Analysis: Sentiments of Political Leaders' Tweets vs. the SPY Price Range

Moko Sharma (Mukund Raghav Sharma)

Introduction

Once our data is successfully acquired and is in the form we want it in, our next step is to analyze it to confirm or disconfirm if there is any correlation between the sentiment of tweets of 3 famous political leaders and the price range of the SPY.

Since we are dealing with time series data, there are some nuances associated with dealing with correlations due to effects of **Seasonality** and **Trends**. Hence, our first step is to discern if the Time Series is stationary, which in a nutshell means that the seasonality and trend component are basically nullified, by the **Augmented Dickey Fuller Test** that confirms the stationarity of the time series in the alternative hypothesis if the p-value is lower than our significance level of 0.05.

If there are some non-stationary effects, we will difference the said series twice to remove them. We could have used the log-transformation but since we have a lot of negative values in both the sentiment and price range column, we would lose a lot of information in the process.

Once we have successfully got the series in a stationary form, our next task is to remove Autocorrelation that can be tested using the **Durbin-Watson Test**. If we detect autocorrelation, it can be fixed by differencing the two series by 1.

After this, we will plot the **Cross-Correlation** between the Sentiment Score and SPY Price Range time series, a prerequisite of which is that the time series in question has to be stationary. And eventually look at the **Pearson's product-moment Correlation** test after all the transformations to truly confirm if there is any correlation between the two time series at hand.

Loading the Appropriate Libraries

We start off by loading in the appropraite libraries used for the analysis.

```r
library( tseries )
library( lmtest )
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

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

```r
library( ggplot2 )
```

Loading in the Data for Trump, Pence and Ryan

Next, we take the data produced by the Python portion of this project and loading the csvs as dataframes.

```
generateFileName <- function( fileName ) {
  dataFolder  <- './data/'
  paste( dataFolder, fileName, sep = '')
}

trumpDf     <- read.csv( generateFileName( 'realDonaldTrump.csv' ))
mikePenceDf <- read.csv( generateFileName( 'VP.csv' ))
paulRyanDf  <- read.csv( generateFileName( 'SpeakerRyan.csv' ))
```

Trump's Dataframe: Answering if Trump's Tweets' Sentiment is Correlated with the SPY Price Ranges

Let's start off by doing a smoke test and take a look at the first few rows of the Trump dataframe.

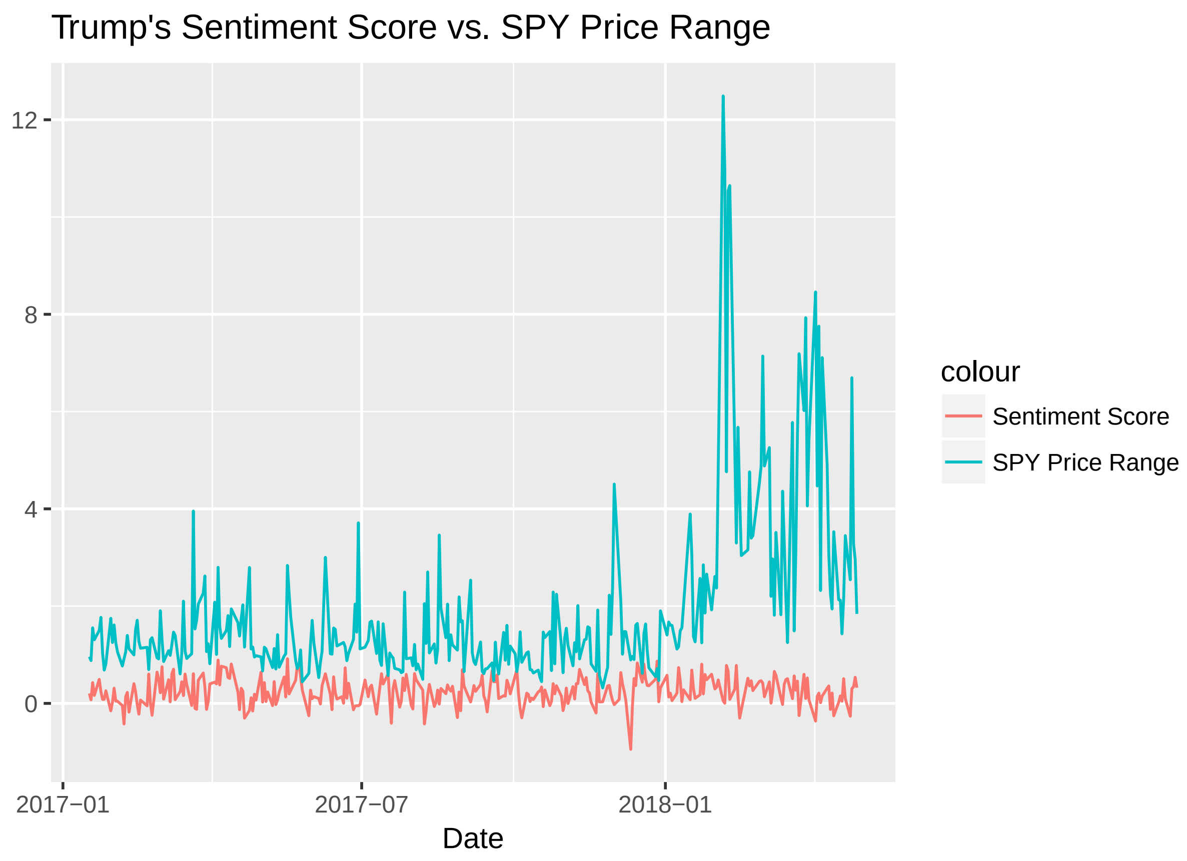
```
head( trumpDf )
```

