y_{t-h}. It's used to detect the order of an autogressive process.

What is an autoregressive model? An autoregressive process is a regression on itself. It's where the current value Y_t is a linear combination of the past values plus an error term e_t which incorporates everyting new in the series at time t that is not explained in the past values. Here is an example of a third order autoregressive process: $Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \phi_3 Y_{t-3} + e_t$.

What is a moving average model? A moving average process is one whose current value Y_t is a linear combination of the past error terms plus e_t which incorporates everyting new in the series at time t that is not explained in the past values. Here is an example of a second order moving average process: $Y_t = \theta_1 e_{t-1} + \theta_2 e_{t-2} + e_t$.

What is an ARMA model? An ARMA model is partly autoregressive and partly moving average. Here is an examle of ARMA(3,2) model: $Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \phi_3 Y_{t-3} + \theta_1 e_{t-1} + \theta_2 e_{t-2} + e_t$.

Creating a Time Series Object

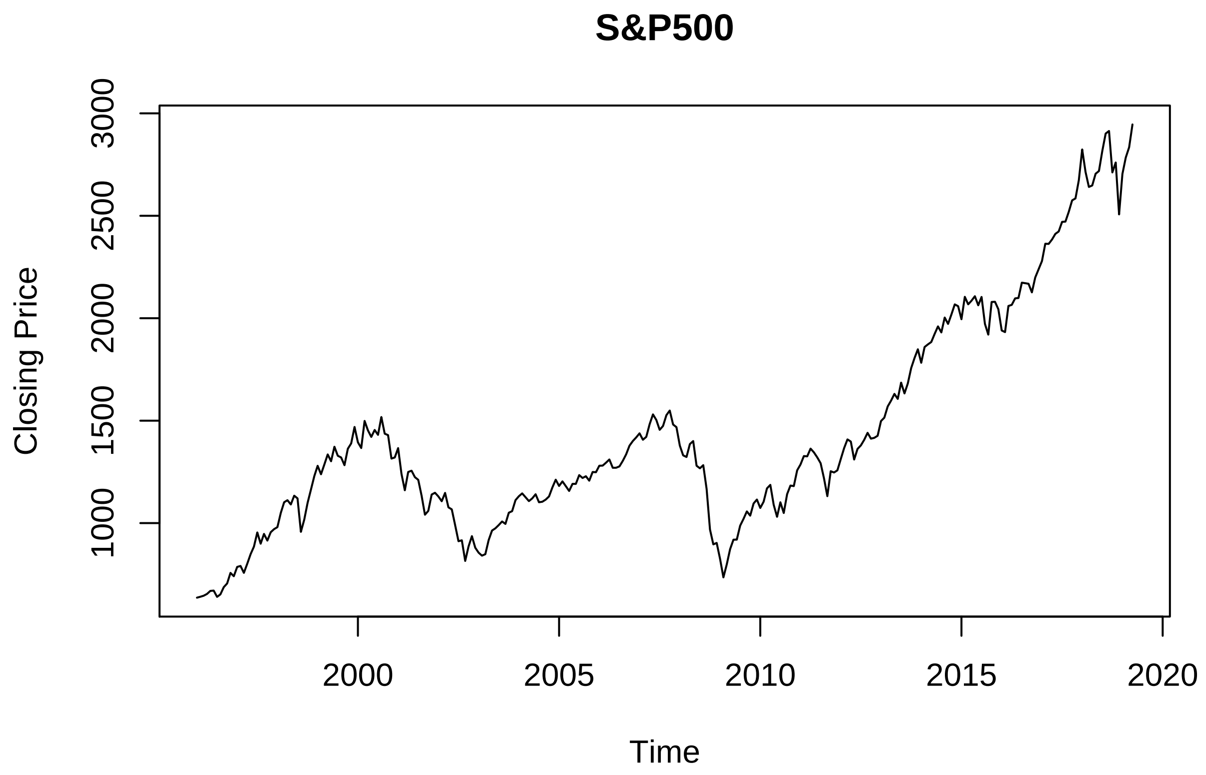
Here we're going to create a time series object of the S&P500. We can choose to make it for the Open, High, Low or Close. I will create it for the Close price.

```r
sp500.close.ts = ts(sp500$Close, start=c(1996,1), freq=12)
```

Plotting the Time Series

Now that we have created a time series object of the S&P500, we can simply plot it.

```r
plot(sp500.close.ts, ylab="Closing Price", main="S&P500")
```



Testing for Stationarity

From observing our time series plot, it does not seem that our series is stationary. We notice that the mean is not constant, it is increasing with time. To confirm our observation, we can conduct the Dickey-Fuller Test by calling the function `adf.test()`. Here our null hypothesis is that the series is non-stationary and our alternative hypothesis is that the series is stationary.

```r
adf.test(sp500.close.ts)
```

