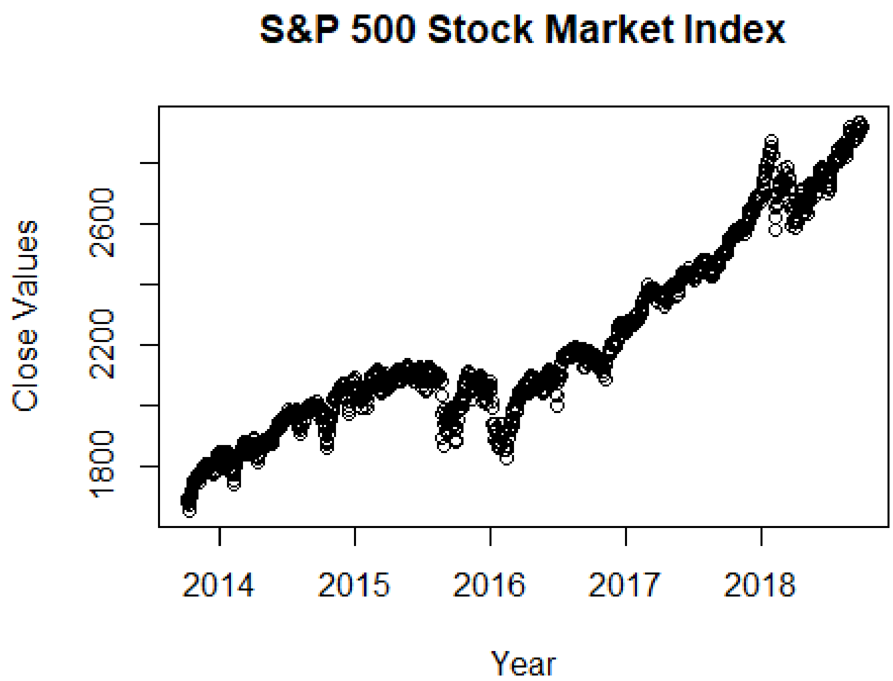
A Time-Series Analysis on the S&P 500 Stock Index

Chunmei Gao

```
library('ggplot2')

## Warning: package 'ggplot2' was built under R version 3.4.4

library('forecast')

## Warning: package 'forecast' was built under R version 3.4.4

library('tseries')

## Warning: package 'tseries' was built under R version 3.4.4

library('e1071')

## Warning: package 'e1071' was built under R version 3.4.4
```

Step 1: Load, Visualize and Examine Data

```
sp500 <- read.csv("C:/Techs/Ryerson-DataScience/CMKE136-
Capstone/data/SP500_10012013-09302018.csv", header = TRUE, stringsAsFactors =
FALSE)

head(sp500)

##         Date    Open    High     Low   Close Adj.Close     Volume
## 1 2013-10-01 1682.41 1696.55 1682.07 1695.00   1695.00 3238690000
## 2 2013-10-02 1691.90 1693.87 1680.34 1693.87   1693.87 3148600000
## 3 2013-10-03 1692.35 1692.35 1670.36 1678.66   1678.66 3279650000
## 4 2013-10-04 1678.79 1691.94 1677.33 1690.50   1690.50 2880270000
## 5 2013-10-07 1687.15 1687.15 1674.70 1676.12   1676.12 2678490000
## 6 2013-10-08 1676.22 1676.79 1655.03 1655.45   1655.45 3569230000

sp500$Date <- as.Date(sp500$Date, format="%Y-%m-%d")
attach(sp500)
plot(Date, Close, main = "S&P 500 Stock Market Index", xlab = "Year", ylab =
"Close Values")
```




```
detach(sp500)
```

Step 2: Stationarize the time series

step 2.1 Create time series object and remove any potential outliers

```
ts_close <- tsclean(ts(sp500[, c('Close')], frequency = 365.25))
```

