# Index vs Equity Volatility Analysis

Daniel Peslherbe

January 6th, 2021

## Investing basics

The world of investing, for the uninitiated, can seem like an impossible thing to understand, and while no one has been able to completely predict the market, there are some general concepts that have stood the test of time. For many, the issue with investing is the availability of time: unless you have been blessed with fortunes, most of your time in a week is consumed by work, leaving very little time to day trade. In this case, the solution offered is usually in the form of *passive investing* in the form of different types of funds (Mutual, Exchange-Traded, etc...). Passive investing is a solution to help those who would otherwise be too busy to trade equities or funds during the day when the market is open. In this case, we will try to show the differences in volatility, and share price for the S&P 500 Index.

The goal of this report is not to show which specific equities have outperformed the index, or to predict their future value, but simply to compare over time to show why passive investing in index funds can be a less risky approach.

## Methodology

### Acquiring Data

To lead this analysis, we must first start by acquiring the data for S&P 500 Index, as well as the data for the equities the index comprises. We will be doing through the help of the BatchGetSymbols package (found here: link). First, we use the rvest package to scrape the names of companies found in the S&P 500 Index, as well as their respective tickers (using link). Then, we make sure to clean up the ticker names (for example, scraping gives us BRK.B for Berkshire Hathaway Class B stock, whereas Yahoo Finance uses the BRK-B ticker notation).

For this first part, we will use data from the start of this calendar year (January 1st, 2020 to the current date using Sys.Date()), and taking stock prices at the opening and closing of the market, as well as the daily high and lows of the stock price and its trading volume and adjusted price for the day. BatchGetSymbols also calculates the daily return of the stock price based on either its adjusted or closing price, compared to the previous day.

```
library(tidyverse)
library(rvest)
library(ggplot2)
library(gridExtra)
library(BatchGetSymbols)

url <- 'https://en.wikipedia.org/wiki/List_of_S%26P_500_companies'
webpage <- read_html(url)
companies <- webpage %>% html_nodes('#constituents td:nth-child(2)') %>% html_text()
tickers <- webpage %>% html_nodes('#constituents td:nth-child(1)') %>% html_text()
tickers <- sub('\n', '', tickers)
```

```
tickers <- sub('\\.', '-', tickers)
start.date <- '2020-01-01'
end.date <- Sys.Date()
frequency.of.data <- 'daily'
```

```
stocks <- BatchGetSymbols(tickers = tickers, first.date = start.date,
                          last.date = end.date, freq.data = frequency.of.data)
```

```
head(stocks$df.control)
```

```
## # A tibble: 6 x 6
##   ticker src   download.status total.obs perc.benchmark.dates threshold.decision
##   <chr>  <chr> <chr>                <int>                <dbl> <chr>
## 1 MMM    yahoo OK                     256                    1 KEEP
## 2 ABT    yahoo OK                     256                    1 KEEP
## 3 ABBV   yahoo OK                     256                    1 KEEP
## 4 ABMD   yahoo OK                     256                    1 KEEP
## 5 ACN    yahoo OK                     256                    1 KEEP
## 6 ATVI   yahoo OK                     256                    1 KEEP
```

```
head(stocks$df.tickers)
```

```
##   price.open price.high price.low price.close  volume price.adjusted   ref.date
## 1     177.68     180.01    177.14      180.00 3601700       175.0395 2020-01-02
## 2     177.02     178.66    175.63      178.45 2466900       173.5322 2020-01-03
## 3     177.15     178.71    176.35      178.62 1998000       173.6975 2020-01-06
## 4     178.28     178.51    176.82      177.90 2173000       172.9973 2020-01-07
## 5     178.00     181.50    177.65      180.63 2758300       175.6521 2020-01-08
## 6     181.51     181.59    179.76      181.20 2746300       176.2064 2020-01-09
##   ticker ret.adjusted.prices ret.closing.prices
## 1    MMM                 NA                NA
## 2    MMM       -0.0086111605      -0.0086111278
## 3    MMM        0.0009526417       0.0009526366
## 4    MMM       -0.0040309447      -0.0040309093
## 5    MMM        0.0153458149       0.0153457622
## 6    MMM        0.0031555270       0.0031555776
```

```
index <- BatchGetSymbols(tickers = '^GSPC', first.date = start.date,
                         last.date = end.date, freq.data = frequency.of.data)
```

```
head(index$df.control)
```

```
## # A tibble: 1 x 6
##   ticker src   download.status total.obs perc.benchmark.dates threshold.decision
##   <chr>  <chr> <chr>                <int>                <dbl> <chr>
## 1 ^GSPC  yahoo OK                     256                    1 KEEP
```

```
head(index$df.tickers)
```

```
##   price.open price.high price.low price.close     volume price.adjusted
## 1    3244.67    3258.14   3235.53     3257.85 3458250000        3257.85
## 2    3226.36    3246.15   3222.34     3234.85 3461290000        3234.85
## 3    3217.55    3246.84   3214.64     3246.28 3674070000        3246.28
## 4    3241.86    3244.91   3232.43     3237.18 3420380000        3237.18
## 5    3238.59    3267.07   3236.67     3253.05 3720890000        3253.05
## 6    3266.03    3275.58   3263.67     3274.70 3638390000        3274.70
##     ref.date ticker ret.adjusted.prices ret.closing.prices
```

```
## 1 2020-01-02  ^GSPC                  NA                 NA
## 2 2020-01-03  ^GSPC        -0.007059871       -0.007059871
## 3 2020-01-06  ^GSPC         0.003533373        0.003533373
## 4 2020-01-07  ^GSPC        -0.002803238       -0.002803238
## 5 2020-01-08  ^GSPC         0.004902451        0.004902451
## 6 2020-01-09  ^GSPC         0.006655262        0.006655262
```