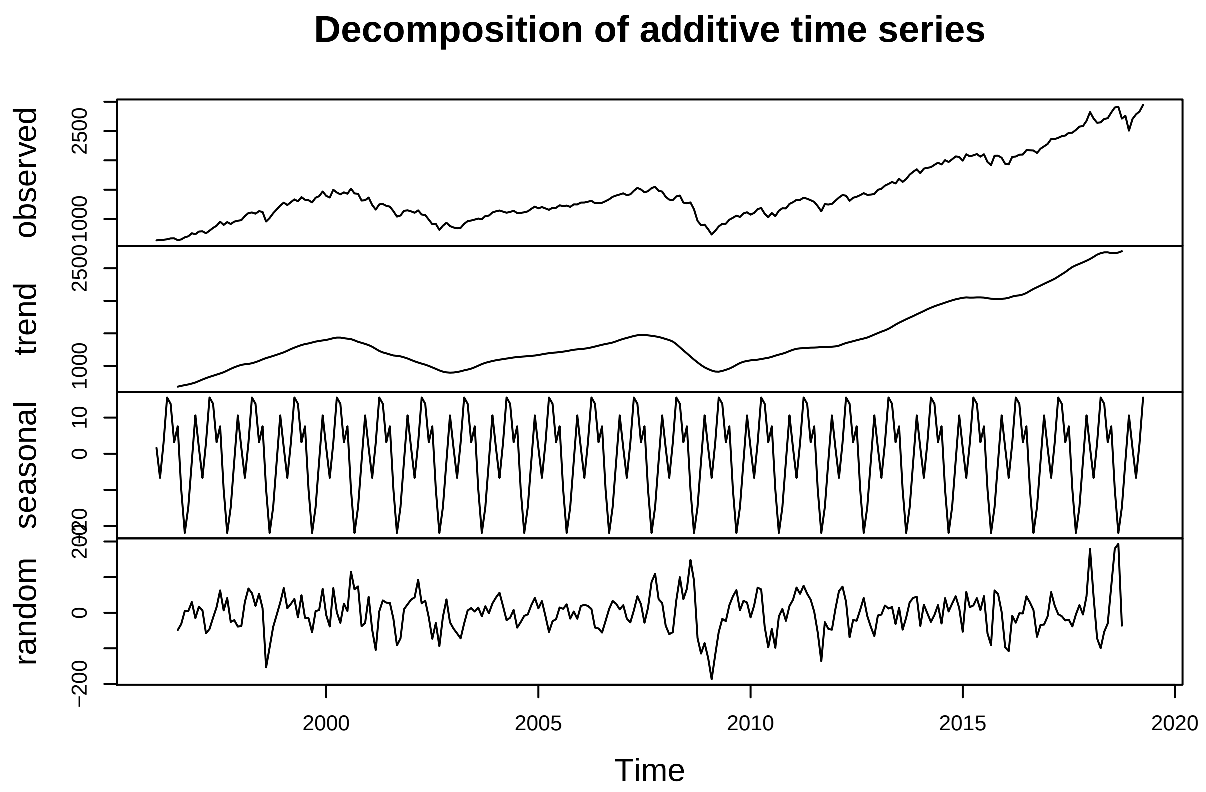
```
##
##  Augmented Dickey-Fuller Test
##
## data:  sp500.close.ts
## Dickey-Fuller = -0.50363, Lag order = 6, p-value = 0.9813
## alternative hypothesis: stationary
```

Since the p-value is 0.9813 we fail to reject the null hypothesis.

Decomposing the Time Series

Here we're going to "decompose" our time series. We are going to separate the trend, the seasonality pattern and the random errors within our series. This can be achieved with the `decompose()` function.

```r
sp500.decomposed = decompose(sp500.close.ts)
plot(sp500.decomposed)
```



This allows us to see the general trend of our time series, we see that it's increasing over time. We also notice that the seasonal trend is bounded between 15 and -20. In the whole realm of things, this is quite negligible. Therefore, we can exclude the seasonlity portion when estimating our model.

Model Estimation and Model Selection

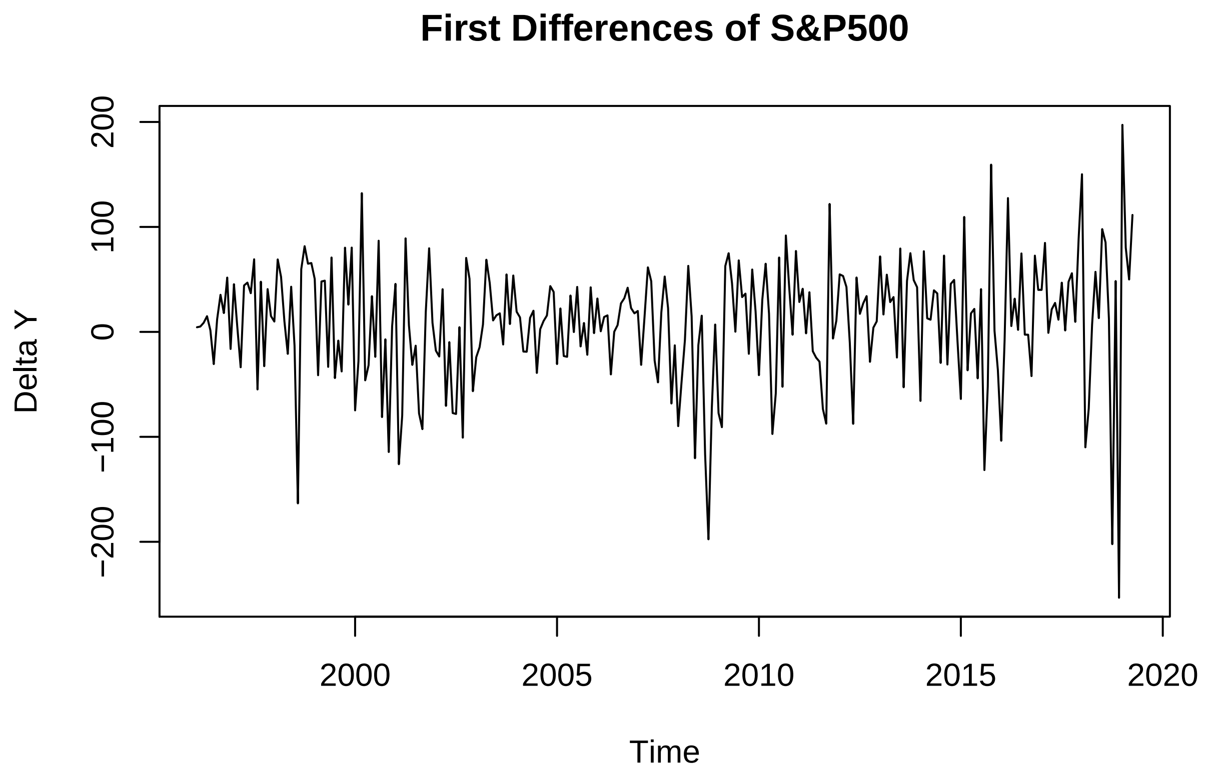
In this step we will be investigating the time series to see which model is the most appropriate. Will it be a moving average model? Will be it an autoregressive model? Or will it be an ARMA model? What would be the order for each one? However, before we can determine the model and the order for the series, we must first make our series stationary.

