Group Members

- Yonas Menghis Berhe (yonix500@gmail.com (mailto:yonix500@gmail.com))
- Micheal Lucky (smgmol56@gmail.com (mailto:smgmol56@gmail.com))
- Boluwatife Adeyeye (adeyeyebolu027@gmail.com (mailto:adeyeyebolu027@gmail.com))
- Muhammed Jamiu Saka (sakasim_jay@yahoo.com (mailto:sakasim_jay@yahoo.com))
- Sola-Aremu Oluwapelumi (solaaremu.pelumi@gmail.com (mailto:solaaremu.pelumi@gmail.com))

Pairs Algorithimic Trading strategy

Pairs trading is a popular investment trading strategy dated back to the 1980s and has been adopted by organizations since then. It was created when the first quants in wall street were looking for principles based on statistics which can be used to take advantage of short-run differences in the prices of two assets with similar characteristics which have had consistent long-run equilibrium overtime. Basically, it works by matching an asset having a long position with another asset having a short position bearing in mind that both assets have high correlation. When this principle of correlation is in place over a period of time and a correlation discrepancy is noticed, the pairs trade can be deployed. When this discrepancy comes into play, the pairs trader would purchase the asset matched in the long position when it underperforms and then sell the asset matched in the short position when it outperforms. The trader then makes his profit afterwards when the prices converge. Therefore, pairs trading can be seen as both a hedging and speculation instrument nevertheless, it has its own limitations.

Steps taken in Pairs Trading are as follows

Loading necessary packages

```
library(Quandl)
```

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

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

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

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

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

```
library(quantmod)
```

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

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

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

```r
library(timeSeries)
```

```
## Loading required package: timeDate
```

```
##
## Attaching package: 'timeSeries'
```

```
## The following object is masked from 'package:zoo':
##
##     time<-
```

```r
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
library(ggplot2)
library(urca)
library(PerformanceAnalytics)
```

```
##
## Attaching package: 'PerformanceAnalytics'
```

```
## The following objects are masked from 'package:timeDate':
##
##     kurtosis, skewness
```

```
## The following object is masked from 'package:graphics':
##
##     legend
```

```r
library(tseries)
#install.packages("doParallel")
library(doParallel)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
library(foreach)
library(urca)
```

Johnsen cointegration test of selected stocks

```
 jtest <- function(t1, t2) {
  start <- st_date
  getSymbols(t1, from = start)
  getSymbols(t2, from = start)
  j <- summary(ca.jo(cbind(get(t1)[, 6], get(t2)[, 6])))
  r <- data.frame(stock1 = t1, stock2 = t2, stat = j@teststat[2])
  r[, c("pct10", "pct5", "pct1")] <- j@cval[2, ]
  return(r)
}

pair <- function(lst) {
  d2 <- data.frame(t(combn(lst, 2)))
  stat <- foreach(i = 1:nrow(d2), .combine = rbind) %dopar% jtest(as.character(d2[i, 1
]), as.character(d2[i, 2]))
  stat <- stat[order(-stat$stat), ]
  rownames(stat) <- NULL
  return(stat)
}

st_date <- "2010-01-01"
tickers <- c('AAPL', 'ADBE', 'COKE', 'CSCO', 'GOOG', 'IBM', 'INTC', 'MSFT', 'NFLX','PEP'
, 'SPY', 'TSCO')
pair(tickers)
```

```
## Warning: executing %dopar% sequentially: no parallel backend registered
```

```
## '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 setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
##    stock1 stock2      stat pct10 pct5  pct1
## 1    ADBE   INTC 23.483689 12.91 14.9 19.19
## 2    MSFT    SPY 22.934053 12.91 14.9 19.19
## 3    INTC   MSFT 18.452126 12.91 14.9 19.19
## 4    INTC   NFLX 17.420863 12.91 14.9 19.19
## 5    GOOG    SPY 17.361447 12.91 14.9 19.19
## 6    AAPL   INTC 16.363705 12.91 14.9 19.19
## 7    ADBE   NFLX 16.039275 12.91 14.9 19.19
## 8     IBM   MSFT 15.979108 12.91 14.9 19.19
## 9    GOOG    PEP 15.649095 12.91 14.9 19.19
## 10   GOOG   INTC 14.205750 12.91 14.9 19.19
## 11   COKE    PEP 14.121594 12.91 14.9 19.19
## 12   GOOG   MSFT 13.777274 12.91 14.9 19.19
## 13   CSCO   MSFT 13.591981 12.91 14.9 19.19
## 14   ADBE    IBM 13.545604 12.91 14.9 19.19
## 15   AAPL    IBM 13.083739 12.91 14.9 19.19
## 16   COKE   MSFT 12.583236 12.91 14.9 19.19
## 17    PEP    SPY 11.607652 12.91 14.9 19.19
## 18    IBM    SPY 11.016049 12.91 14.9 19.19
## 19   GOOG    IBM 10.953049 12.91 14.9 19.19
## 20   COKE   GOOG 10.948590 12.91 14.9 19.19
## 21   ADBE    SPY 10.830013 12.91 14.9 19.19
## 22   ADBE   MSFT 10.537898 12.91 14.9 19.19
## 23   ADBE   CSCO 10.491588 12.91 14.9 19.19
## 24    IBM   TSCO 10.485630 12.91 14.9 19.19
## 25   CSCO    IBM 10.423285 12.91 14.9 19.19
## 26   ADBE   GOOG 10.291306 12.91 14.9 19.19
## 27    IBM    PEP 10.247091 12.91 14.9 19.19
## 28   COKE   CSCO 10.182796 12.91 14.9 19.19
## 29   CSCO   NFLX 10.179755 12.91 14.9 19.19
## 30   AAPL   MSFT 10.178455 12.91 14.9 19.19
## 31   AAPL   ADBE 10.139951 12.91 14.9 19.19
## 32   MSFT   NFLX 10.125836 12.91 14.9 19.19
## 33   INTC    PEP  9.923093 12.91 14.9 19.19
## 34    IBM   NFLX  9.916114 12.91 14.9 19.19
## 35    IBM   INTC  9.905348 12.91 14.9 19.19
## 36   COKE    IBM  9.673231 12.91 14.9 19.19
## 37   MSFT   TSCO  9.213765 12.91 14.9 19.19
## 38   MSFT    PEP  9.187188 12.91 14.9 19.19
## 39   ADBE   COKE  8.800667 12.91 14.9 19.19
## 40   COKE    SPY  8.343314 12.91 14.9 19.19
## 41   CSCO   GOOG  7.947911 12.91 14.9 19.19
## 42   ADBE    PEP  7.641108 12.91 14.9 19.19
## 43    PEP   TSCO  7.557714 12.91 14.9 19.19
## 44   CSCO   INTC  7.356980 12.91 14.9 19.19
## 45   AAPL   NFLX  7.202607 12.91 14.9 19.19
## 46   COKE   INTC  7.197185 12.91 14.9 19.19
## 47   GOOG   NFLX  7.120733 12.91 14.9 19.19
## 48   CSCO    PEP  7.016129 12.91 14.9 19.19
## 49   INTC    SPY  6.918962 12.91 14.9 19.19
## 50   CSCO    SPY  6.913125 12.91 14.9 19.19
## 51   COKE   TSCO  6.863924 12.91 14.9 19.19
## 52   COKE   NFLX  6.816128 12.91 14.9 19.19
```

```
## 53   AAPL   COKE  6.520817 12.91 14.9 19.19
## 54   AAPL    PEP  6.437457 12.91 14.9 19.19
## 55   AAPL   CSCO  6.172765 12.91 14.9 19.19
## 56   GOOG   TSCO  6.036614 12.91 14.9 19.19
## 57   NFLX    PEP  5.959804 12.91 14.9 19.19
## 58   NFLX   TSCO  5.915862 12.91 14.9 19.19
## 59   CSCO   TSCO  5.501285 12.91 14.9 19.19
## 60   INTC   TSCO  5.366053 12.91 14.9 19.19
## 61   ADBE   TSCO  5.353381 12.91 14.9 19.19
## 62   AAPL   GOOG  5.221159 12.91 14.9 19.19
## 63   NFLX    SPY  5.192385 12.91 14.9 19.19
## 64   AAPL    SPY  5.130378 12.91 14.9 19.19
## 65   AAPL   TSCO  4.688852 12.91 14.9 19.19
## 66    SPY   TSCO  4.669784 12.91 14.9 19.19
```