### Data Cleaning

While BatchGetSymbols already removes stock data if it does not the threshold requirement in stocks$df.stocks, we will check which stocks do not meet the threshold requirements:

```
invalid.index <- which(stocks$df.control$threshold.decision == 'OUT')
valid.index <- which(stocks$df.control$threshold.decision == 'KEEP')
invalid.ticker <- stocks$df.control$ticker[invalid.index]
invalid.ticker
```

```
## [1] "VNT"
```

Somehow, BatchGetSymbols shows data for 2020-10-02 in double (influencing the return, which becomes 0, since there is no variation between the same day price). Thus, we also make sure to clean this up as well:

```
wrong.info.index <- stocks$df.tickers %>% filter(ref.date == '2020-10-02' &
                                                   ret.adjusted.prices == 0)
right.info.index <- stocks$df.tickers %>% filter(ref.date == '2020-10-02' &
                                                   !ret.adjusted.prices == 0)
stocks$df.tickers <- stocks$df.tickers %>% filter(!ref.date == '2020-10-02')
stocks$df.tickers <- bind_rows(stocks$df.tickers, right.info.index)
stocks$df.tickers <- stocks$df.tickers %>% mutate(ref.date =
                                                    as.Date(ref.date,
                                                            '%d-%m-%Y')) %>% arrange(ref.date)
index$df.tickers <- index$df.tickers[-192,]
```

Thus, let us remove them from our control data, as we will make no use of them;

```
stocks$df.control <- stocks$df.control %>%
  filter((threshold.decision != 'OUT'))
```

## Random Sampling for Analysis

In this section, we will use Random Sampling with the sample function on our Valid Index to select 25 stocks at Random:

```
set.seed(1)
random.sampling.index <- sample(valid.index, 25, replace = FALSE)
random.sample.stocks <- stocks$df.control$ticker[random.sampling.index]
random.sample.stocks
```

```
##  [1] "MS"   "EMR"  "COST" "LUV"  "VFC"  "MPC"  "KEY"  "VTR"  "FRT"  "MXIM"
## [11] "WM"   "CAH"  "KR"   "PNR"  "TSLA" "NFLX" "SJM"  "NTAP" "BF-B" "GPC"
## [21] "AME"  "CI"   "GS"   "PRGO" "EW"
```

### First Analysis (Single Random Sample 1: MS)

#### Cumulative Returns

We will explain our analytical process for MS; for the following stocks from our Random Sampling, the methodology will remain the same, but we will do without explanations.

Let us assume that we start with a certain capital, i.e. let us say we have a slush fund of 100,000$ available to invest at the start of 2020 (how lucky !), and we feel very confident in Morgan Stanley; thus we decide to invest it all in MS stock (we assume that we can have partial stock owned):

```
capital <- 100000
shares.data <- filter(stocks$df.tickers, ticker == random.sample.stocks[1])
shares.number <- capital/shares.data$price.open[1]
print(c('number of shares is', shares.number))
```

```
## [1] "number of shares is" "1953.12496185303"
```

```
shares.price <- shares.data$price.open[1]
print(c('price of shares was', shares.price))
```

```
## [1] "price of shares was" "51.200001"
```

Then, on the opening of the market in 2020, we are able to purchase 1953.125 shares of MS stock. What if we wanted to the purchase the index instead ? (obviously, no one can buy the actual index, but many funds are available that follow the index; in this case, let us assume that the actual index is purchaseable).

```
index.data <- index$df.tickers
index.number <- capital/index.data$price.open[1]
print(c('number of index shares is', index.number))
```

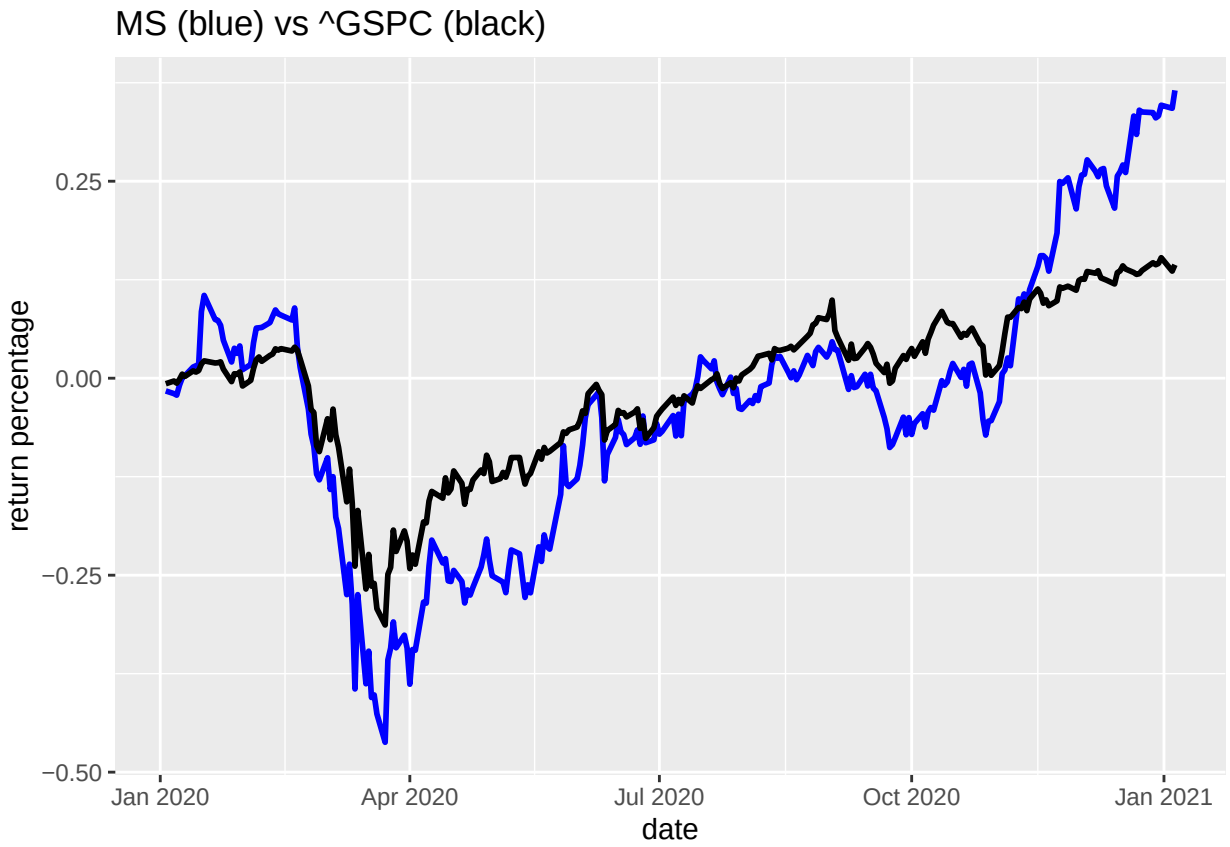
```
## [1] "number of index shares is" "30.8197759414494"
```

```
index.price <- index.data$price.open[1]
print(c('price of index shares was', index.price))
```

```
## [1] "price of index shares was" "3244.669922"
```