Stationarity Through Differencing

For example, suppose we have an AR(1) model: $Y_t = Y_{t-1} + e_t$. This model is not stationary since the coefficient of $Y_{t-1} \geq 1$. However, we know that the error terms are white noise and are normally distributed. If we rewrite the AR(1) model in the following form: $\Delta Y_t = e_t$, where $\Delta Y_t = Y_t - Y_{t-1}$, then ΔY_t is stationary and is called the first difference of Y_t.

We will use this transformation to make our time series stationary. To perform this tranformation we will use the `diff()` function.

```
##Setting difference to 1 is calculating the first differences.
sp500.diff = diff(sp500.close.ts, differences = 1)
plot(sp500.diff, ylab="Delta Y", main="First Differences of S&P500")
```



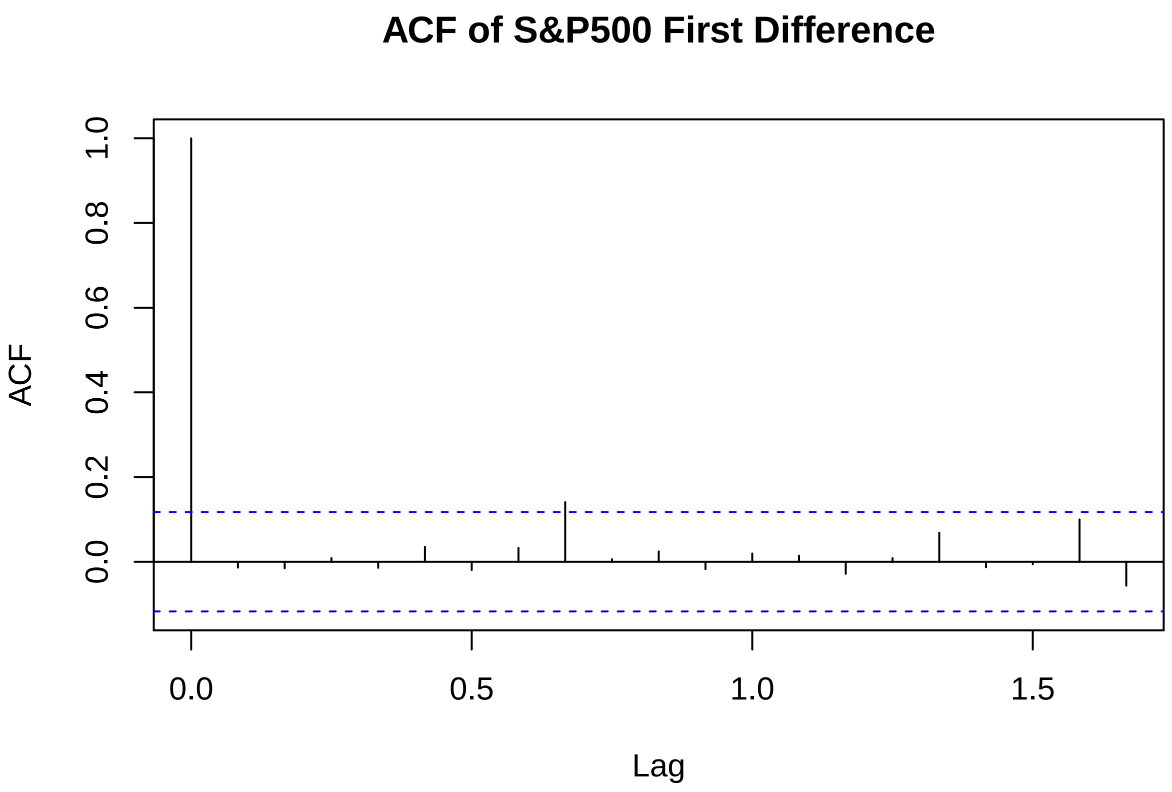
Selecting an Appropriate Model

Since we have transformed our time series by differencing, we must choose an appropriate ARIMA(p,d,q) model, where p is the order of the moving average process, d is the number of times we differenced our series and q is the order of the autoregressive process.

Next we must examine the ACF and PACF to determine the order of our model.