These cointegration test are sorted in descending oder in comparrsoin to the p-values of different levels. We have several candidate of stock pairs for modeling buildig such as:

ADBE/INTC MSFT/SPY INTC/MSFT

For this project we will be using Microsoft and Intel Corp stocks as pairs for our model.

Getting time series data

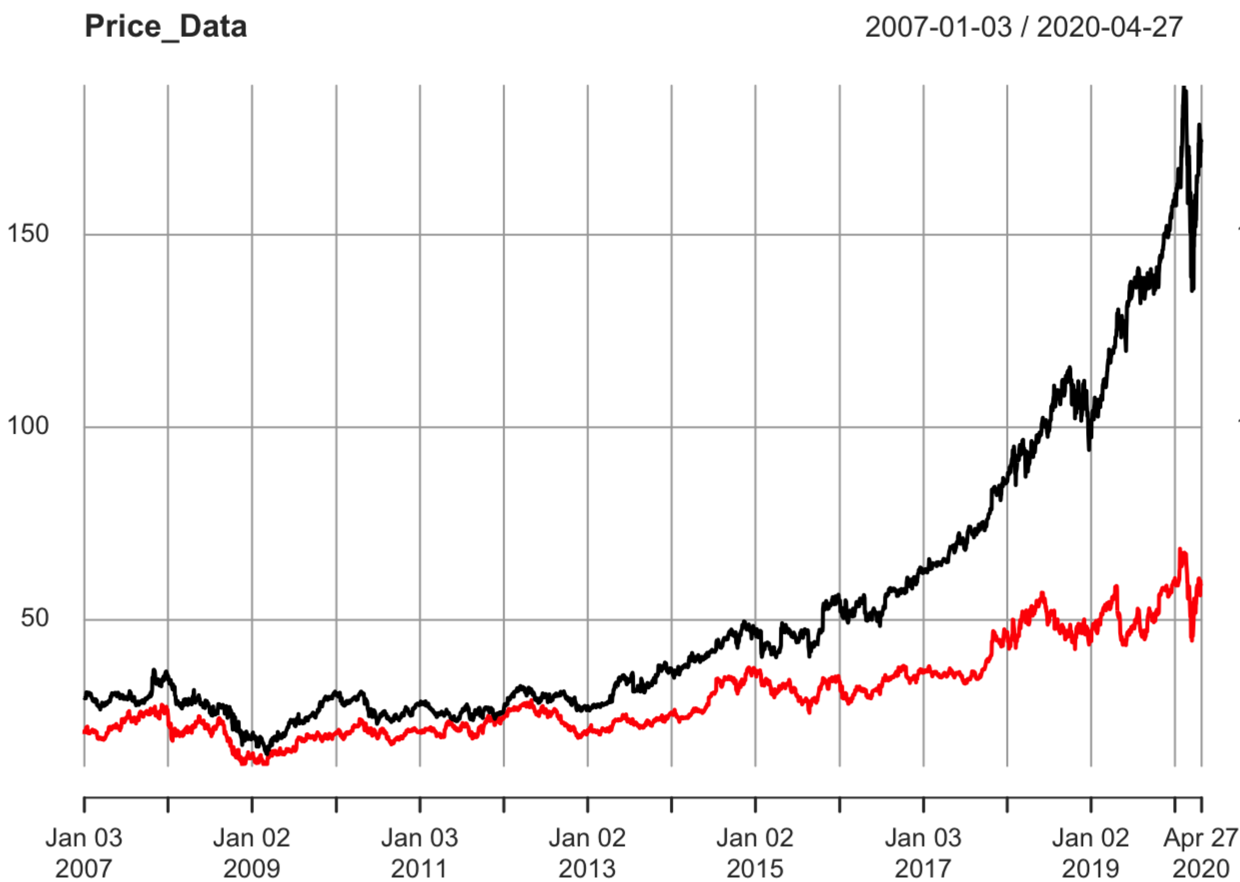
```
getSymbols("MSFT",src="yahoo")
```

```
## [1] "MSFT"
```

```
getSymbols("INTC",src="yahoo")
```

```
## [1] "INTC"
```

```
MSFT<- MSFT[,"MSFT.Close"]
INTC<- INTC[,"INTC.Close"]

Price_Data <- cbind(MSFT,INTC)
plot(Price_Data)
```