Then, let us look at the return of the selected stock and the index throughout the year; we then plot both returns for comparison:

```
shares.daily.return <- cumprod(1+(shares.data$ret.adjusted.prices[-1]))-1
index.daily.return <- cumprod(1+(index.data$ret.adjusted.prices[-1]))-1
return.data <- tibble(date = index.data$ref.date[-1])
return.data <- return.data %>% add_column(shares.return.perc = shares.daily.return)
return.data <- return.data %>% add_column(index.return.perc = index.daily.return)
ggplot() +
  geom_line(data = return.data, aes(date, shares.return.perc), color = 'blue', size = 1) +
  geom_line(data = return.data, aes(date, index.return.perc), color = 'black', size = 1) +
  labs(title = paste0('MS (blue) vs ^GSPC (black)'),
       undertitle = 'Cumulative return', x = 'date', y = 'return percentage')
```




In this case, the returns for MS have outpaced the index, but, what are the chances of having picked MS at random ? Assuming our pick is entirely random, and follows a uniform distribution, and there are 501 possible stocks (since we are picking stocks with all daily data available) to choose from is $1/501 = 0.001996008$ or 0.2%, which is highly unlikely. Before considering another stock, we will take a closer comparative look at MS and the index, as there is still more numbers to crunch (thankfully !).

### Return Variation

As the returns previously calculated are cumulative, i.e. they are Year-To-Date returns of the chosen stock/index, then the equity/index value growth over the year is the final available value;

```
shares.return.YTD <- return.data$shares.return.perc[dim(return.data)[1]]
index.return.YTD <- return.data$index.return.perc[dim(return.data)[1]]
print(c('MS YTD return is ', shares.return.YTD))
```

```
## [1] "MS YTD return is " "0.365354274765403"
```

```
print(c('^GSPC YTD return is ', index.return.YTD))
```

```
## [1] "^GSPC YTD return is " "0.143956202239995"
```

However, this return percentage spans well over 10 months, what about during the year ? Has the index been more prone to changes (sudden or gradual) than MS ? In this case, we will explore the cumulative standard deviation of the stock and the index from themselves respectively, taking only return values from the begining of the year.

```
shares.return.std.dev.YTD <- sqrt(sum((shares.data$ret.adjusted.prices[-1] -
                                       shares.return.YTD)^2))
index.return.std.dev.YTD <- sqrt(sum((index.data$ret.adjusted.prices[-1] -
```

```
                                          index.return.YTD)^2))
shares.return.std.dev.YTD/length(shares.daily.return)
```

```
## [1] 0.02291893
```

```
index.return.std.dev.YTD/length(index.daily.return)
```

```
## [1] 0.009086448
```

While this does not seem like a big variation, we also have to consider the amount of money invested; in this case, the difference in standard deviations would represent a 2500$+ for the shares, versus only 525$+. Would you be comfortable with this difference if you could not follow the market closely at all times ?

```
((shares.return.std.dev.YTD/length(shares.daily.return) + 1) * capital) - capital
```

```
## [1] 2291.893
```