ACF:

```
acf(sp500.diff, lag.max=20, main="ACF of S&P500 First Difference")
```

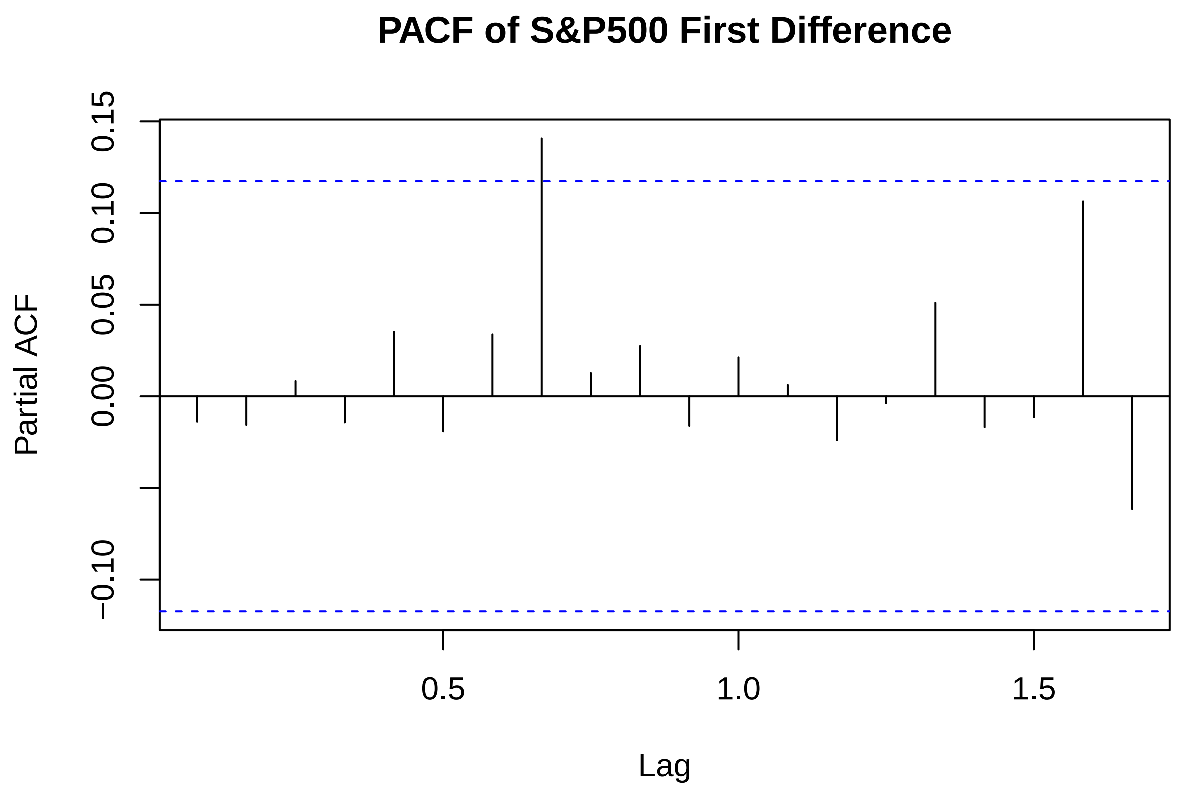


```
##
## Autocorrelations of series 'sp500.diff', by lag
##
## 0.0000 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500
##  1.000 -0.014 -0.015  0.009 -0.014  0.035 -0.020  0.033  0.141  0.006
## 0.8333 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833
##  0.024 -0.017  0.019  0.014 -0.029  0.009  0.069 -0.013 -0.006  0.100
## 1.6667
## -0.057
```

Remember the ACF functions tells us the magnitude of the correlation at different time lags and is used to examine the order of a moving average process. From the correlogram, we see the autocorrelation at lag 8 (0.141) exceeds the signifance bound and all the others do not. Therefore, the potential order for the moving average portion of our model may be 8. Since it only sightly crosses the significance bound at lag 8, it may be negligible and we could proceed with having a moving average of order 0. We would need to compare the two models, one with p = 8 and the other with p = 0, in order to see which produces better results.

PACF:

```
pacf(sp500.diff, lag.max=20, main="PACF of S&P500 First Difference")
```



```
##
## Partial autocorrelations of series 'sp500.diff', by lag
##
## 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500 0.8333
## -0.014 -0.016  0.008 -0.014  0.035 -0.019  0.034  0.141  0.013  0.027
## 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833 1.6667
## -0.016  0.021  0.006 -0.024 -0.004  0.051 -0.017 -0.011  0.106 -0.062
```

Observing the PACF, we notice that the partial autocorrelation at lag 8 (0.141) exceeds the signifiance bound. Therefore, the potential order for the autoregressive portion of our model may be 8.

Now we have some potential ARIMA models:

- ARIMA(0,1,0)
- ARIMA(8,1,0)
- ARIMA(0,1,8)
- ARIMA(8,1,8)

We need to investigate futher to see which model is the best fit. We can do this manually, which is tedious, or alternatively we can use the `auto.arima()` function to find the appropriate ARIMA model. This function returns the best ARIMA model according to Akaike's Information Criteria. It finds the optimal balance between the complexity of the model and the fit of the model.

Model Comparison Example

For the sake of understanding, I'll demonstrate how we compare models manually. Let's first pick two models to represent our time series. Suppose we pick a naive linear model and a naive autoregressive model to represent our time series.