```
##         date   open     high    low  close    volume Name  range
## 1 2018-04-27 267.00 267.3400 265.50 266.56  57053647  SPY 1.8400
## 2 2018-04-26 264.79 267.2452 264.29 266.31  67731942  SPY 2.9552
## 3 2018-04-25 262.91 264.1300 260.85 263.63 103756753  SPY 3.2800
## 4 2018-04-24 267.73 267.9762 261.28 262.98 112885452  SPY 6.6962
## 5 2018-04-23 267.26 267.8900 265.35 266.57  65557954  SPY 2.5400
## 6 2018-04-20 268.81 269.0600 265.61 266.61  99953133  SPY 3.4500
##   realDonaldTrump.Vader.Sentiment.Score
## 1                             0.3223364
## 2                             0.5372000
## 3                             0.3394000
## 4                             0.3054125
## 5                            -0.2659000
## 6                             0.0996000
```

Next, let's plot the Timeseries of the Sentiment Score and SPY Values for Trump's Tweets' Sentiment.

```
trumpPlot <-
  ggplot( data = trumpDf,  aes( x = as.Date( trumpDf$date ))) +
    geom_line( aes(  y     = trumpDf$realDonaldTrump.Vader.Sentiment.Score,
                     color = 'Sentiment Score' )) +
    geom_line( aes( y = trumpDf$range, color = 'SPY Price Range' )) +
    scale_x_date( 'Date' ) + ylab( ' ' ) +
    ggtitle( "Trump's Sentiment Score vs. SPY Price Range" )

trumpPlot
```

```
## Warning in strptime(xx, f <- "%Y-%m-%d", tz = "GMT"): unknown timezone
## 'default/America/New_York'
```



At first glance, the highs of the SPY Price Range don't necessarily match the highs of the Sentiment Score and vice versa about lows.

Extracting Time Series From the Dataframe

Once we have plotted the raw time series via the dataframe, we will simply extract out the time series portion to make our analysis a bit simplier. We aren't to worry too much about the start and end time since we have confirmed from our python analysis that both the sentiment and spy range series have the same cardinality.

```
trumpTs.Sentiment <-
  ts( data  = trumpDf$realDonaldTrump.Vader.Sentiment.Score )

trumpTs.SPY <-
  ts( data = trumpDf$range )
```

Test for Stationarity

As mentioned before, before diving into creating cross-correlations, we want to test for stationarity of the two series. We do this for both the Sentiment and SPY Price Range series by

Sentiment Time Series

```
adf.test( trumpTs.Sentiment )
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  trumpTs.Sentiment
## Dickey-Fuller = -5.9023, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

Seems like since the p-value of the Sentiment Analysis is 0.01, we accept the Alternative Hypothesis that the time series is stationary with a significance level of 0.05.