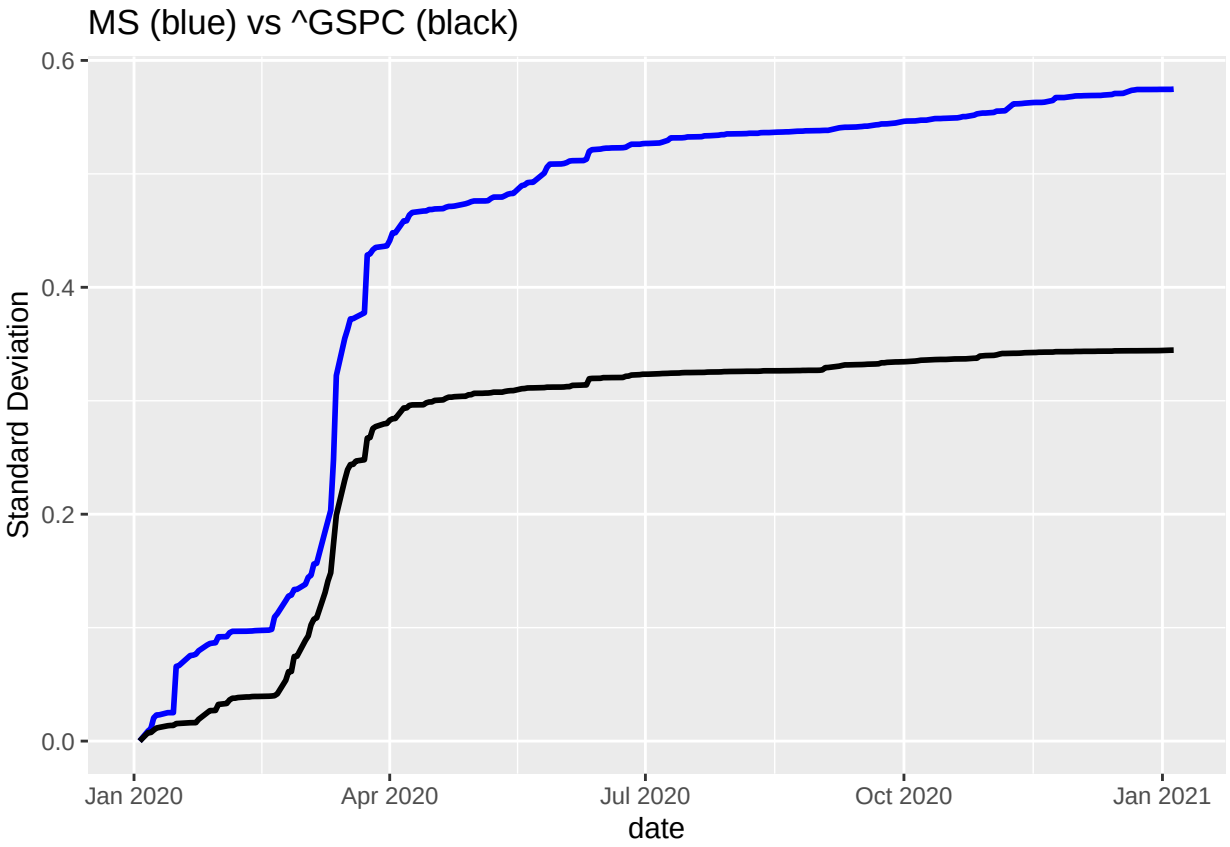
```
((index.return.std.dev.YTD/length(index.daily.return) + 1) * capital) - capital
```

```
## [1] 908.6448
```

Then, what if we look at volatility through the year; this helps us see if the stock or index volatility is due one or more specific events (hint: the markets had a very bad reaction to the initial COVID measures in March), and whether both the index and stock's cumulative standard deviations are similar; if there cumulative standard deviation curves are very different, this could mean that there are different factors to look at before comparing them (which we will not do, but it would signal that the state of the market/index is not completely reflected in the particular stock).

```
shares.return.mean.cumul <- c()
index.return.mean.cumul <- c()
for (i in 2:length(shares.data$ret.adjusted.prices)) {
  shares.return.mean.cumul <- c(shares.return.mean.cumul,
                                mean(shares.data$ret.adjusted.prices[2:i]))
  index.return.mean.cumul <- c(index.return.mean.cumul,
                               mean(index.data$ret.adjusted.prices[2:i]))
}
shares.list.cumul <- vector(mode = 'list', length =
                              (length(shares.data$ret.adjusted.prices)-1))
index.list.cumul <- vector(mode = 'list', length =
                             (length(index.data$ret.adjusted.prices)-1))
shares.return.std.dev.cumul <- c()
index.return.std.dev.cumul <- c()
for (j in 1:(length(shares.data$ret.adjusted.prices)-1)) {
  shares.list.cumul[[j]] <- c(shares.data$ret.adjusted.prices[2:(1+j)])
  shares.return.std.dev.cumul <- c(shares.return.std.dev.cumul,
                                   sqrt(sum((shares.list.cumul[[j]] -
                                               shares.return.mean.cumul[j])^2)))
  index.list.cumul[[j]] <- c(index.data$ret.adjusted.prices[2:(1+j)])
  index.return.std.dev.cumul <- c(index.return.std.dev.cumul,
                                  sqrt(sum((index.list.cumul[[j]] -
                                              index.return.mean.cumul[j])^2)))
}
return.data <- return.data %>% add_column(shares.return.std.dev =
                                            shares.return.std.dev.cumul)
return.data <- return.data %>% add_column(index.return.std.dev =
                                            index.return.std.dev.cumul)
ggplot() +
```

```
  geom_line(data = return.data, aes(date, shares.return.std.dev), color = 'blue',
            size = 1) +
  geom_line(data = return.data, aes(date, index.return.std.dev), color = 'black',
            size = 1) +
  labs(title = paste0('MS (blue) vs ^GSPC (black)'),
       undertitle = 'Cumulative standard deviations over the year',
       x = 'date', y = 'Standard Deviation')
```