Getting the desired time period

The time period selected to test the model will be the period from 2017 to 2020.

```
start_date <- "2017-01-01"
end_date <- "2020-01-01"

MSFT <- MSFT[(index(MSFT) >= start_date & index(MSFT) <= end_date)]
INTC <- INTC[(index(INTC) >= start_date& index(INTC) <= end_date)]
```

Checking for missing data Calculating Daily returns

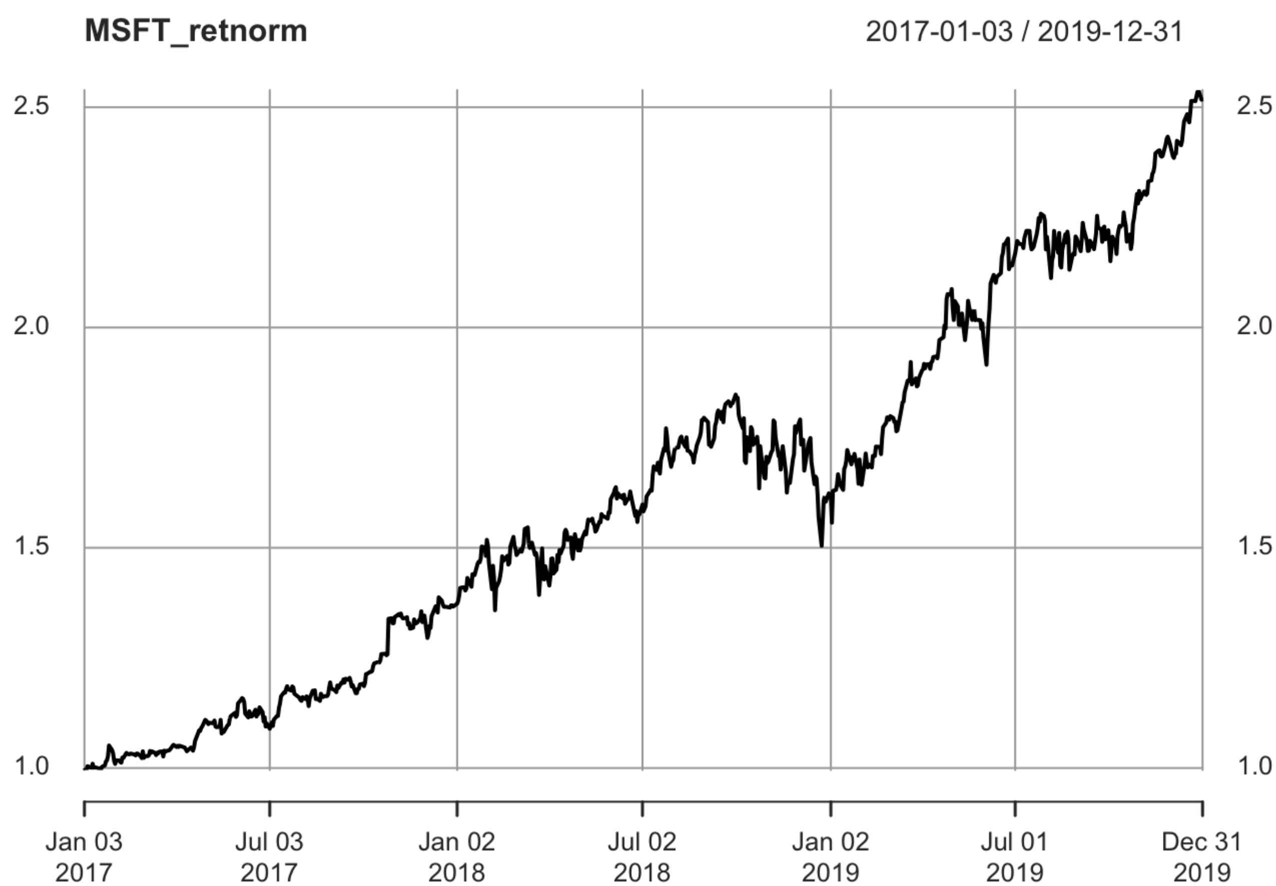
Incase of missing data the function "locf" will fill it with the previosly observed data after that we proceed to calculating the daily returns

```
MSFT <- na.locf(MSFT)
INTC <- na.locf(INTC)

MSFT_ret <- Delt(MSFT, k=1)
INTC_ret <- Delt(INTC, k=1)
```