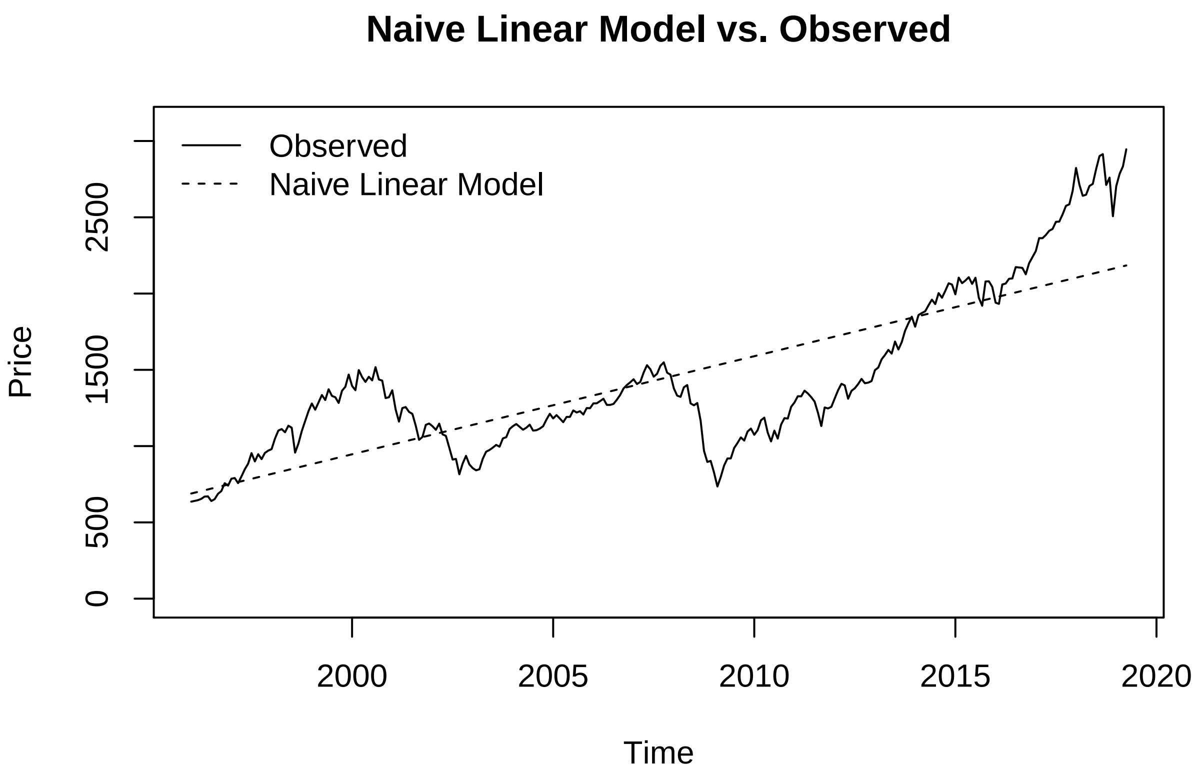
Naive Linear Model

We are going to a fit a naive linear model using `lm()` function.

```
Time = 1:280
lin.model = lm(sp500$Close ~ Time)

##Creaing time series object
naive.lin.model = ts(lin.model$fitted.values, start=c(1996,1), freq=12)

##Comparing our time series to our naive linear model
plot(sp500.close.ts, ylim=c(0,3100), ylab="Price", main="Naive Linear Model vs. Observed")
lines(naive.lin.model, lty=2)
legend("topleft", bty="n", legend=c("Observed", "Naive Linear Model"), lty=1:2)
```



Naive Autoregressive Model

We are going to fit a naive autoregressive model using `ar.yw()` function.

```
ar.model = ar.yw(sp500.close.ts)
ar.model$aic
```

```
##          0          1          2          3          4          5
## 851.600499   0.000000   1.726645   3.696073   5.608734   6.126263
##          6          7          8          9         10         11
##   4.493680   6.491458   5.556051   6.910281   8.897960  10.569881
##         12         13         14         15         16         17
##  12.397049  14.166493  16.161022  17.810793  18.754359  20.447819
##         18         19         20         21         22         23
##  22.405381  24.404990  26.146398  27.598075  29.436028  30.887519
##         24
##  32.866788
```

```
ar.model$ar
```

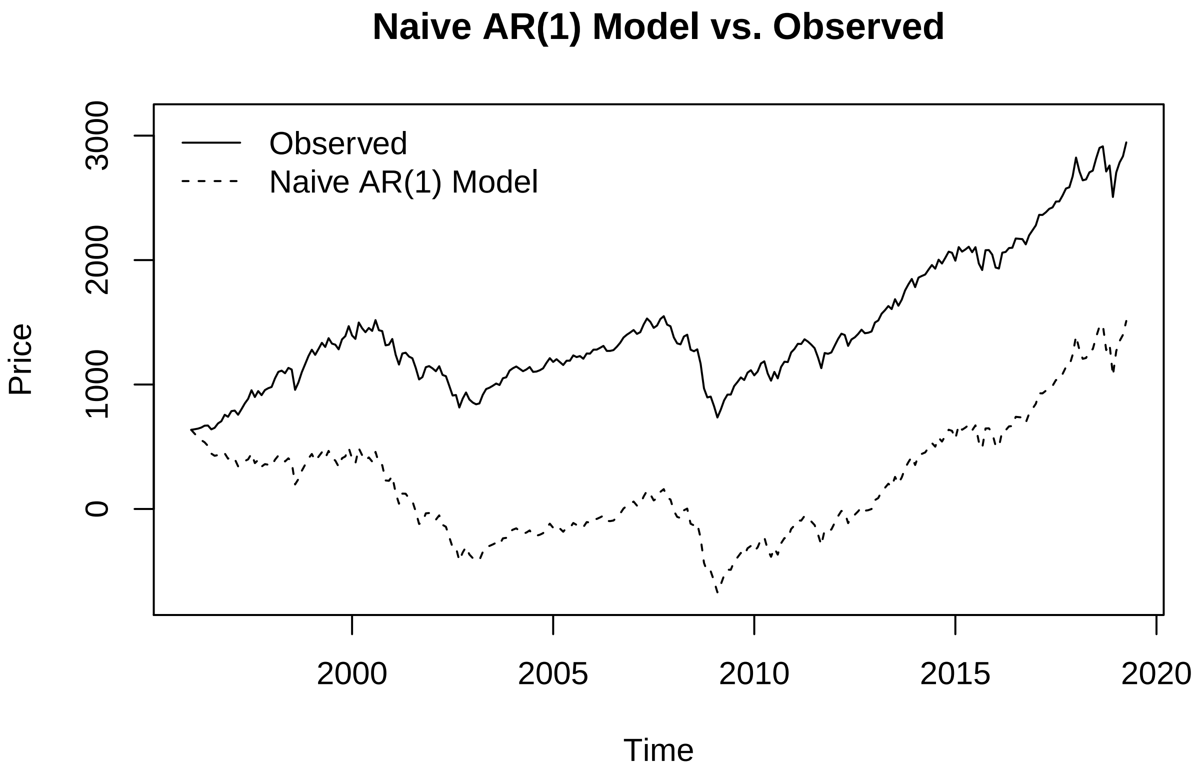
```
## [1] 0.9759987
```

Since AIC is minimized at 1, we are looking at an AR(1) model: $Y_t = 0.9759987Y_{t-1} + e_t$. Next, we will find Y_t estimates using the AR(1) model and the residuals.

```
y.estimate = numeric(280)
y.estimate[1] = sp500$Close[1]
for (t in 2:length(y.estimate)) {
  y.estimate[t] = ar.model$ar * y.estimate[t-1] + ar.model$resid[t]
}

##Creating time series object
naive.ar.model = ts(y.estimate, start=c(1996,1), freq=12)

##Comparing our time series to our naive AR(1) model
plot(sp500.close.ts, ylim=c(-700,3100), ylab="Price", main="Naive AR(1) Model vs. Observed")
lines(naive.ar.model, lty=2)
legend("topleft", bty="n", legend=c("Observed", "Naive AR(1) Model"), lty=1:2)
```



We have our two models. We must calculate Akaike's Information Criteria for each model and the model with the lower AIC value will be a better fit. Akaike's Information Criteria: $AIC = n \ln(\text{SSE}) + 2(p+1)$ where n is the number of residuals, p is the number of parameters and SSE is the sum of squared residuals.

