Time Series Modeling of the S&P 500 Index

Objectives

This analysis aims to perform time series modeling of the S&P 500 index in Jan 2018 using ARIMA with a given data set (1 Jan 2010 to 29 Dec 2017).

The price data is obtained using the quantmod R package.

```
library("quantmod")
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: TTR
```

```
## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```
# Obtain data online
# ^GSPC = S&P 500 Index
getSymbols("^GSPC", from="2010-1-1")
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by settin
g
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
##
## WARNING: There have been significant changes to Yahoo Finance data.
## Please see the Warning section of '?getSymbols.yahoo' for details.
##
## This message is shown once per session and may be disabled by settin
g
## options("getSymbols.yahoo.warning"=FALSE).
```

```
## [1] "GSPC"
```

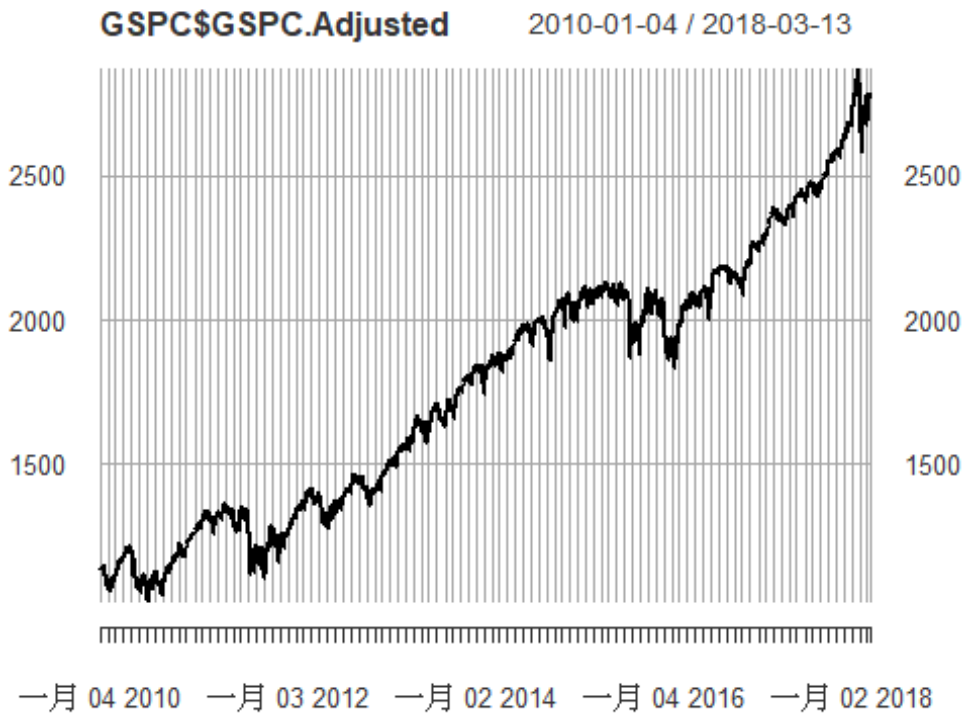
```
head(GSPC)
```

```
##            GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume
## 2010-01-04   1116.56   1133.87  1116.56    1132.99  3991400000
## 2010-01-05   1132.66   1136.63  1129.66    1136.52  2491020000
## 2010-01-06   1135.71   1139.19  1133.95    1137.14  4972660000
## 2010-01-07   1136.27   1142.46  1131.32    1141.69  5270680000
## 2010-01-08   1140.52   1145.39  1136.22    1144.98  4389590000
## 2010-01-11   1145.96   1149.74  1142.02    1146.98  4255780000
##            GSPC.Adjusted
## 2010-01-04       1132.99
## 2010-01-05       1136.52
## 2010-01-06       1137.14
## 2010-01-07       1141.69
## 2010-01-08       1144.98
## 2010-01-11       1146.98
```

```
#Take a brief look at the data
```

There are different types of available data. The adjusted close price is used in the price modeling in this analysis. It would be easier to visualize the adjusted close price with a plot.

```
plot(GSPC$GSPC.Adjusted)
```



The plot of the adjusted price shows a generally increasing trend from 2010 to the early Mar 2018. Moreover, non-constant variance and non-constant mean are observed in this

plot. I am curious to check if the adjusted price time series is stationary. Stationarity of a time series is very important to using ARIMA as the modeling technique.

To check for the stationarity of a time series, the Augmented Dickey-Fuller statistical test is performed.