SPY Time Series

```
adf.test( trumpTs.SPY )
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  trumpTs.SPY
## Dickey-Fuller = -4.1837, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

Since the p-value is less than our significance level of 0.05, we accept the Alternative Hypothesis i.e. the time series is stationary.

Great - no de-trending and de-seasonality needed as we accept the alternative hypothesis in both cases. Next, let's test for **Auto-Correlation**.

Test for Autocorrelation

Now, let's check for Autocorrelation. We do this by using the **Durbin-Watson Test**.

```
trump.model <- trumpTs.Sentiment ~ trumpTs.SPY
dwtest( trump.model )
```

```
##
##  Durbin-Watson test
##
## data:  trump.model
## DW = 1.6951, p-value = 0.002748
## alternative hypothesis: true autocorrelation is greater than 0
```

The p-value of 0.0101 means that we accept the alternative hypothesis i.e. there is a true autocorrelation greater than 0 implying there is autocorrelation. In an effort to reduce the autocorrelation, let's difference the time series by 1.

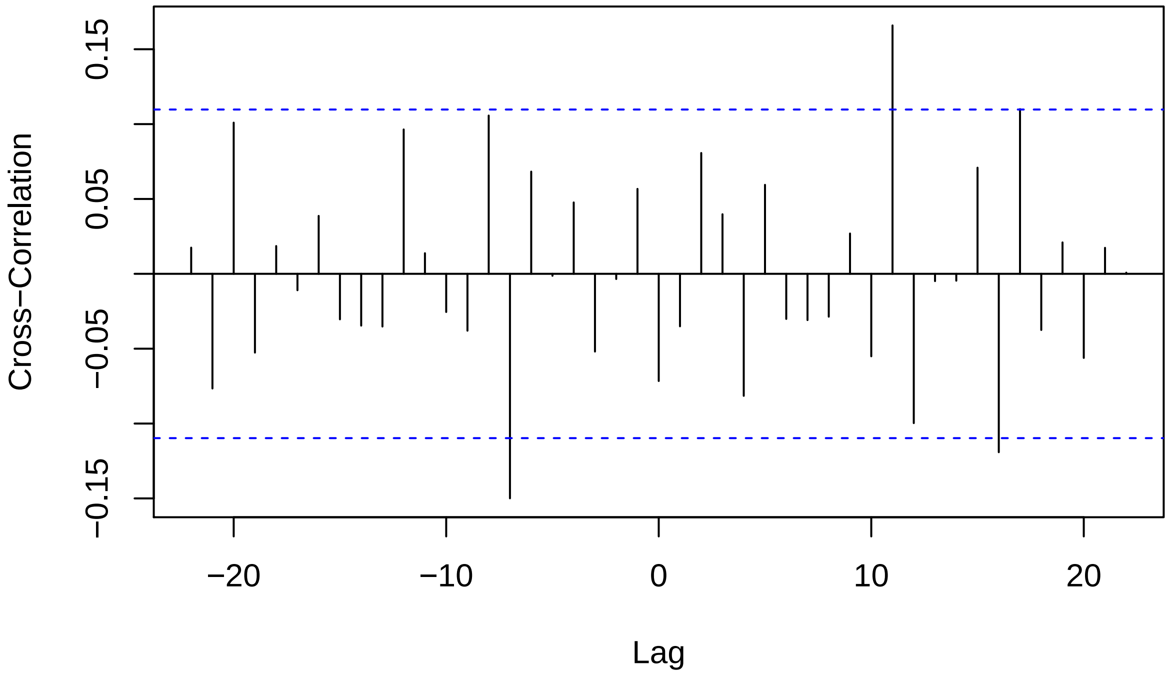
```
trumpTs.Sentiment.1diff <-
  diff( trumpTs.Sentiment, differences = 1 )

trumpTs.SPY.1diff <-
  diff( trumpTs.SPY, differences = 1 )

dwtest( trumpTs.Sentiment.1diff ~ trumpTs.SPY.1diff )
```

```
##
##  Durbin-Watson test
```

```
##
## data:  trumpTs.Sentiment.1diff ~ trumpTs.SPY.1diff
## DW = 2.8474, p-value = 1
## alternative hypothesis: true autocorrelation is greater than 0
```

We have now successfully fixed the autocorrelation issue since the p-value = 1 and hence we cannot reject the null hypothesis. We are at a point where we can take a look at the **Cross-Correlation** Plot.