Calculating AIC for Naive Linear Model

```
SSE.lin = sum(lin.model$residuals^2)
AIC.lin = 280*log(SSE.lin) + 2*(2 + 1)
AIC.lin
```

```
## [1] 4812.003
```

Calculating AIC for Naive AR(1) Model

```
SSE.ar = 0
for (i in 2:length(ar.model$resid)) {
  SSE.ar = SSE.ar + ar.model$resid[i]^2
}
AIC.ar = 279*log(SSE.ar) + 2*(1+1)
AIC.ar
```

```
## [1] 3866.961
```

In this case, since AIC for naive AR(1) model is lower, the naive AR(1) model is a better fit.

Using `auto.arima()` to Find the Best ARIMA Model

As mentioned earlier, instead of finding a potential model manually and then comparing each model to find the best fit, we can alternatively use `auto.arima()` function to do all the tedious work for us.

```
sp500.arima = auto.arima(sp500.close.ts)
sp500.arima
```

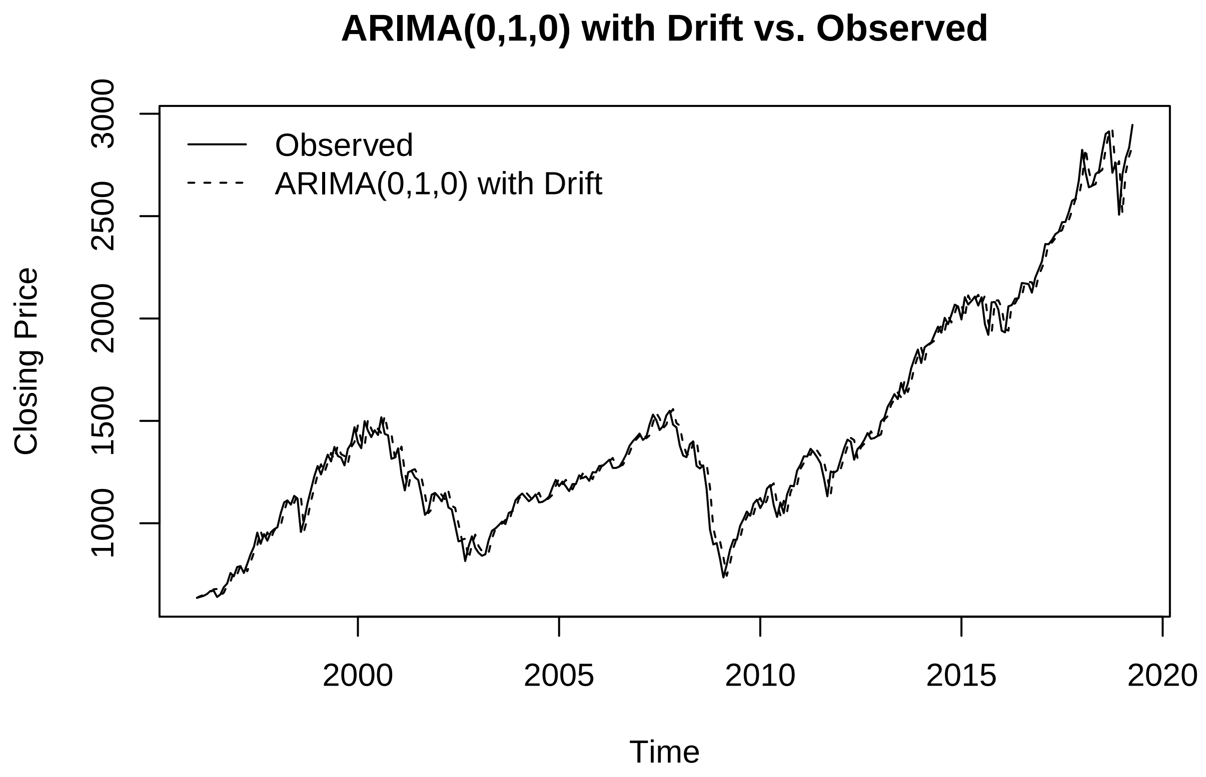
```
## Series: sp500.close.ts
## ARIMA(0,1,0) with drift
##
## Coefficients:
##        drift
##       8.2789
## s.e.  3.4939
##
## sigma^2 estimated as 3418:  log likelihood=-1530.47
## AIC=3064.94   AICc=3064.98   BIC=3072.2
```

The `auto.arima()` function found the best model to be ARIMA(0,1,0) with drift. Notice the AIC here is 3064.94, much lower than the naive models before.

Let's plot the model against the observed values to get a visual.

```
##Creating time series object
sp500.arima.ts = ts(sp500.arima$fitted, start=c(1996,1), freq=12)

##Comparing our time series with the ARIMA model
plot(sp500.close.ts, ylab="Closing Price", main="ARIMA(0,1,0) with Drift vs. Observed")
lines(sp500.arima.ts, lty=2)
legend("topleft", bty="n", legend=c("Observed", "ARIMA(0,1,0) with Drift"), lty=1:2)
```