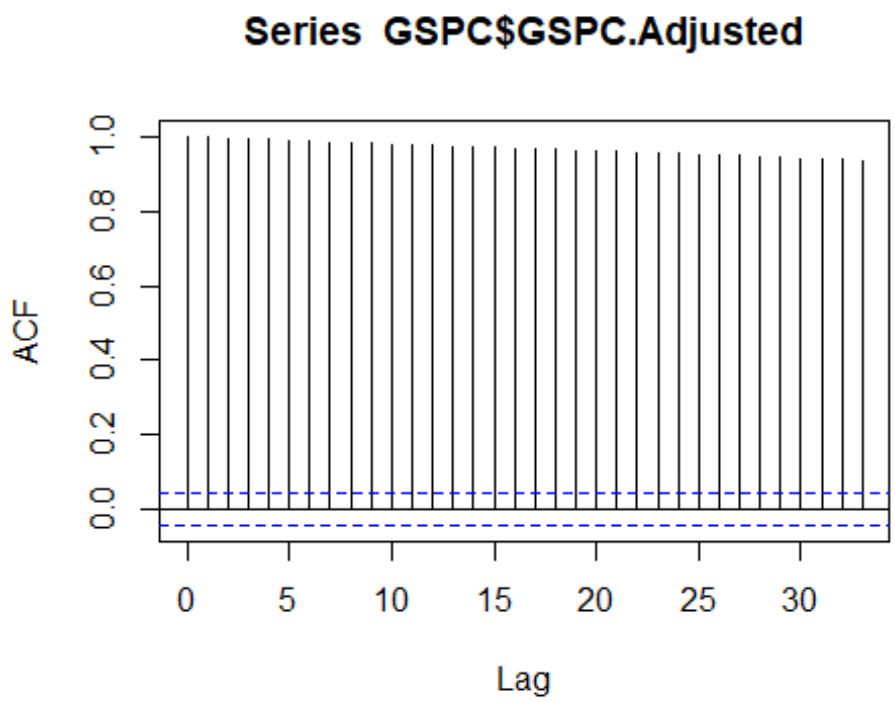
```
library("tseries")
adf.test(GSPC$GSPC.Adjusted)

##
##  Augmented Dickey-Fuller Test
##
## data:  GSPC$GSPC.Adjusted
## Dickey-Fuller = -2.6146, Lag order = 12, p-value = 0.3181
## alternative hypothesis: stationary
```

Since the p-value is large (compared to 0.05 at 95% confidence interval), the null hypothesis cannot be rejected. Thus, the time series is not stationary.

I am also curious to check if there is severe autocorrelation in this time series by using the autocorrelation function plot.

```
acf(GSPC$GSPC.Adjusted)
```

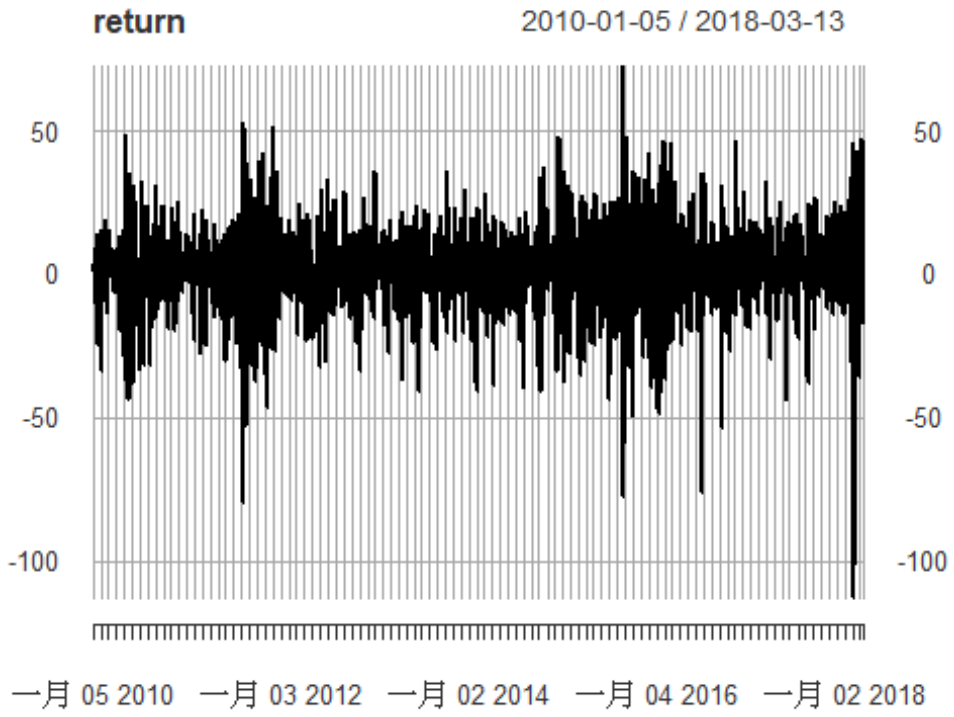


Since all peaks exceed the 95% confidence interval bands, autocorrelation is severe in this time series.