```
trump.ccf <- ccf( y    = trumpTs.Sentiment.1diff,
                  x    = trumpTs.SPY.1diff,
                  ylab = "Cross-Correlation",
                  xlab = "Lag",
                  main = " " )
```



```
trump.ccf
```

```
##
## Autocorrelations of series 'X', by lag
##
##    -22    -21    -20    -19    -18    -17    -16    -15    -14    -13
##  0.018 -0.077  0.101 -0.053  0.019 -0.011  0.039 -0.030 -0.035 -0.035
##    -12    -11    -10     -9     -8     -7     -6     -5     -4     -3
##  0.096  0.014 -0.026 -0.038  0.106 -0.150  0.068 -0.001  0.048 -0.052
##     -2     -1      0      1      2      3      4      5      6      7
## -0.003  0.057 -0.072 -0.035  0.081  0.040 -0.081  0.059 -0.030 -0.031
##      8      9     10     11     12     13     14     15     16     17
## -0.029  0.027 -0.055  0.166 -0.100 -0.005 -0.005  0.071 -0.119  0.110
##     18     19     20     21     22
## -0.037  0.021 -0.056  0.017  0.001
```

We notice that there is high correlation on **just** the -13th, 11th and 16th lag. Next, let's conduct the Pearson's Moment Correlation test on the stationary and non-autocorrelated data.

```r
cor.test( x = trumpTs.SPY.1diff,
          y = trumpTs.Sentiment.1diff )
```

```
##
##  Pearson's product-moment correlation
##
## data:  trumpTs.SPY.1diff and trumpTs.Sentiment.1diff
## t = -1.2778, df = 317, p-value = 0.2022
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.17998310  0.03852887
## sample estimates:
##         cor
## -0.07158597
```

This p-value of 0.1901 >> 0.05 highlights that the two series **aren't** correlated and we evidently reject our alternative hypothesis that the true correlation is not equal to 0.

Generalizing the Pipeline

Now that we have successfully completed the test for correlation for Trump, let's generalize the pipeline to make it easier to do the same for Pence and Ryan.

```r
correlationTest <- function( sentimentColumn, spyColumn, alpha = 0.05 ) {

  ##############################################
  # Converting the data into Time Series Data #
  ##############################################
  sentiment.ts <- ts( sentimentColumn )
  spy.ts       <- ts( spyColumn )

  result.original.sentiment <- sentiment.ts
  result.original.spy       <- spy.ts

  #########################
  # Test for Stationarity #
  #########################
  # Conduct ADF Test for the Sentiment.
  sentiment.adf.test <- adf.test( sentiment.ts )

  if ( sentiment.adf.test$p.value > alpha ) {
    sentiment.ts <-
      diff( sentiment.ts, differences = 2 )
  }

  # Conduct ADF Test for the SPY Price Range.
  spy.adf.test <- adf.test( spy.ts )

  if ( spy.adf.test$p.value > alpha ) {
    spy.ts <-
      diff( spy.ts, differences = 2 )
  }

  ############################
  # Test for Autocorrelation #
  ##############################
  # Conduct DW Test
  model      <- sentiment.ts ~ spy.ts
  ts.dw.test <- dwtest( model )

  # Alternative Hypothesis for DW Test => There is Auto-correlation
  if ( ts.dw.test$p.value < alpha ) {

    # Fix auto-correlation by diffing series 1 time.
    sentiment.ts <-
      diff( sentiment.ts, differences = 1 )
    spy.ts <-
      diff( spy.ts, differences = 1 )
  }

  result.final.sentiment <- sentiment.ts
  result.final.spy       <- spy.ts

  ########################
  # Test for Correlation #
  ########################
  # Conduct the Correlation Test
  result.cor <-
    cor.test( y = sentiment.ts,
              x = spy.ts )