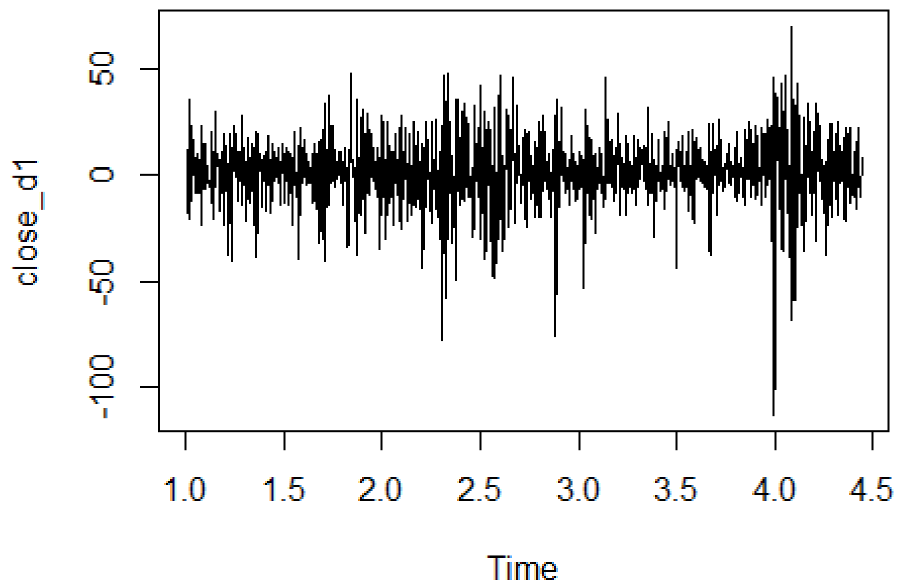
step 2.2 Stationarity check - Dicky-Fuller test

```
adf.test(ts_close, alternative = "stationary")

##
##  Augmented Dickey-Fuller Test
##
## data:  ts_close
## Dickey-Fuller = -1.6597, Lag order = 10, p-value = 0.7224
## alternative hypothesis: stationary
```

p-value > 0.5 so accept null hypothesis - non-stationary

step 2.3 Differencing series to make it stationary, d=1

```
close_d1 <- diff(ts_close, differences = 1)
plot(close_d1)
```



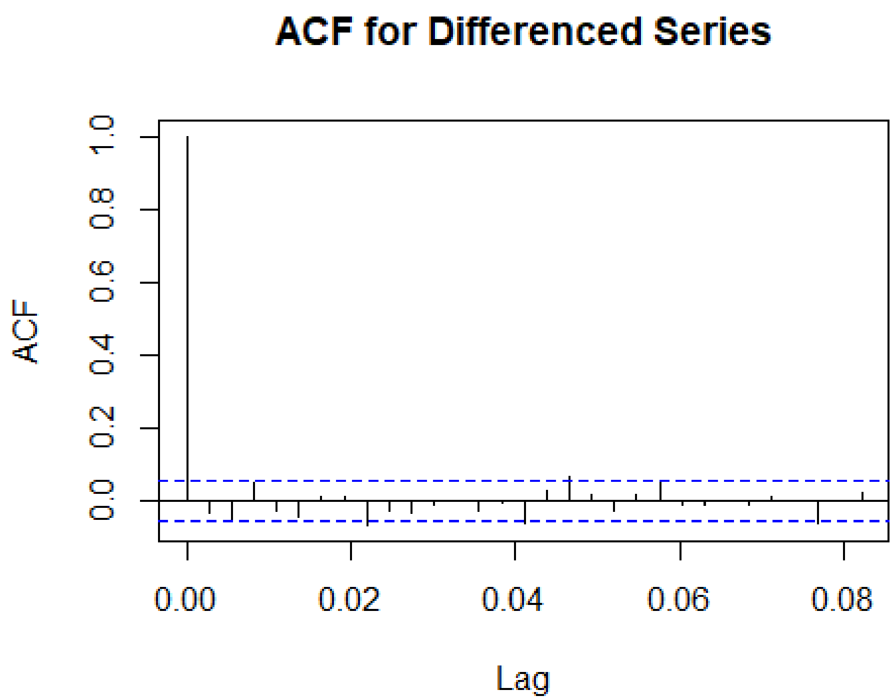

```r
adf.test(close_d1, alternative = "stationary")
```

```
## Warning in adf.test(close_d1, alternative = "stationary"): p-value smaller
## than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  close_d1
## Dickey-Fuller = -12.328, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

Step 3: Plot ACF/PACF charts and choose optimal parameters

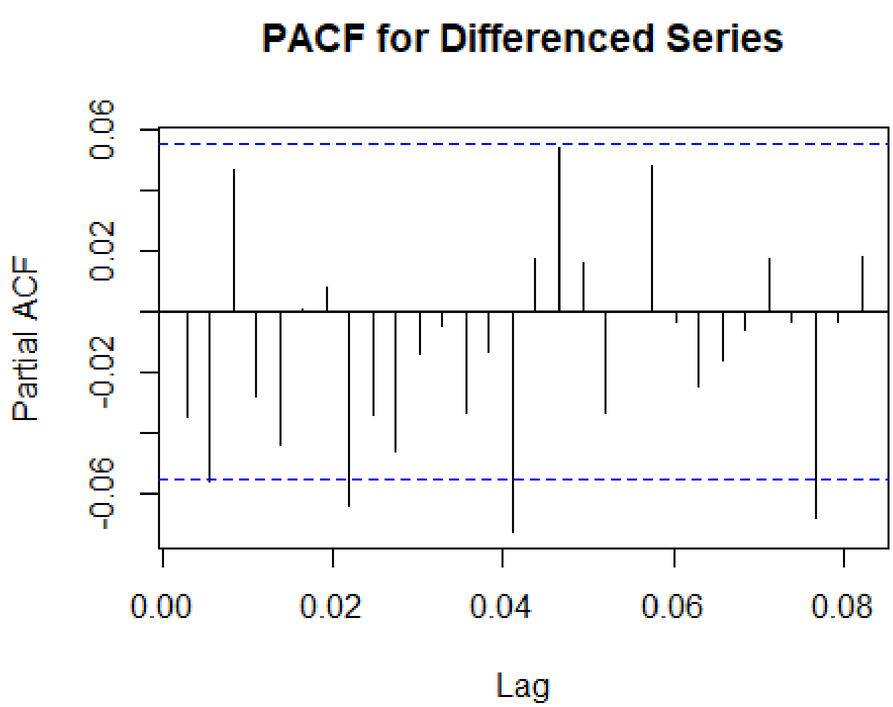
step 3.1 ACF to determine parameter q in ARIMA(p, d, q), q = 0