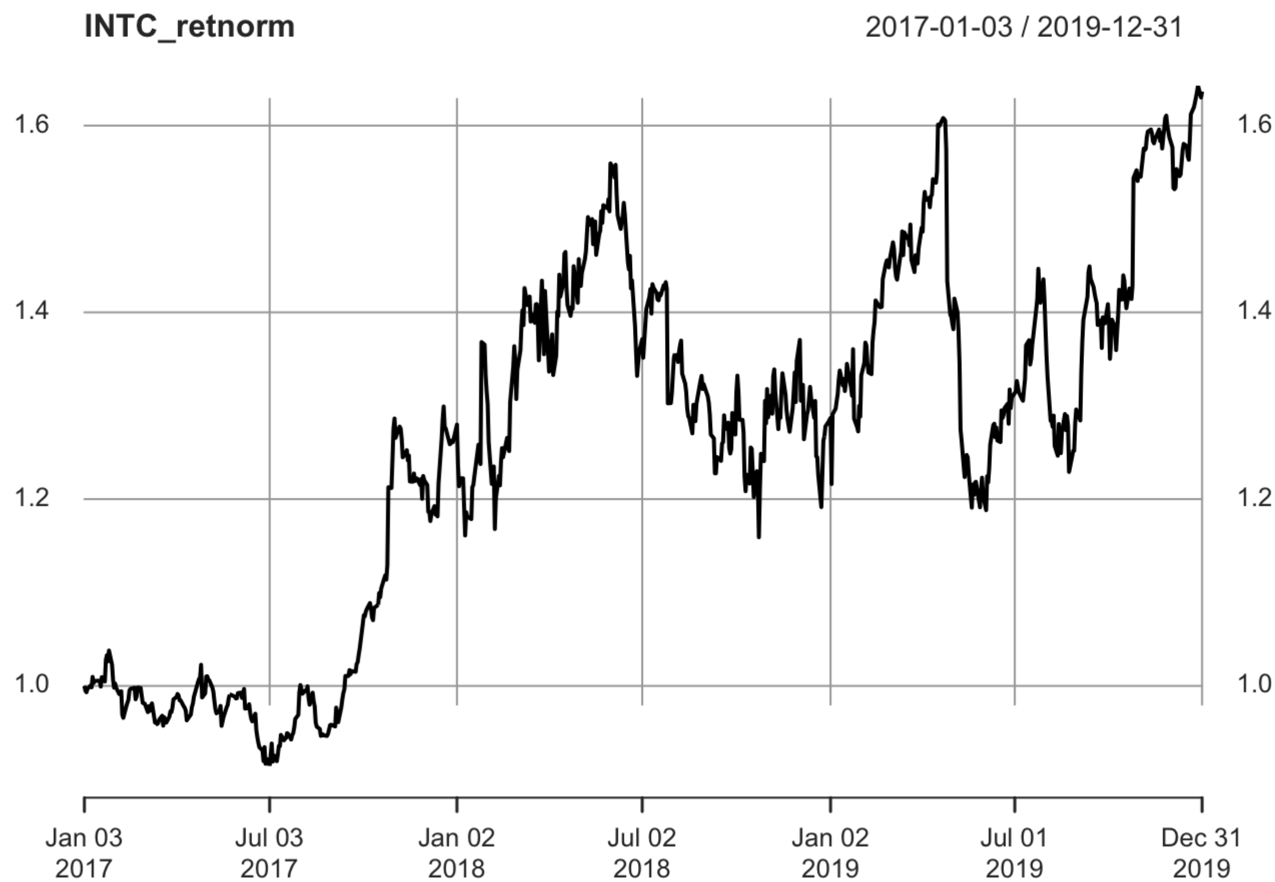
Normalization of prices and returns

Next we need to normalize the data of the stocks first and then check the distance between them and check their movement and correlation. In order to determine the normalized price for both the stocks we can use by taking the cumlative product of stock prices or we can divide the stock prices by the first day price. we have done applied the former to the daily returns and the latter to the prices. Then we plot the normalized price for both the stocks to visualize the normalized price

```
MSFT_ret <- round(MSFT_ret+1, 4)
MSFT_ret[1] <- 1
MSFT_retnorm <- cumprod(MSFT_ret)
plot(MSFT_retnorm)
```




```
INTC_ret <- round(INTC_ret+1, 4)
INTC_ret[1] <- 1
INTC_retnorm <- cumprod(INTC_ret)
plot(INTC_retnorm)
```




```
Normalized_returns <- cbind(MSFT_retnorm, INTC_retnorm)
plot(Normalized_returns)
```