  result.pval <- result.cor$p.value

  list( originalSentiment = result.original.sentiment,
        originalSPY       = result.original.spy,
        finalSentiment    = result.final.sentiment,
        finalSPY          = result.final.spy,
        corr              = result.cor )
}
```

Pence's Dataframe: Answering if Pence's Tweets' Sentiment is Correlated with the SPY Price Ranges

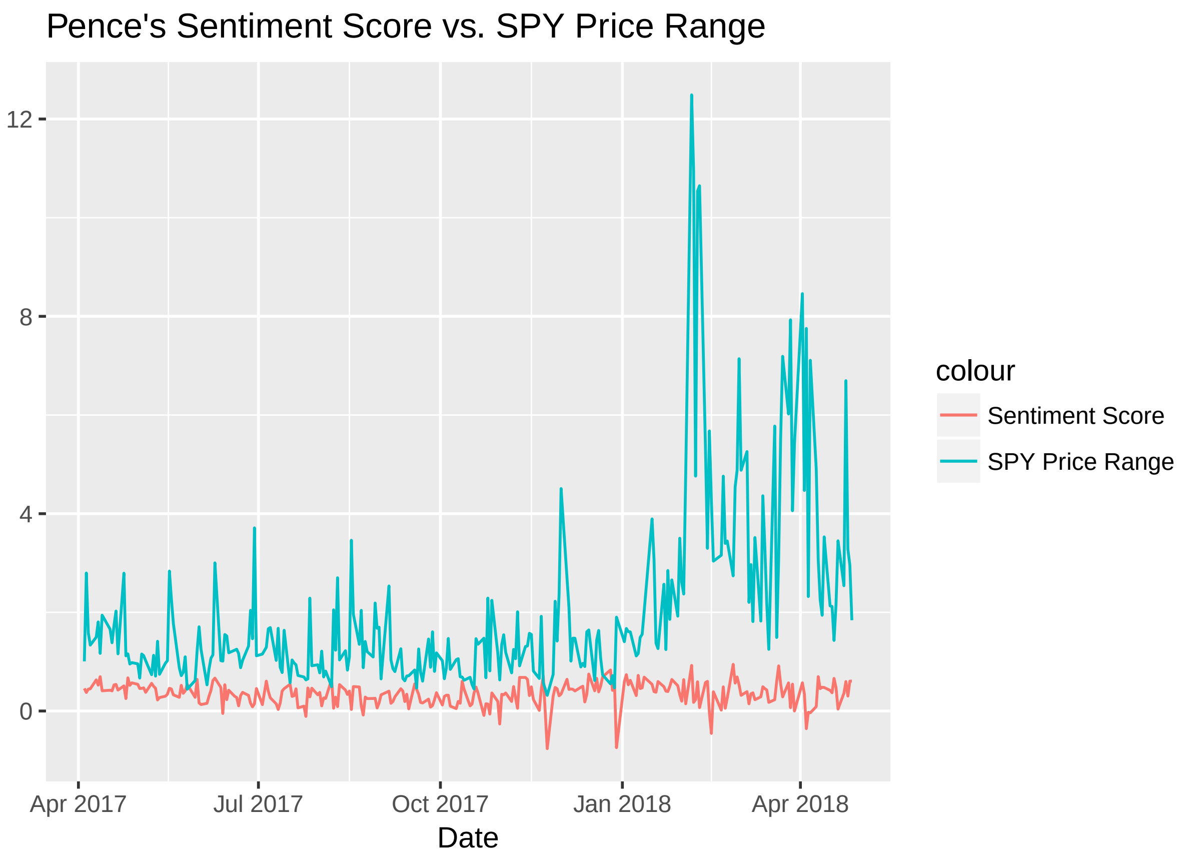
Let's do a smoke test and print out the first few rows of the dataframe.

```
head( mikePenceDf )
```

```
##         date   open     high    low  close    volume Name  range
## 1 2018-04-27 267.00 267.3400 265.50 266.56  57053647  SPY 1.8400
## 2 2018-04-26 264.79 267.2452 264.29 266.31  67731942  SPY 2.9552
## 3 2018-04-25 262.91 264.1300 260.85 263.63 103756753  SPY 3.2800
## 4 2018-04-24 267.73 267.9762 261.28 262.98 112885452  SPY 6.6962
## 5 2018-04-23 267.26 267.8900 265.35 266.57  65557954  SPY 2.5400
## 6 2018-04-20 268.81 269.0600 265.61 266.61  99953133  SPY 3.4500
##   VP.Vader.Sentiment.Score
## 1               0.60760000
## 2               0.60566923
## 3                  0.30245556
## 4                  0.59701429
## 5                  0.36136667
## 6                  0.03916364
```

Next, like before, let's do a line plot of the Sentiment Score vs. Price Range.