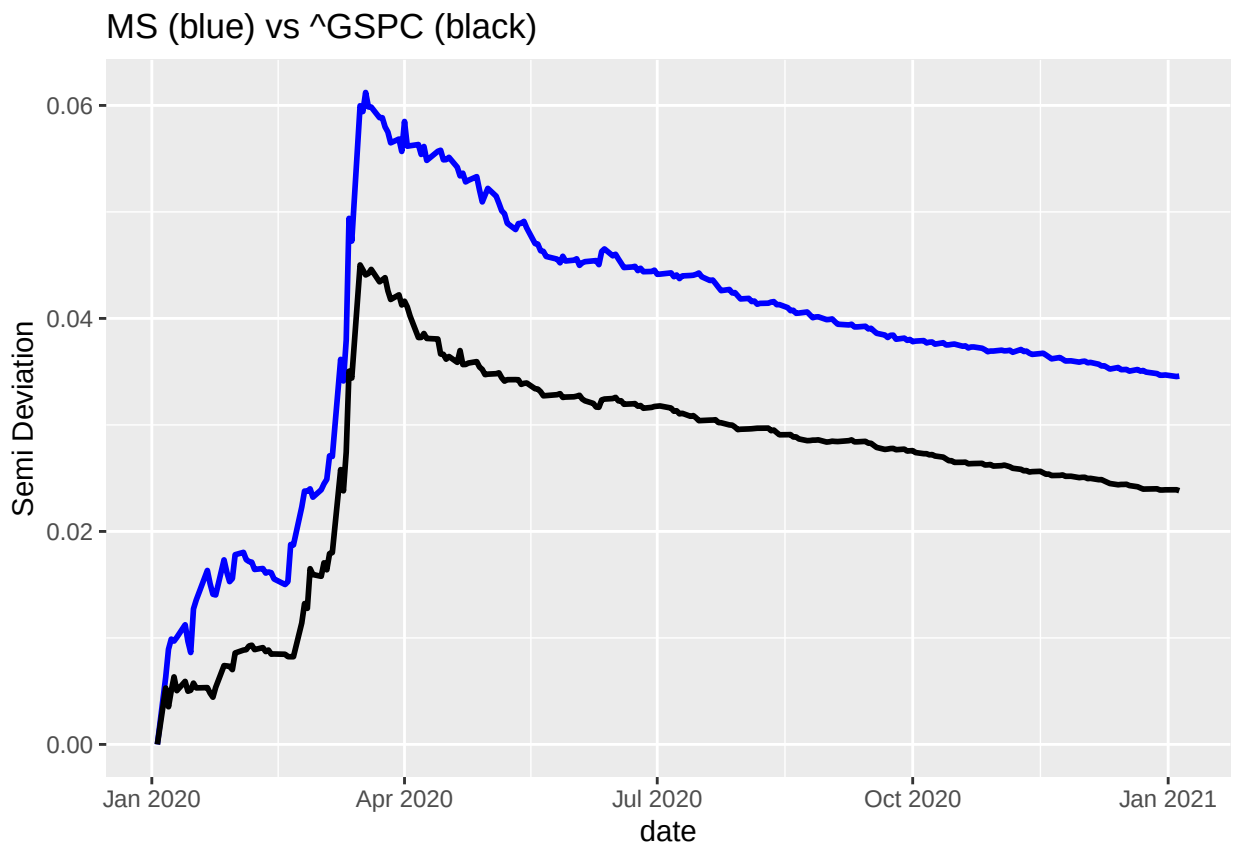

We can see here that MS and index are pretty well correlated, the changes in variance happen around the same time, but the size of the change is different. This shows that MS was more likely to change price by a larger percentage than the index (whether it was a positive or negative price change). However, what if we only wanted to look at changes that negatively impacted the stock or the index ? Luckily, the answer to this problem is found through the semi-deviation metric.

### Return Semi-Variation & Downside Deviation

The semi-deviation metric is about calculating variations in values where the values are under their average.

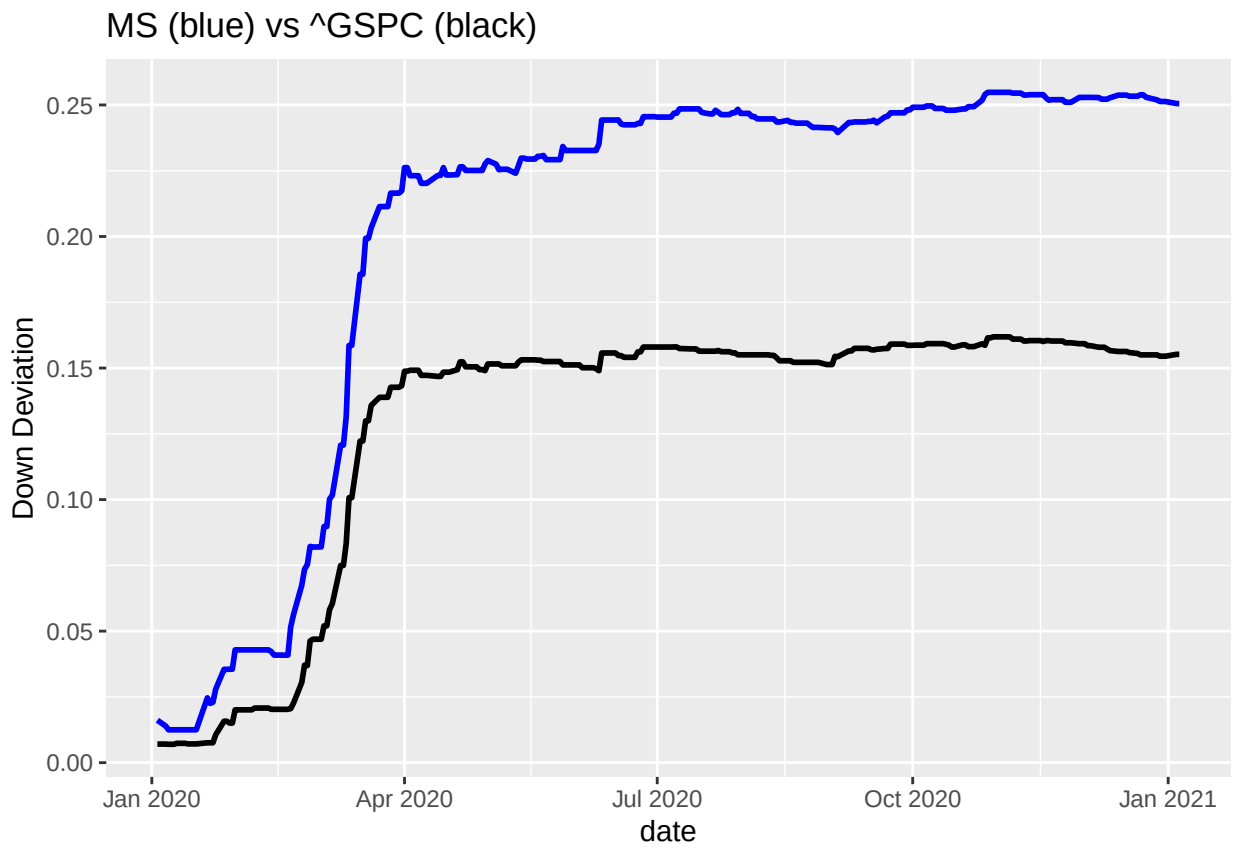
```
shares.return.semi.dev.cumul <- c()
index.return.semi.dev.cumul <- c()
for (j in 1:length(shares.list.cumul)) {
  shares.return.cumul <- shares.list.cumul[[j]]
  shares.return.semi.dev.values <- shares.return.cumul[shares.return.cumul <=
                                                         mean(shares.return.cumul)]
  shares.return.semi.dev.cumul <- c(shares.return.semi.dev.cumul,
                                    sqrt((1/length(shares.return.semi.dev.values))*
                                           sum((shares.return.semi.dev.values -
```

```
                                                    mean(shares.return.cumul))^2)))
  index.return.cumul <- index.list.cumul[[j]]
  index.return.semi.dev.values <- index.return.cumul[index.return.cumul <=
                                                       mean(index.return.cumul)]
  index.return.semi.dev.cumul <- c(index.return.semi.dev.cumul,
                                   sqrt((1/length(index.return.semi.dev.values))*
                                          sum((index.return.semi.dev.values -
                                                 mean(index.return.cumul))^2)))
}
return.data <- return.data %>% add_column(shares.return.semi.dev =
                                            shares.return.semi.dev.cumul)
return.data <- return.data %>% add_column(index.return.semi.dev =
                                            index.return.semi.dev.cumul)
ggplot() +
  geom_line(data = return.data, aes(date, shares.return.semi.dev), color = 'blue',
            size = 1) +
  geom_line(data = return.data, aes(date, index.return.semi.dev), color = 'black',
            size = 1) +
  labs(title = paste0('MS (blue) vs ^GSPC (black)'),
       undertitle = 'Cumulative standard deviations over the year',
       x = 'date', y = 'Semi Deviation')
```