```r
acf(close_d1, main = "ACF for Differenced Series")
```



step 3.2 PACF to determine parameter p in ARIMA (p,d, q), p = 0

```
pacf(close_d1, main = "PACF for Differenced Series")
```



Step 4: Build and fit ARIMA model

```
arima(close_d1, order=c(0,0,0))  #ARIMA(0,1,0) aic = 10685.41

##
## Call:
## arima(x = close_d1, order = c(0, 0, 0))
##
## Coefficients:
##       intercept
##          0.9690
## s.e.     0.4698
##
## sigma^2 estimated as 277.6:  log likelihood = -5323.96,  aic = 10651.93

arima(close_d1, order=c(1,0,0))  #ARIMA(1,1,0) aic = 10686.73

##
## Call:
## arima(x = close_d1, order = c(1, 0, 0))
##
## Coefficients:
##           ar1  intercept
##       -0.0346     0.9693
## s.e.   0.0282     0.4538
##
## sigma^2 estimated as 277.3:  log likelihood = -5323.21,  aic = 10652.42

arima(close_d1, order=c(1,0,1))  #ARIMA(1,1,1) aic = 10679.06, lowest

##
## Call:
## arima(x = close_d1, order = c(1, 0, 1))
##
## Coefficients:
##          ar1      ma1  intercept
##       0.9365  -0.9678     0.9443
## s.e.  0.0276   0.0199     0.2408
##
## sigma^2 estimated as 275.4:  log likelihood = -5319.02,  aic = 10646.03

arima(close_d1, order=c(0,0,1))  #ARIMA(0,1,1) aic = 10686.63

##
## Call:
## arima(x = close_d1, order = c(0, 0, 1))
##
## Coefficients:
##           ma1  intercept
##       -0.0386     0.9690
## s.e.   0.0297     0.4513
```

```
##
## sigma^2 estimated as 277.3:  log likelihood = -5323.12,  aic = 10652.24
```

```
arima(close_d1, order=c(0,0,2))  #ARIMA(0,1,2) aic = 10683.67
```

```
##
## Call:
## arima(x = close_d1, order = c(0, 0, 2))
##
## Coefficients:
##           ma1      ma2  intercept
##       -0.0316  -0.0551     0.9692
## s.e.   0.0283   0.0292     0.4282
##
## sigma^2 estimated as 276.5:  log likelihood = -5321.36,  aic = 10650.71
```