```
pencePlot <-
  ggplot( data = mikePenceDf,  aes( x = as.Date( mikePenceDf$date ))) +
    geom_line( aes(  y      = mikePenceDf$VP.Vader.Sentiment.Score,
                     color = 'Sentiment Score' )) +
    geom_line( aes( y = mikePenceDf$range, color = 'SPY Price Range' )) +
    scale_x_date( 'Date' ) + ylab( ' ' ) +
    ggtitle( "Pence's Sentiment Score vs. SPY Price Range" )

pencePlot
```

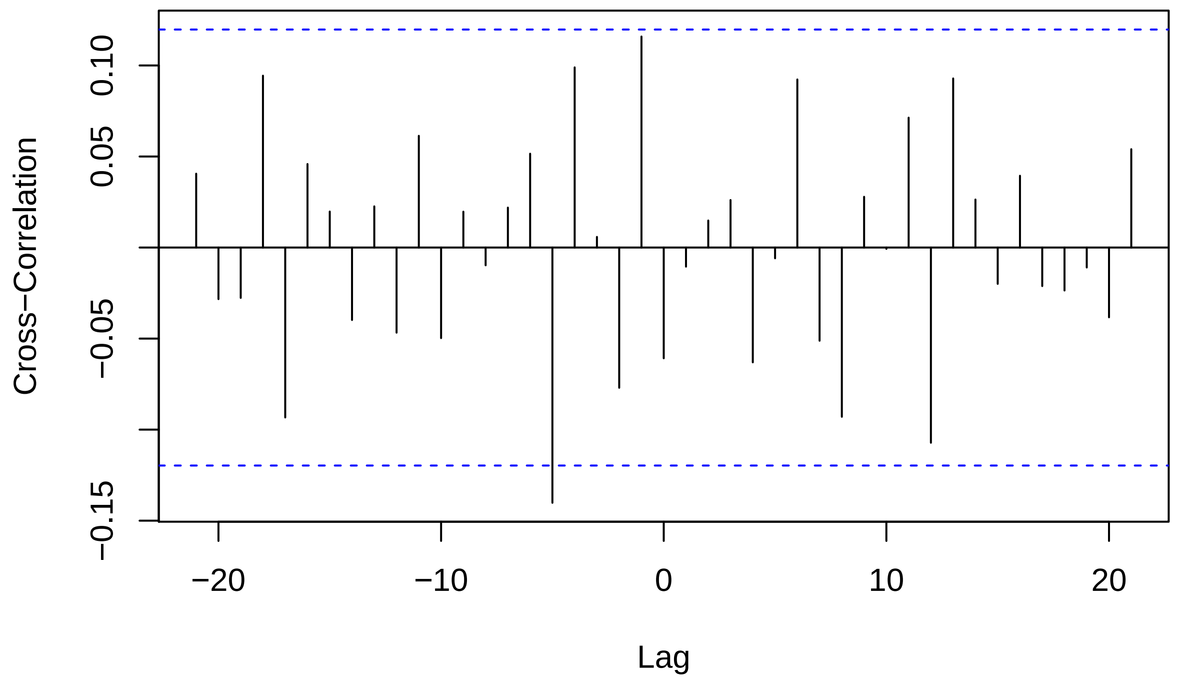


Next, let's run the generalized pipeline of the correlation for Pence's tweets.

```
penceResult <-
  correlationTest( mikePenceDf$VP.Vader.Sentiment.Score,
                   mikePenceDf$range )
```

Cross-Correlation Plot

```
pence.ccf <- ccf( y    = penceResult$finalSentiment,
                  x    = penceResult$finalSPY,
                  ylab = "Cross-Correlation",
                  xlab = "Lag",
                  main = " " )
```