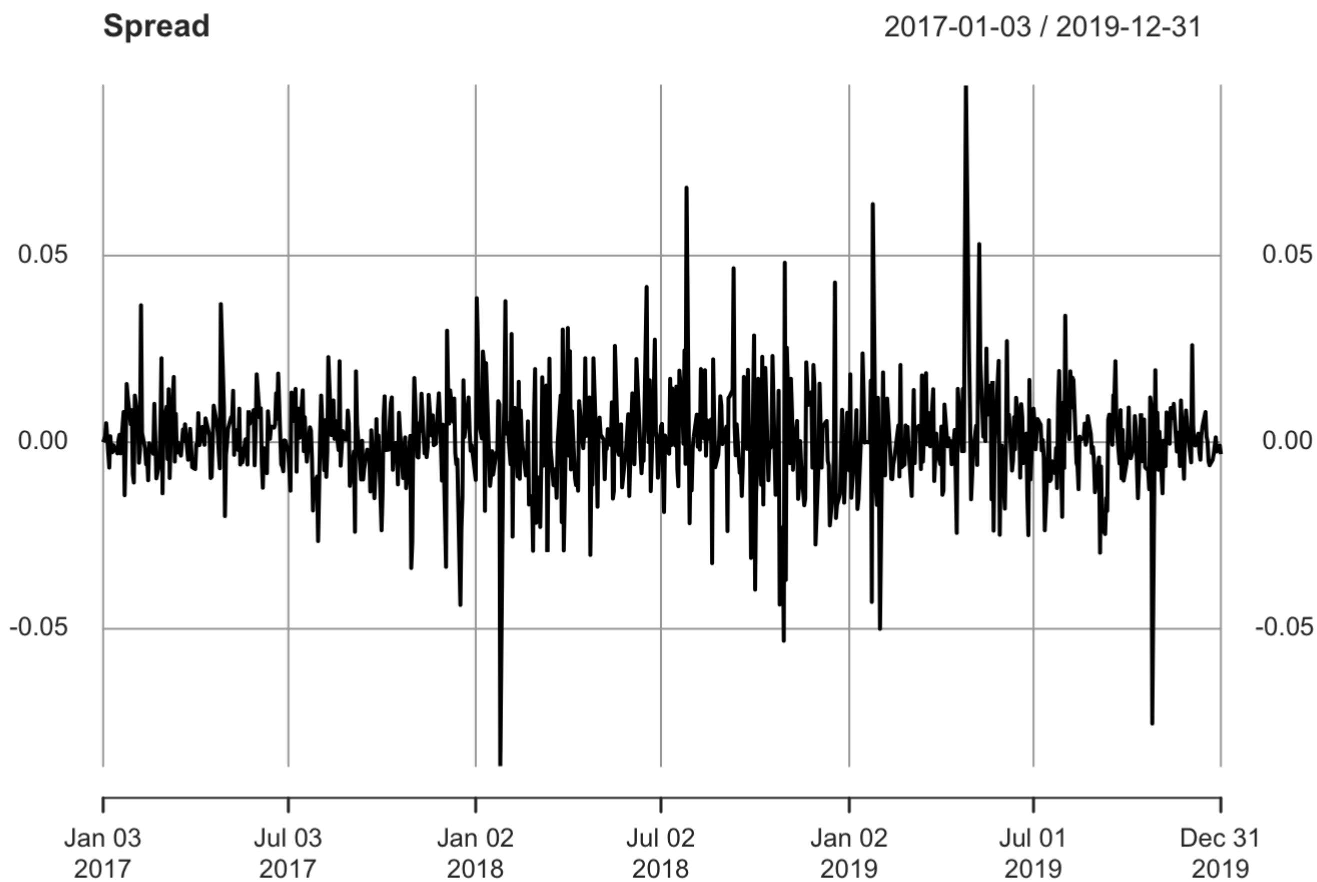

```
MSFT_ts <- xts(MSFT)
MSFT_norm_pr <-  MSFT_ts/MSFT_ts[[1]]

INTC_ts <- xts(INTC)
INTC_norm_pr <-  INTC_ts/INTC_ts[[1]]

Normalized_prices <- cbind(MSFT_norm_pr,INTC_norm_pr)
plot(Normalized_prices)
```

As we can see both prices as well as the returs tends to move together most of the time.

Calculating the spread of the returns and check the stationarity

We move to calculating the Spread which is the residual of the pairs movements in this case and try to validate the hypothesis that it follows stationary process.

```
Spread <- MSFT_ret - INTC_ret
plot(Spread)
```




```
adf.test(Spread)
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  Spread
## Dickey-Fuller = -8.4075, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

According to the visual inspection and ADF test the spread follows a stationary process except few moments of high volatility.

Calculating the Z-score

After we confirm that the Spread follows stationry process we move to calculating the Z-score or standardizing and normalizing of the spread using its mean and standard deviation for 10 day average.

First we obtaing the rollling mean and standard deviation for 10 days. then obtain the Z-score by dividing the diffrencing of each values of the spread from the mean by the standard deviation.

```
mean_dynamic <- rollapply(Spread, 10, mean)
std_dynamic <-  rollapply(Spread, 10, sd)

Z_spread <- (Spread - mean_dynamic)/std_dynamic
```

Generating signal

The idea behind the signal genrating process is to enter and exit the trading with comparison of the z-score and the bounding by critical values. This critical values are calculated from the mean and standard devitation of the spread. The logic of the critical values are as follows

```
# Critical value calculations

enter_short <- mean_dynamic + 3*std_dynamic # sell short
enter_long <- mean_dynamic - 3*std_dynamic # long buy

exit_short <- mean_dynamic - 1*std_dynamic # do nothing
exit_long <- mean_dynamic + 1*std_dynamic # do nothing


signal <- ifelse(Z_spread <= enter_long,1, ifelse(Z_spread >= enter_short,-1, ifelse(Z_s
pread >= exit_long ,0, ifelse(Z_spread <= exit_short,0,0))))
```

Making the Final tabel and merging the important data in one data frame

Since we are working with Closing prices, therefore we can act (BUY or SELL) on our signal next day only. So our return will depend on the return for the period next to that of the signal. Hence, we'll use the lag function to calculate the return of this strategy