I'd like to recap the problems observed so far for the adjusted price time series. 1. Non-constant mean 2. Non-constant variance 3. Non-stationary 4. Severe autocorrelation

The first problem can be solved by applying a first order differencing. By differencing the adjusted close price time series, the time series of return is obtained. The return time series is then plotted.

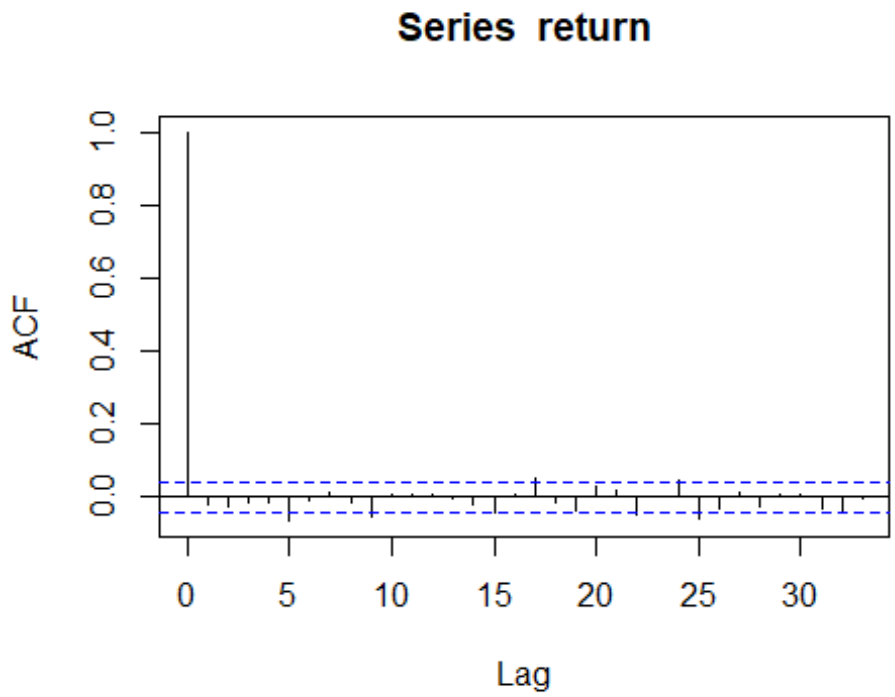
```
return <- diff(GSPC$GSPC.Adjusted)
return <- return[-1]
names(return) <- c("Return")
plot(return)
```



By inspecting this time series, it has approximately a constant mean and a constant variance.

The next step is to investigate the autocorrelation in this time series.

```
acf(return)
```



The ACF plot shows that there is no autocorrelation in the return time series at 95% confidence interval.

Returns from 1 Jan 2010 to 29 Dec 2017 will be used as the training set while returns from the first working day to the last working day in Jan 2019 will be forecasted.

```
nrow(return)

## [1] 2061
```

The total number of rows of the price data is 2059.

The return entry on 29 Dec 2017 is at row 2012.

```
return[2012,]

##                Return
## 2017-12-29 -13.92993

#Split the data into training set and testing set
ts.train <- return[1:2012,] # 80%
ts.test <- return[2013:2059,] #20%

head(ts.train)

##                 Return
## 2010-01-05   3.530030
## 2010-01-06   0.619995
```

```
## 2010-01-07   4.549926
## 2010-01-08   3.290039
## 2010-01-11   2.000000
## 2010-01-12 -10.760009
```

```
head(ts.test)
```

```
##               Return
## 2018-01-02 22.199952
## 2018-01-03 17.250000
## 2018-01-04 10.929931
## 2018-01-05 19.159912
## 2018-01-08  4.560059
## 2018-01-09  3.580078
```

Since the training set will be used for building up the ARIMA model, it is important for it to be stationary. The Augmented Dickey-Fuller test is performed again to check if the time series is stationary.

```
adf.test(ts.train)
```

```
## Warning in adf.test(ts.train): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts.train
## Dickey-Fuller = -12.983, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary
```

Since the p-value < 0.05 at 95% confidence interval, the null hypothesis is rejected and the alternative hypothesis is accepted. Thus, it is very likely to be a stationary time series.