```
penceResult$corr
```

```
##
##  Pearson's product-moment correlation
##
## data:  spy.ts and sentiment.ts
## t = -0.99373, df = 266, p-value = 0.3213
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.17933086  0.05943789
## sample estimates:
##         cor
## -0.06081645
```

The p-value is >> 0.05, our significance level implying that we fail to reject our null hypothesis i.e. the true correlation is close to 0. In other words, Mike Pence's Tweets' Sentiment and SPY Price Range are not correlated.

Paul Ryan's Dataframe: Answering if Paul Ryan's Tweets' Sentiment is Correlated with the SPY Price Ranges

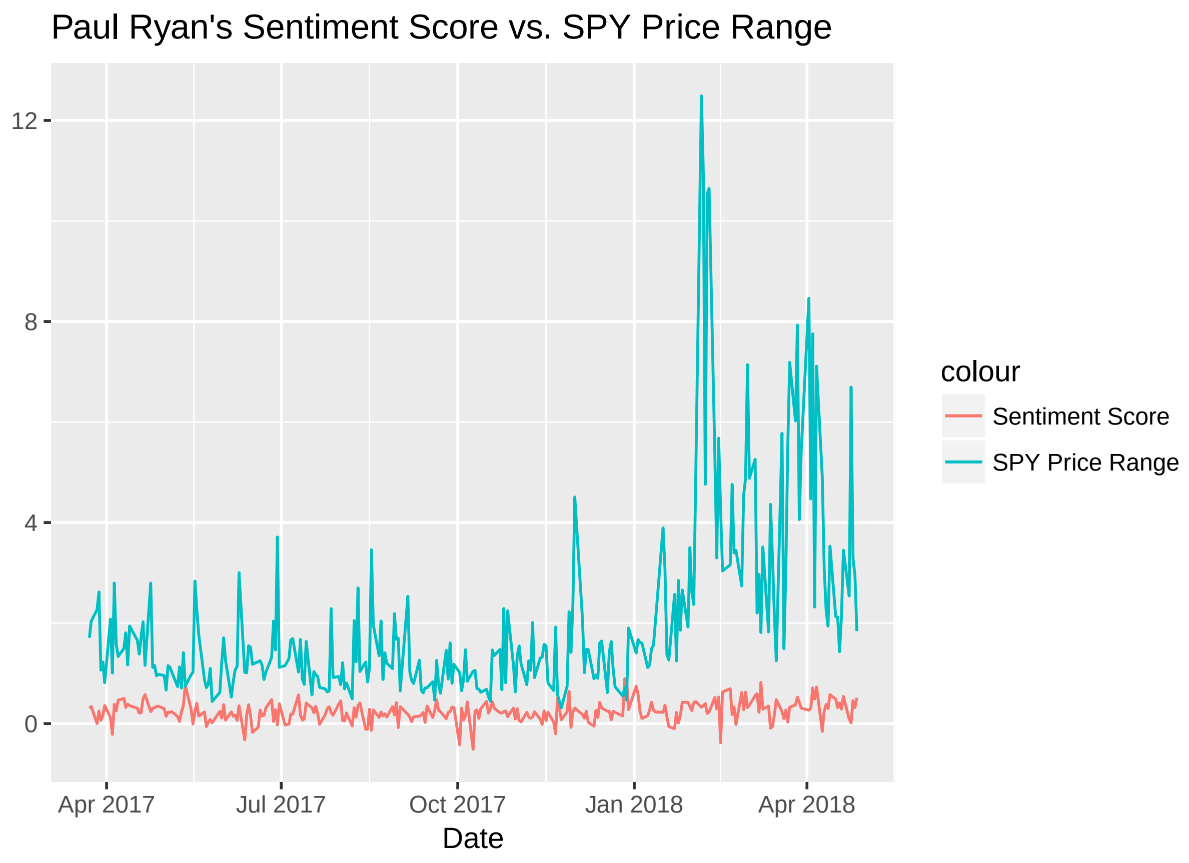
Like before, let's do a smoke test of the dataframe.

```
head( paulRyanDf )
```

```
##         date   open     high    low  close    volume Name  range
## 1 2018-04-27 267.00 267.3400 265.50 266.56  57053647  SPY 1.8400
## 2 2018-04-26 264.79 267.2452 264.29 266.31  67731942  SPY 2.9552
## 3 2018-04-25 262.91 264.1300 260.85 263.63 103756753  SPY 3.2800
## 4 2018-04-24 267.73 267.9762 261.28 262.98 112885452  SPY 6.6962
## 5 2018-04-23 267.26 267.8900 265.35 266.57  65557954  SPY 2.5400
## 6 2018-04-20 268.81 269.0600 265.61 266.61  99953133  SPY 3.4500
##   SpeakerRyan.Vader.Sentiment.Score
## 1                        0.51633333
## 2                        0.31692857
## 3                        0.46001429
## 4                        0.01466667
## 5                        0.09050000
## 6                        0.54230000
```

Let's do a line plot of the Sentiment Score vs. Price Range.