Thus, we see that MS have more variation when it comes to underperforming returns. To push this even further, what if we don't use the average as our differentiator for semi-deviation, but a 0 return instead (i.e. look at variation for negative returns only). This is called downside deviation.

```
shares.return.down.dev.cumul <- c()
index.return.down.dev.cumul <- c()
for (j in 1:length(shares.list.cumul)) {
  shares.return.cumul <- shares.list.cumul[[j]]
  shares.return.down.dev.values <- shares.return.cumul[shares.return.cumul <= 0]
  shares.return.down.dev.cumul <- c(shares.return.down.dev.cumul,
                                    sqrt((1/length(shares.return.down.dev.values))*
                                           sum((shares.return.down.dev.values))^2))
  index.return.cumul <- index.list.cumul[[j]]
  index.return.down.dev.values <- index.return.cumul[index.return.cumul <= 0]
  index.return.down.dev.cumul <- c(index.return.down.dev.cumul,
                                   sqrt((1/length(index.return.down.dev.values))*
                                          sum((index.return.down.dev.values))^2))
}
return.data <- return.data %>% add_column(shares.return.down.dev =
                                            shares.return.down.dev.cumul)
return.data <- return.data %>% add_column(index.return.down.dev =
                                            index.return.down.dev.cumul)
ggplot() +
  geom_line(data = return.data, aes(date, shares.return.down.dev), color = 'blue',
            size = 1) +
  geom_line(data = return.data, aes(date, index.return.down.dev), color = 'black',
            size = 1) +
  labs(title = paste0('MS (blue) vs ^GSPC (black)'),
       undertitle = 'Cumulative standard deviations over the year',
       x = 'date', y = 'Down Deviation')
```