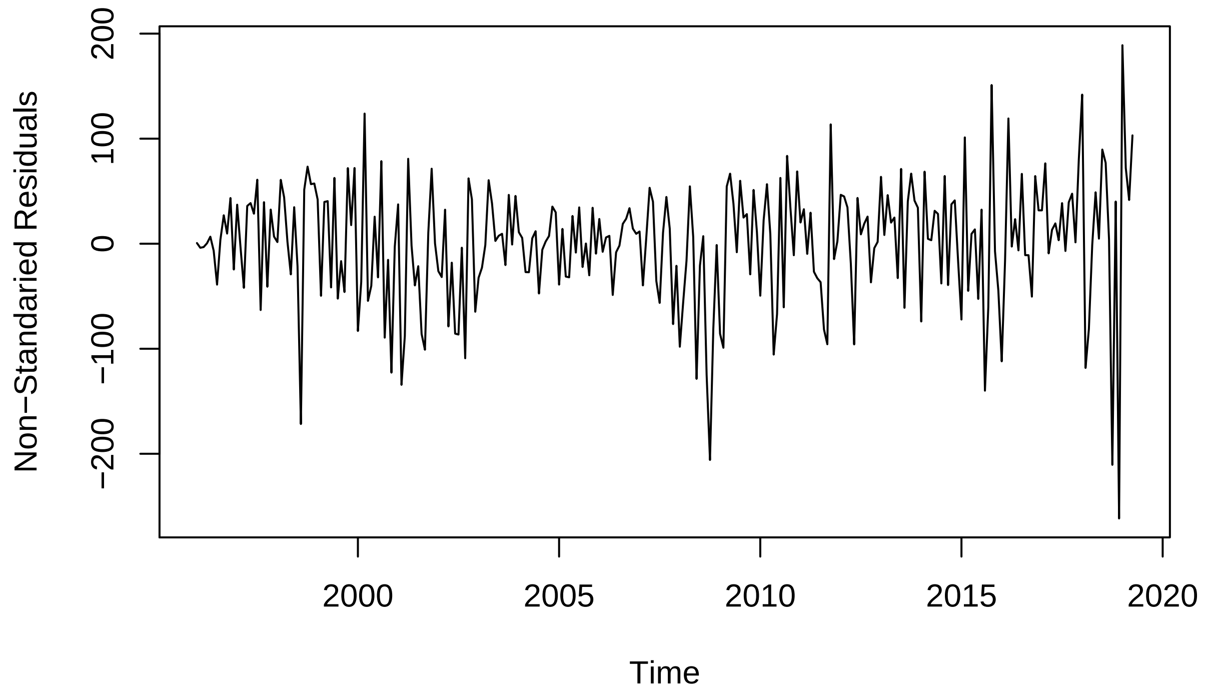


Residual Analysis of Our Model

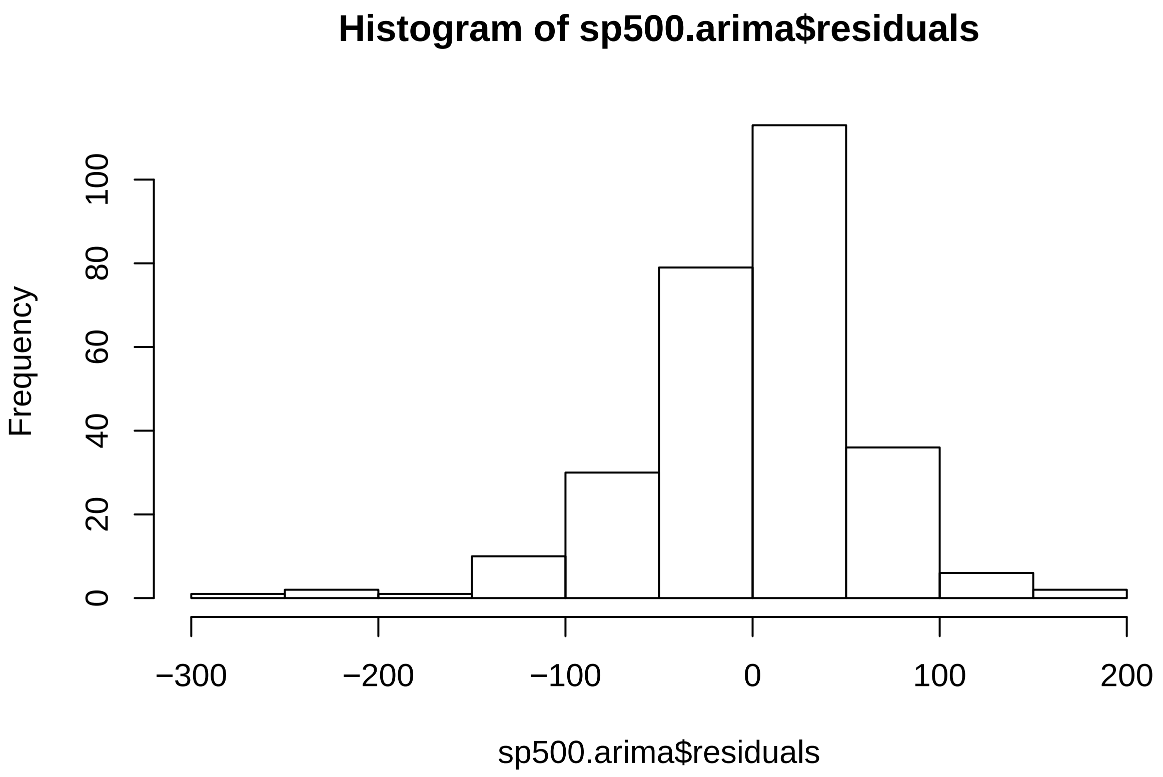
We need to conduct an analysis of our residuals to confirm they meet the assumptions of being normally distributed with mean zero. We should also check the ACF of our residuals and conduct the Ljung-Box test to check if there is autocorrleation present.