```
trade_returns = lag(signal)*Spread
Output = merge(Spread, signal, trade_returns)

head(Output, 20)
```

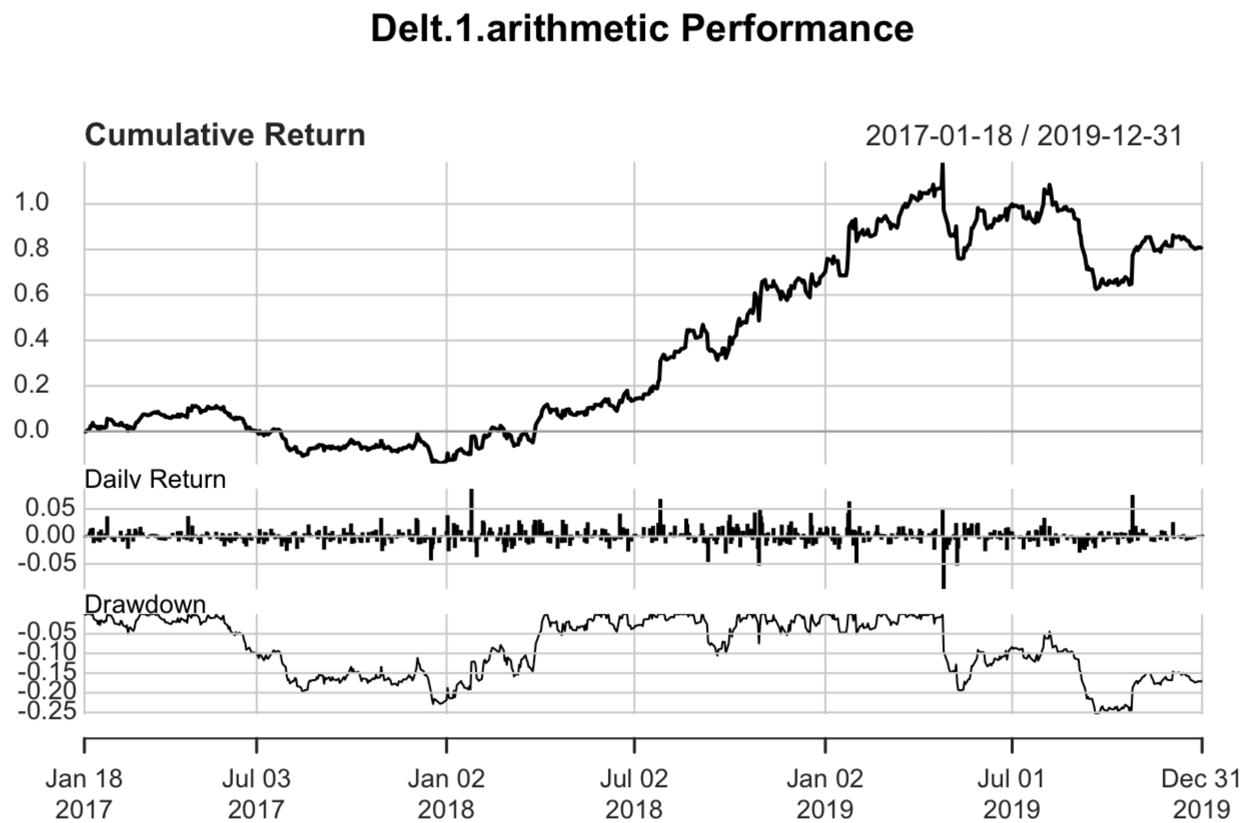
```
##            Delt.1.arithmetic Delt.1.arithmetic.1 Delt.1.arithmetic.2
## 2017-01-03            0.0000                  NA                  NA
## 2017-01-04            0.0007                  NA                  NA
## 2017-01-05            0.0016                  NA                  NA
## 2017-01-06            0.0051                  NA                  NA
## 2017-01-09           -0.0068                  NA                  NA
## 2017-01-10            0.0016                  NA                  NA
## 2017-01-11           -0.0021                  NA                  NA
## 2017-01-12           -0.0027                  NA                  NA
## 2017-01-13           -0.0008                  NA                  NA
## 2017-01-17           -0.0030                   1                  NA
## 2017-01-18            0.0006                  -1              0.0006
## 2017-01-19            0.0020                  -1             -0.0020
## 2017-01-20           -0.0030                   1              0.0030
## 2017-01-23            0.0081                  -1              0.0081
## 2017-01-24           -0.0142                   1              0.0142
## 2017-01-25           -0.0023                   1             -0.0023
## 2017-01-26            0.0156                  -1              0.0156
## 2017-01-27            0.0123                  -1             -0.0123
## 2017-01-30            0.0048                  -1             -0.0048
## 2017-01-31            0.0086                  -1             -0.0086
```

Evaluate our model with Performace Analytics

```
summary(as.ts(trade_returns))
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
## -0.095600 -0.006225  0.000000  0.000903  0.006800  0.086800        10
```

```
charts.PerformanceSummary(trade_returns)
```




```r
Return.cumulative(trade_returns)
```

```
##                     Delt.1.arithmetic
## Cumulative Return         0.8133304
```

```r
Return.annualized(trade_returns)
```

```
##                     Delt.1.arithmetic
## Annualized Return         0.2233441
```

```r
maxDrawdown(trade_returns)
```

```
## [1] 0.2548583
```

```r
SharpeRatio(trade_returns, Rf = 0, p=0.95, FUN = "StdDev")
```

```
##                                Delt.1.arithmetic
## StdDev Sharpe (Rf=0%, p=95%):         0.0628164
```