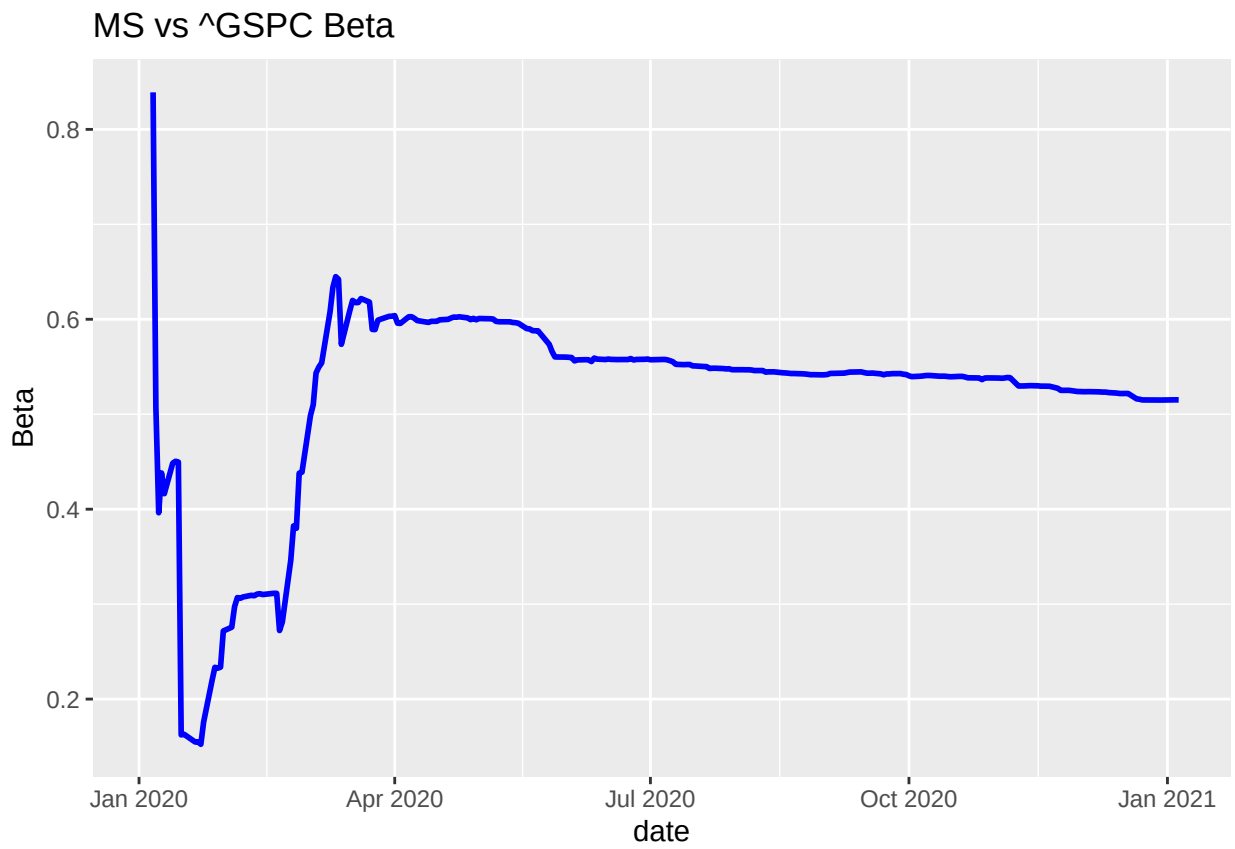



Thus, MS has more variation than ^GSPC in every case; whether the returns are positive, above-average, under-average, or negative, MS presents more variation than ^GSPC. Thus, while MS has shown better returns, it can also be considered more risky. Thus, let us try to make some verifications by using more traditional financial investment metrics.

### Beta

A widely used investment metric the Beta of a particular equity offering: it measures the volatility in comparison to the overall systemic risk of the market (in this case, the market we track is the S&P 500 Index). A Beta of value 1.0 would indicate that the particular stock equity is theoretically perfectly or near perfectly correlated witht the market tracked. A Beta **under** this value would indicate less risk than the market, while a Beta **over** this value would be considered riskier than the market.

```
beta.cumul <-c()
for (j in 1:length(shares.list.cumul)) {
  beta.cumul <- c(beta.cumul, cov(shares.list.cumul[[j]],
                                  index.list.cumul[[j]])/var(shares.list.cumul[[j]]))
}
return.data <- return.data %>% add_column(return.beta = beta.cumul)
ggplot() +
  geom_line(data = return.data, aes(date, return.beta), color = 'blue', size = 1) +
  labs(title = paste0('MS vs ^GSPC Beta'),
       undertitle = 'Cumulative standard deviations over the year',
       x = 'date', y = 'Beta')
```




In this case, MS would tend to indicate less systemic risk than the Index we are tracking.

**Alpha**

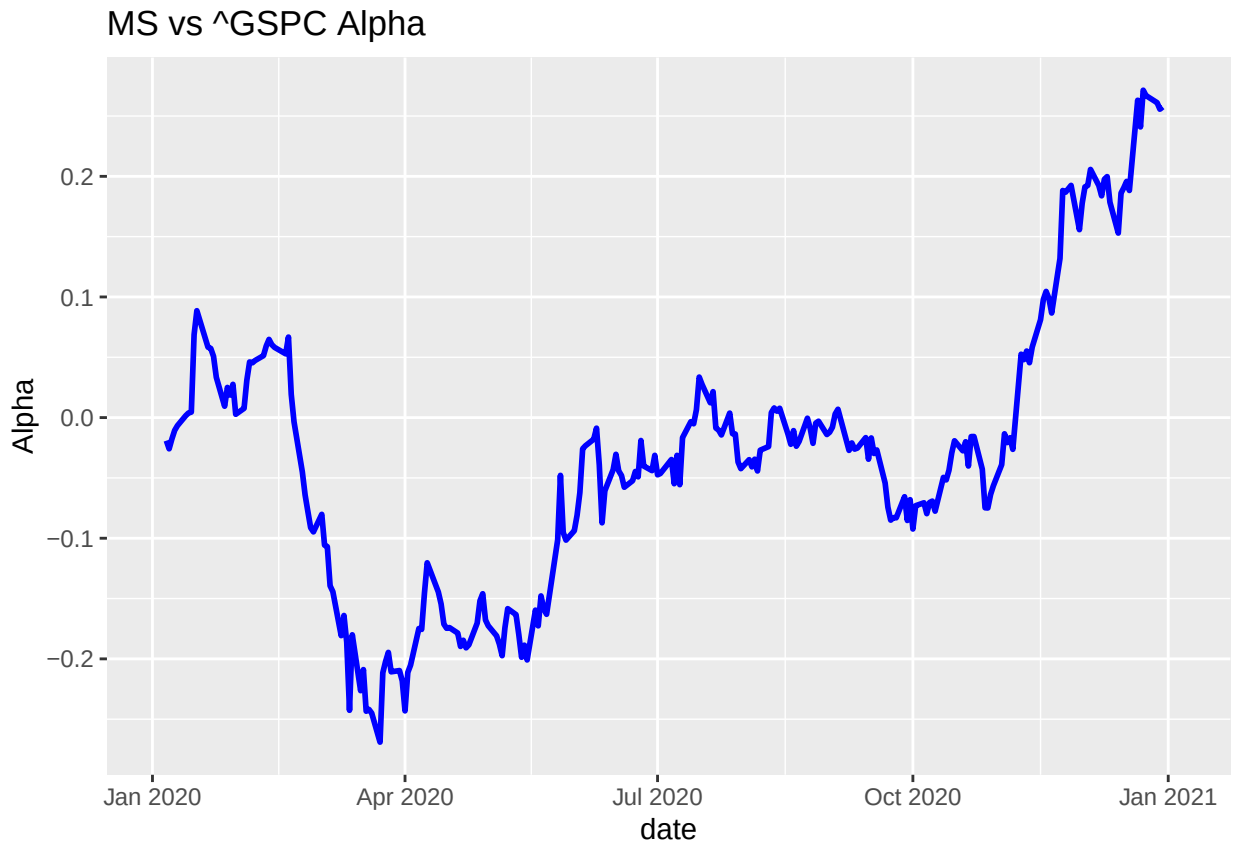
Let us check another well known financing metric: Alpha, which measures the excess return of a particular equity. We mention excess return, because Alpha takes into consideration the equity's return, the index/market return, as well as the guaranteed return rate of the market. For this guaranteed return rate, we simply scrape from the US treasury website at https://www.treasury.gov/resource-center/data-chart-center/interest-rates/pages/TextView.aspx?data=yieldYear&year=2020, the guaranteed return from a yearly treasury bill on every market open date of 2020 so far.

```
url <- 'https://www.treasury.gov/resource-center/data-chart-center/interest-rates/pages/TextView.aspx?da
webpage <- read_html(url)
tbill.yield <- webpage %>% html_nodes('.text_view_data:nth-child(6)') %>% html_text()
tbill.yield <- as.numeric(tbill.yield)/100
```