Next, the forecast R library is loaded in order to perform the ARIMA modeling. In terms of the selection of the AR and MA orders, auto.arima() is employed to building the ARIMA model using the training set and to find out the combination of those orders which gives the smallest AIC value.

```
library("forecast")
model <- auto.arima(ts.train)
summary(model)
```

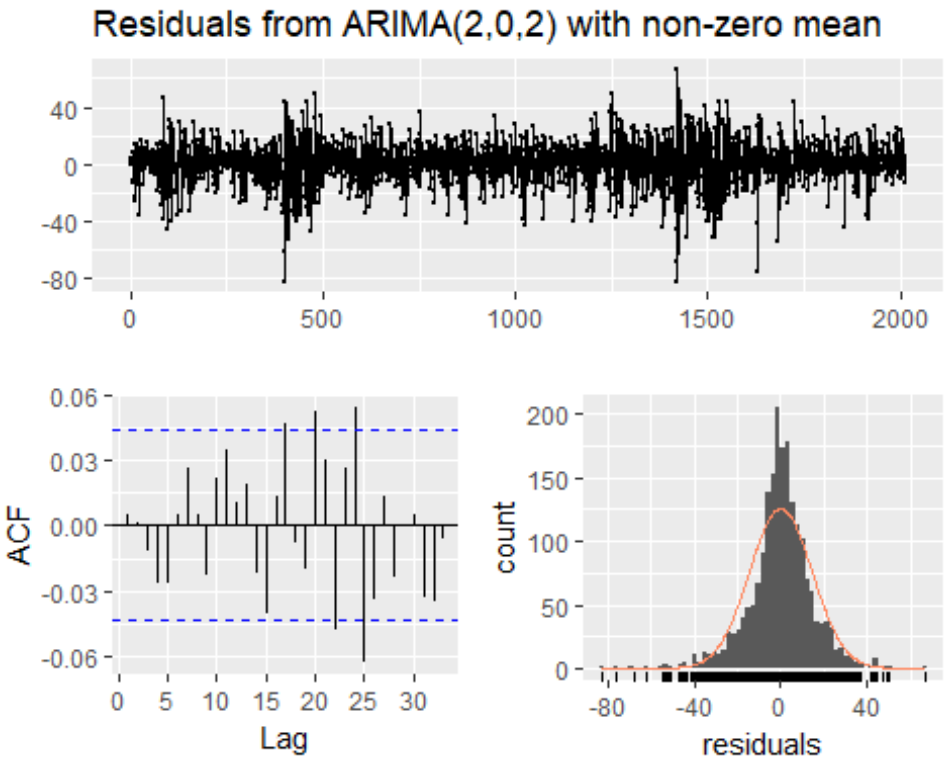
```
## Series: ts.train
## ARIMA(2,0,2) with non-zero mean
##
## Coefficients:
##          ar1     ar2      ma1      ma2    mean
##       0.0778  0.8096  -0.1234  -0.8098  0.7647
## s.e.  0.1303  0.1169   0.1361   0.1271  0.1928
##
```

```
## sigma^2 estimated as 209.8:  log likelihood=-8230.63
## AIC=16473.26   AICc=16473.3   BIC=16506.9
##
## Training set error measures:
##                       ME     RMSE      MAE  MPE MAPE      MASE
## Training set -0.009082192 14.46618 10.27125 -Inf  Inf 0.6665256
##                      ACF1
## Training set 0.005058097
```

(p,d,q) = (2,0,2) as a result of the determination of the orders

The next step is to evaluate the ARIMA fitting results and this is done by inspecting the residuals.

```
checkresiduals(model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,2) with non-zero mean
## Q* = 6.8558, df = 5, p-value = 0.2316
##
## Model df: 5.   Total lags used: 10
```

4 clusters of variance are observed. The ACF plot shows that this time series is roughly uncorrelated except at lags 17, 20, 24 and 25. The distribution plot shows

that the distribution of the residuals is not quite normally distributed. The Shapiro-Wilk normality test is then used to check if the residuals are distributed normally.