```
arima(close_d1, order=c(2,0,0))  #ARIMA(2,1,0) aic = 10683.75
```

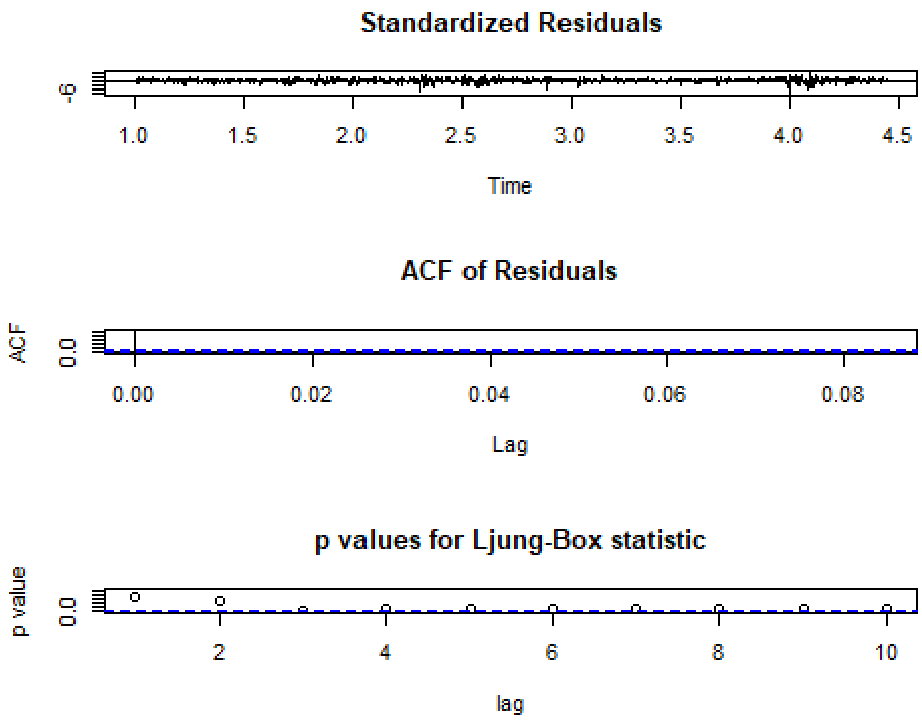
```
##
## Call:
## arima(x = close_d1, order = c(2, 0, 0))
##
## Coefficients:
##           ar1      ar2  intercept
##       -0.0365  -0.0557     0.9694
## s.e.   0.0281   0.0281     0.4292
##
## sigma^2 estimated as 276.4:  log likelihood = -5321.26,  aic = 10650.51
```

```
auto.arima(close_d1, seasonal = FALSE)
```

```
## Series: close_d1
## ARIMA(1,0,1) with non-zero mean
##
## Coefficients:
##          ar1      ma1    mean
##       0.9365  -0.9678  0.9443
## s.e.  0.0276   0.0199  0.2408
##
## sigma^2 estimated as 276.1:  log likelihood=-5319.02
## AIC=10646.03   AICc=10646.06   BIC=10666.58
```

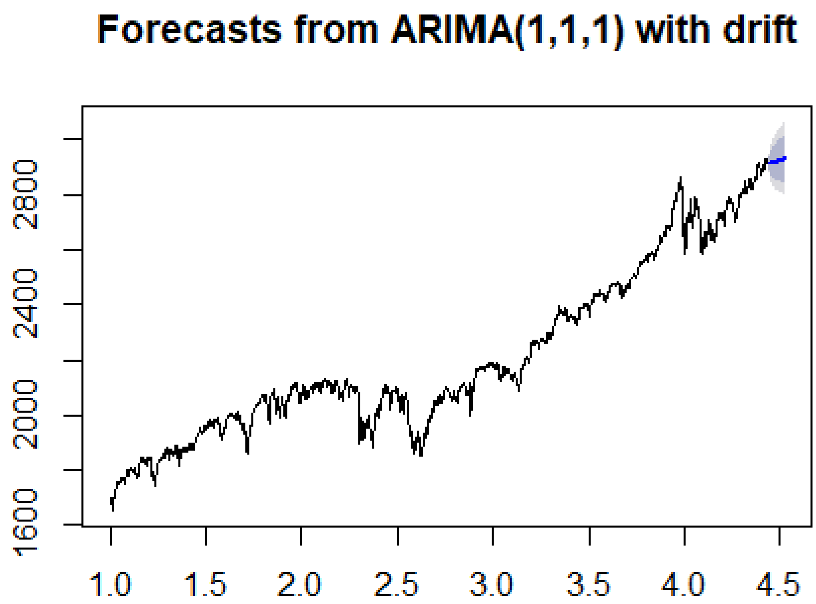
Step 5: Diagnosis the model