Check Constant Mean and Distribution

```
plot.ts(sp500.arima$residuals, ylab="Non-Standaried Residuals")
```



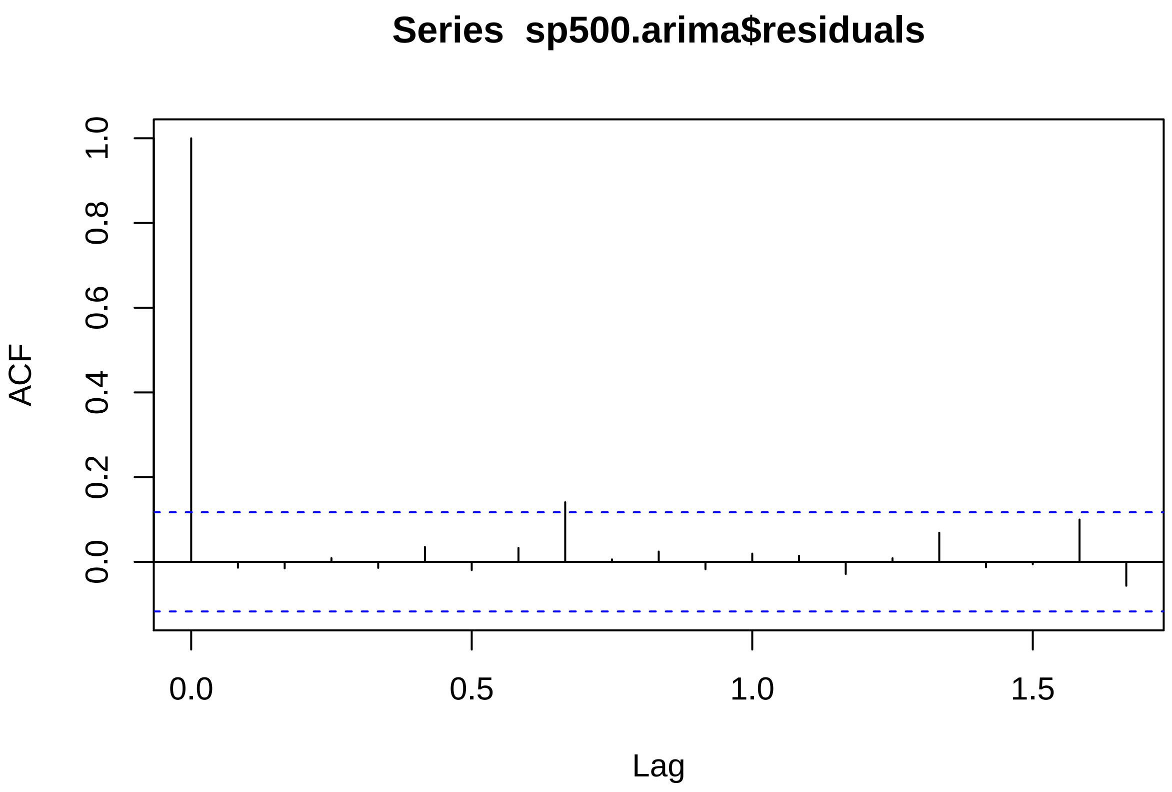
```
hist(sp500.arima$residuals)
```



The plot shows the distribution of our residuals is roughly centered at zero and is approximately normally distribued, though there is a slight skewness.

Check for Autocorrelation

```
acf(sp500.arima$residuals, lag.max = 20)
```



```
Box.test(sp500.arima$residuals, lag = 20, type="Ljung-Box")
```

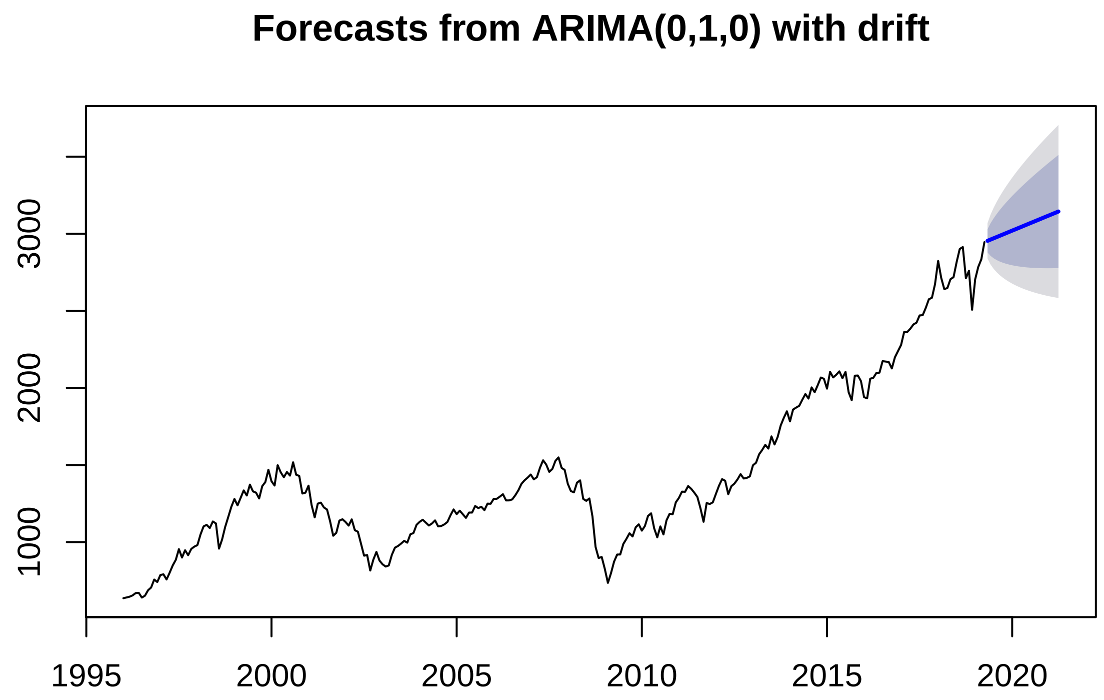
```
##
##  Box-Ljung test
##
## data:  sp500.arima$residuals
## X-squared = 12.912, df = 20, p-value = 0.8811
```

In the correlogram we observe the autocorrelation at lag 8 exceeds the 95% significance bound. However, we can expect one in twenty autocorrelations to exceed the bound. Looking at the Ljung-Box test, we get a p-value of 0.8811, implying that there is little evidence of autocorrelation.

Forecasting

Now that we have selected the best ARIMA(p,d,q) model for our time series, we can use it as a predictive model to create forecasts for the future. We can do this manually by estimating the parameters of our ARIMA model or we can simply use the `forecast()` function.

```
sp500.forecast = forecast(sp500.arima)
plot(sp500.forecast)
```


Forecasts from ARIMA(0,1,0) with drift
1000
2000
3000
1995
2000
2005
2010
2015
2020