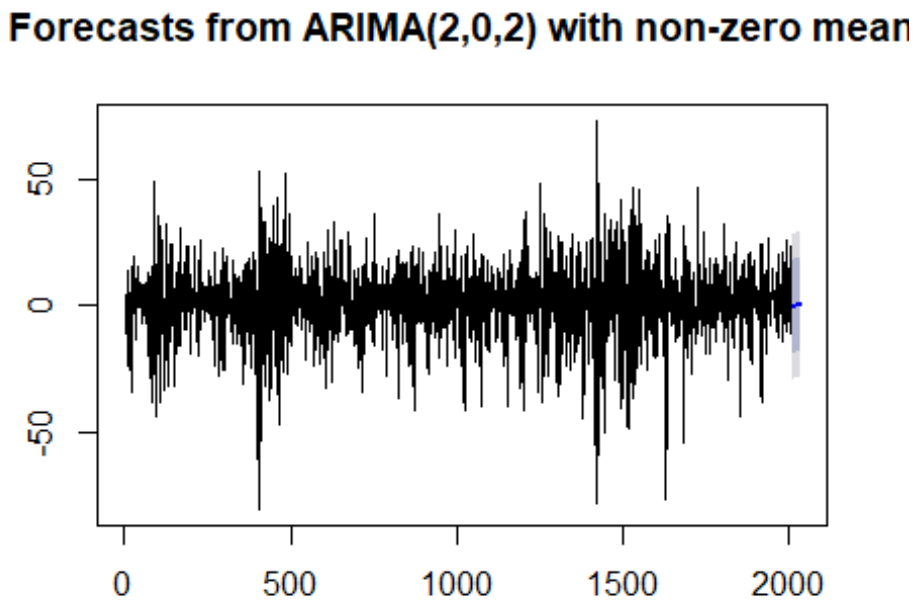
```
shapiro.test(model$residuals)

##
##  Shapiro-Wilk normality test
##
## data:  model$residuals
## W = 0.95882, p-value < 2.2e-16
```

Since the p-value is < 0.05 at 95% confidence interval, the null hypothesis is rejected and the alternative hypothesis is accepted. Thus, the residuals are not normally distributed.

Let's run a forecast model based on the existing fitted ARIMA model for 22 days (there are only 22 working days in Jan 2018).

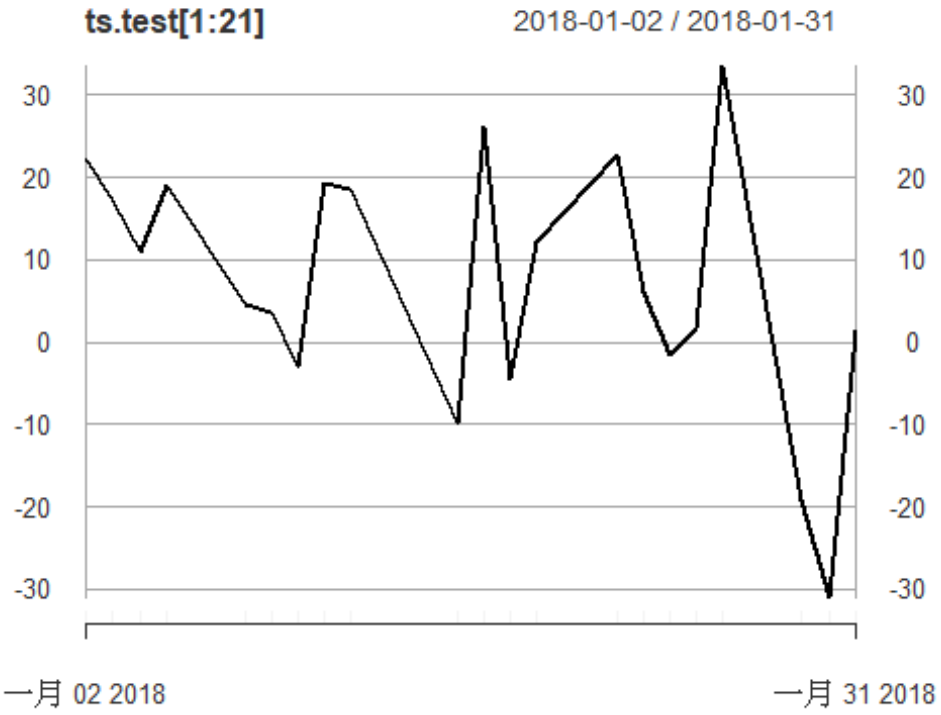
```
return.forecast <- forecast(model, h=22)
plot(return.forecast)
```



Clustered

The ARIMA model predicts a generally increasing trend for the next 22 days. Let's compare this prediction with the actual results.

```
plot(ts.test[1:21])
```



The actual time series is decreasing in Jan 2018.

Conclusion

There is a significant discrepancy between the predicted return and the actual return of S&P 500 in the same time period.

The possible sources of errors of this analysis are:

1. Information available on the market can affect the price change signifnicantly.
2. There are many factors that can affect the increase/decrease of return, which are not accounted for in this analysis.

Further Actions

In order to forecast S&P 500 in Jan 2018 accurately using given data, GARCH may be used in conjunction with ARIMA modeling.