```
fit <- auto.arima(ts_close)
tsdiag(fit)
```



Step 6: Make Forecasts and Cross Validation

```
fcast <- forecast(fit, h=30)
plot(fcast)
```

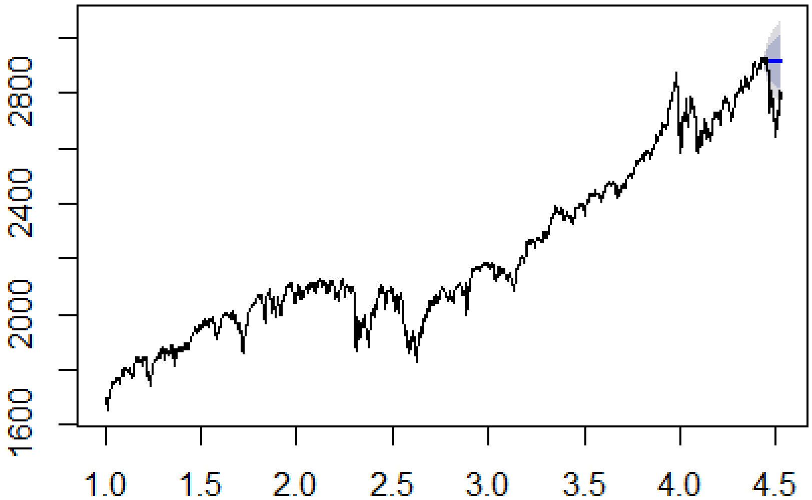


6.1 Actual 30days Index from Oct.1st, 2018

```
sp500_Oct <- read.csv("C:/Techs/Ryerson-DataScience/CMKE136-Capstone/data/SP500_10012018_11092018.csv", header = TRUE, stringsAsFactors = FALSE)
```

6.2 Calculate forecast accuracy by comparing with actual index closing values

```
accuracy(fcast, sp500_Oct$Close)

##                         ME      RMSE       MAE          MPE      MAPE
## Training set    0.02250337  16.58955  11.55024 -0.006219626 0.5332682
## Test set     -139.01892711 162.13242 141.11746 -5.089648731 5.1614038
##                   MASE        ACF1
## Training set  0.9897036 -0.01070866
## Test set     12.0919119          NA

fit_5years <- arima(ts_close, order=c(1,1,1))

fcast_Oct <- forecast(fit_5years, h=30)

ts_5years_Oct <- (ts(c(sp500$Close, sp500_Oct$Close), frequency = 365.25))

plot(fcast_Oct, main=" ")
lines(ts_5years_Oct)
```



Step 7: Compare ARIMA and SVM Models

```
days <- 1:length(sp500$Date)
df_sp500 <- data.frame(days, sp500$Close)
colnames(df_sp500) <- c("Dayth", "Close")
```

7.1 train an svm model, consider further tuning parameters for lower MSE

```
svm_mdl <- svm(Close ~ Dayth,data=df_sp500, type="eps-
regression",kernel="radial",cost=10000, gamma=10)
```

7.2 specify timesteps for forecast, for all series + 30 days ahead

```
total_days <- length(days) + 30
num_days <- 1:total_days
```

7.3 compute forecast for all the days

```
svm_fcast <- predict(svm_mdl, newdata=data.frame(Dayth=num_days))
accuracy(svm_fcast, sp500_Oct$Close)

##                 ME     RMSE      MAE      MPE     MAPE
## Test set 1060.035 1065.328 1060.035 38.04468 38.04468
```