```
paulRyanPlot <-
  ggplot( data = paulRyanDf,  aes( x = as.Date( paulRyanDf$date ))) +
    geom_line( aes(  y     = paulRyanDf$SpeakerRyan.Vader.Sentiment.Score,
                     color = 'Sentiment Score' )) +
    geom_line( aes( y = paulRyanDf$range, color = 'SPY Price Range' )) +
    scale_x_date( 'Date' ) + ylab( ' ' ) +
    ggtitle( "Paul Ryan's Sentiment Score vs. SPY Price Range" )

paulRyanPlot
```



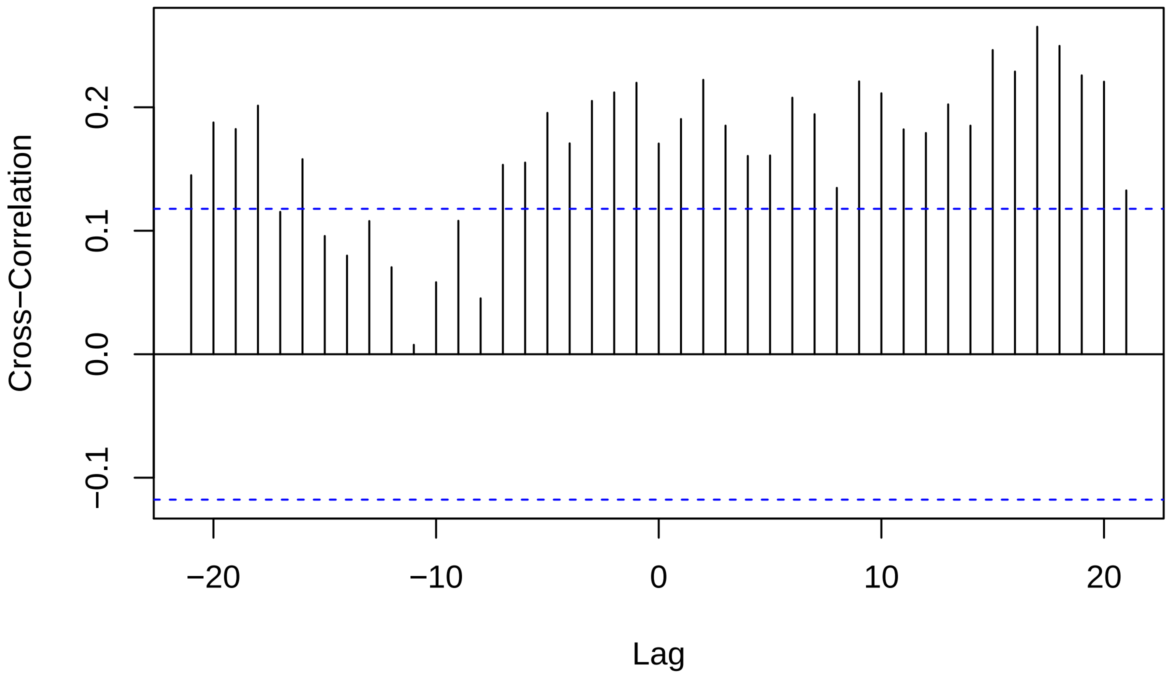
Next, let's run the generalized pipeline of the correlation for Paul Ryan's tweets.

```
ryanResult <-
  correlationTest( paulRyanDf$SpeakerRyan.Vader.Sentiment.Score,
                    paulRyanDf$range )
```

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

Correlation Plot

```
paulRyan.ccf <- ccf( y    = ryanResult$originalSentiment,
                     x    = ryanResult$originalSPY,
                     ylab = "Cross-Correlation",
                     xlab = "Lag",
                     main = " " )
```



Paul Ryan's cross-correlation is extremely indicative of a strong correlation. Let's finally look at the Pearson's product-moment correlation test.

```
ryanResult$corr
```

```
##
##  Pearson's product-moment correlation
##
## data:  spy.ts and sentiment.ts
## t = 2.8719, df = 275, p-value = 0.004397
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.05387259 0.28281237
## sample estimates:
##       cor
## 0.1706446
```

A p-value that low and the cross-correlation plot looking extremely positive highlights the fact that Paul Ryan's tweets are the most correlated out of the 3 accounts we considered. We hence, reject the null hypothesis and accept the alternative hypothesis that the true correlation is not equal to 0.

Conclusion

We looked at the Twitter Sentiment Score of Donald Trump, Mike Pence and Paul Ryan and discerned the highest correlation of the Sentiment Score vs. the SPY price range to be Paul Ryan's.

During our analysis we encountered and handled nuances of time series data such as Autocorrelation and Non-Stationarity via differencing.