```r
SharpeRatio.annualized(trade_returns, Rf = 0)
```

```
##                                  Delt.1.arithmetic
## Annualized Sharpe Ratio (Rf=0%)         0.9786234
```

From the performance analytics summary above we can observe that the the cumulative return at the end of December 2019 is almost at 80%. The strategy seems very risky with several drops in trades and a maximum drop down of 25%. Other metrics includes

Annualized return = 22%

Sharpe ratio = 6.2%

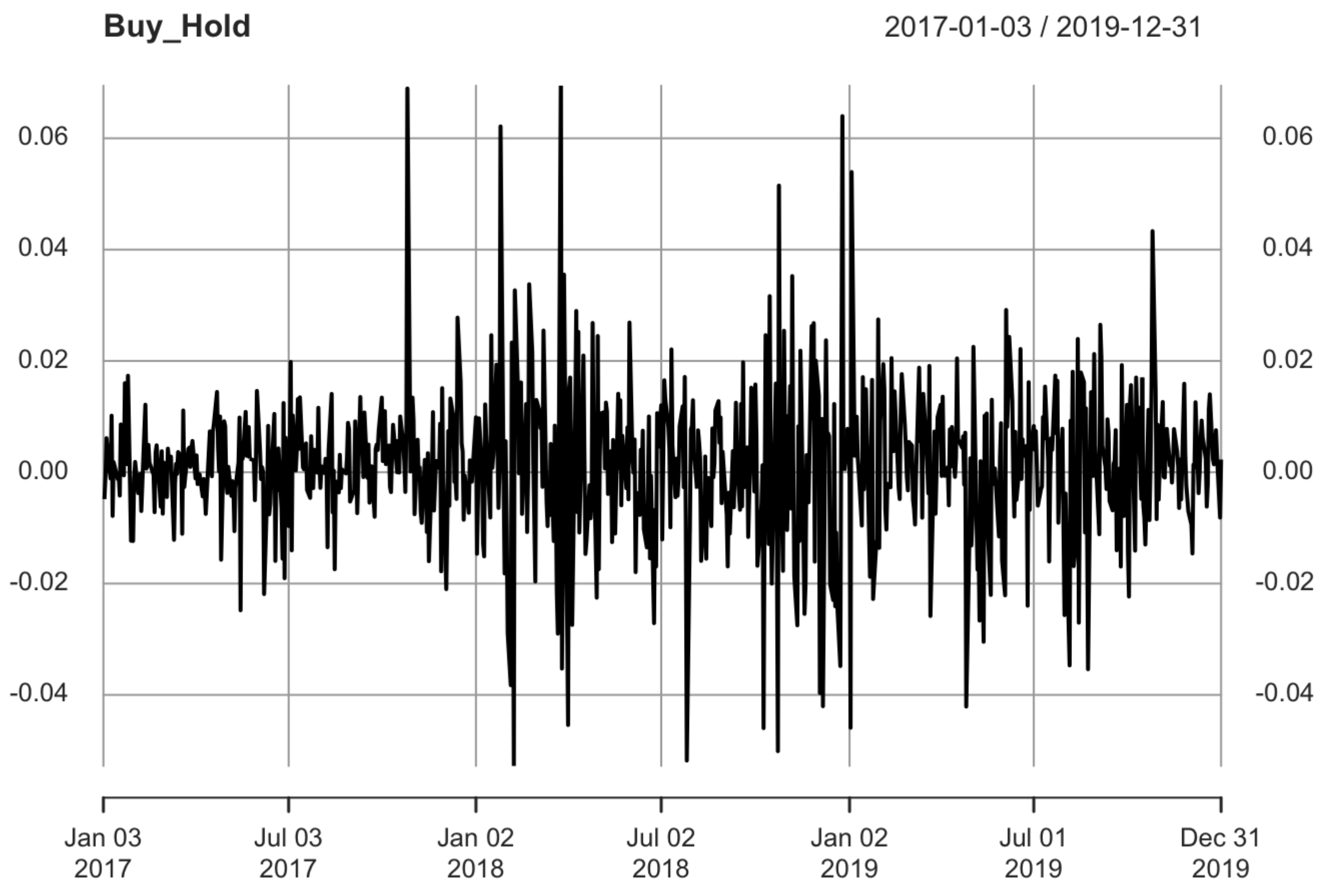
Annualized Sharpe ratio = 97%

our model shows high cumulative refund but it is very risky according to the annualized sharp ratio.

Buy and Hold strategy for Benchmarking Comparison

We are using this stratedy to compare beween the above statistical mean reversing arbitrage model and holding the assest for longer time. Buy and Hold. The idea is that we buy a certain asset and do not do anything for the entire duration of the investment horizon. So at the first possible date, we buy as much stock as we can with our capital and do nothing later.This simple strategy can also be considered a benchmark for more advanced onesbecause there is no point in using a very complex strategy that generates less money than buying once and doing nothing.

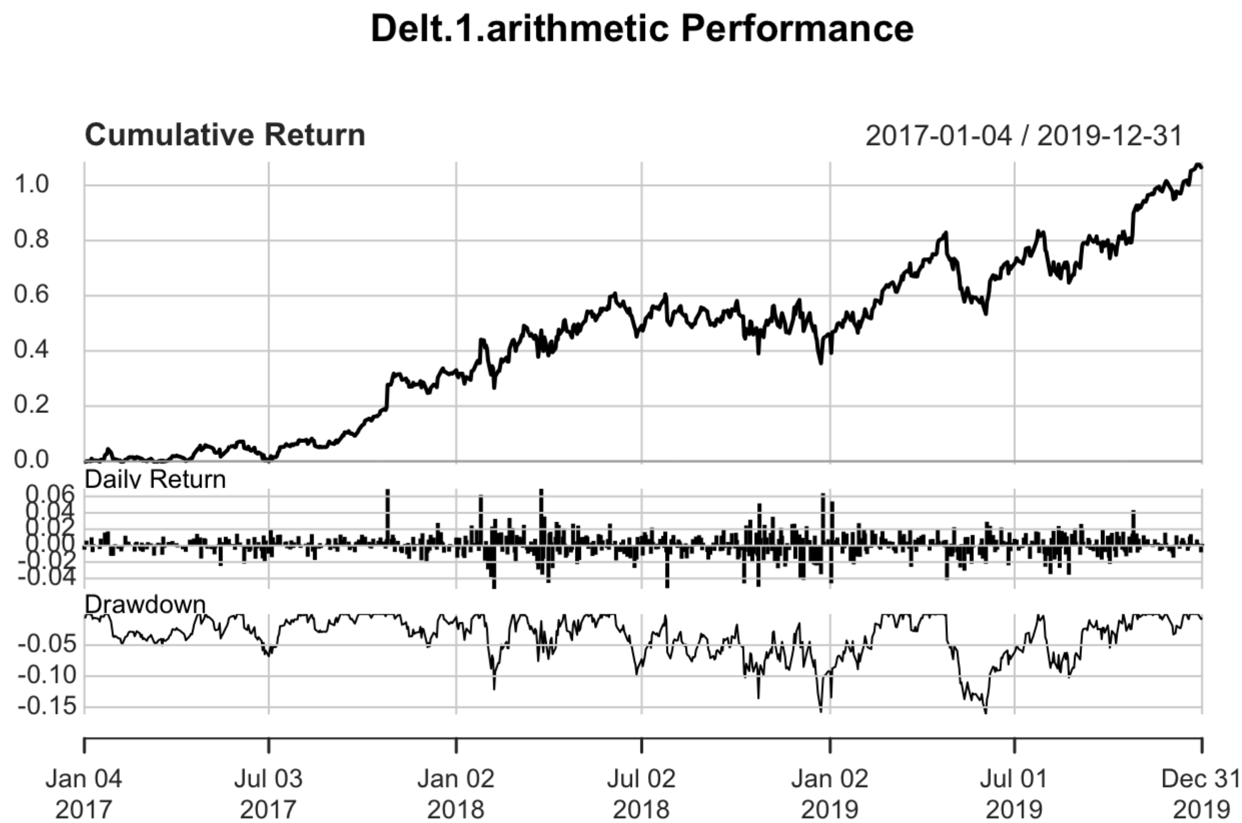
```
# To calculate the benchmark data we hold the returs of both assets in equal weights in
 the same time period.

MSFT_ret <- Delt(MSFT, k=1)
INTC_ret <- Delt(INTC, k=1)


Buy_Hold <- (MSFT_ret + INTC_ret)/2

plot(Buy_Hold)
```




```
summary(as.ts(Buy_Hold))
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's 
## -0.052769 -0.005729  0.001257  0.001059  0.008180  0.069457         1
```

```
charts.PerformanceSummary(Buy_Hold)
```




```
Return.cumulative(Buy_Hold)
```

```
##                     Delt.1.arithmetic
## Cumulative Return            1.070146
```

```
Return.annualized(Buy_Hold)
```

```
##                     Delt.1.arithmetic
## Annualized Return            0.275714
```

```
maxDrawdown(Buy_Hold)
```

```
## [1] 0.1613969
```

```
SharpeRatio(Buy_Hold, Rf = 0, p=0.95, FUN = "StdDev")
```

```
##                                 Delt.1.arithmetic
## StdDev Sharpe (Rf=0%, p=95%):          0.07779448
```

```
SharpeRatio.annualized(Buy_Hold, Rf = 0)
```

```
##                                      Delt.1.arithmetic
## Annualized Sharpe Ratio (Rf=0%)              1.275708
```

The buy and hold model shows that higher cumulative return than the co-integration arbitrage model but it tend to have lower annualized returns and much higher risk.

Annualized return = 27%

Max drawdown = 16%

Sharpe ratio = 7.8%

Annualized Sharpe ratio = 127%

with this resuts ints much better to use the buy and hold one than the statistcal arbitrage model we developed.

Improving the algorithmic trading strategy

Our model seems to have almost similar resluts to the benchmark model, several improvments could be made to construct a better model in the future.

Using highly correlated stocks as example, if the price of the stocks don't revert back to their expected mean position, it is advisable to shift your trade bias at an intraday level to a more long or more short position in order to take advantage of the latest market trend. It is also advisable to watch out for cointegration and not just correlation because pair stocks could have divergent trend over long periods and still appear to be correlated.

Trading volumes can be used to monitor the demand of assets and can in turn be used to identify irregular changes that can affect the relationship between the assets that make up the pair. According to Engelberg, Gao and Jagannathan (2009) it's expected that if a common shock exists then a volume increase will occur in both assets and if that shock only affects one of the assets then the volume increase will be confined to an increase in the respective assets volume

Adding transaction costs and fees into the equation and using data from twitter for sentimental anlysis could also provide invaluabel insights for trading strategies

The above mentioned adjustment can greatly improve andgenerate extensive set of criterias for our signals and allow as to execute more reliable trades.

Conclusion

In conclusion, in this project we implemented, Cointegration based pairs trading with MSFT/INTC pair. the model is viable for automated trading, and outperforms the benchmark. However, we can see that the strategy is very risky, with large drawdowns, but also with high returns.Thus we believe this model could improve when augumented with additonal data, account for trasanctional costs and coule be part of a protofilio to improve the overall performance of the portofolio returns.