Applying the calculations for Alpha:

```
alpha.cumul <- c()
for (j in 1:length(shares.daily.return)) {
  alpha.cumul <- c(alpha.cumul, shares.daily.return[j] -
                     (tbill.yield[j] + beta.cumul[j]*(index.daily.return[j] -
                                                        tbill.yield[j])))
}
return.data <- return.data %>% add_column(return.alpha = alpha.cumul)
ggplot() +
  geom_line(data = return.data, aes(date, return.alpha), color = 'blue', size = 1) +
  labs(title = paste0('MS vs ^GSPC Alpha'),
```

```
       undertitle = 'Cumulative standard deviations over the year',
       x = 'date', y = 'Alpha')
```

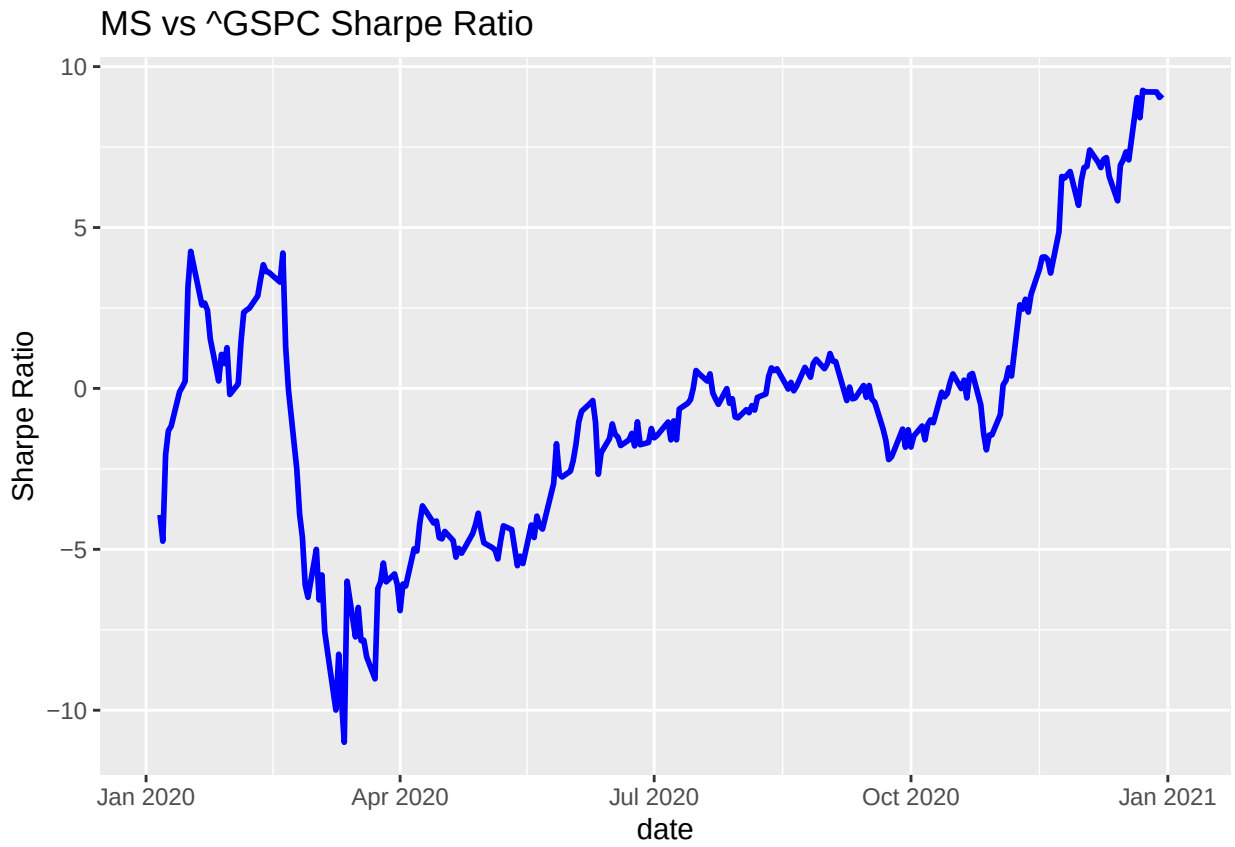



Thus, we see that MS has performed well in this segment as well.

### Sharpe Ratio

Another financial metric to look through is the Sharpe Ratio (as well the very closely related Sortino and Treynor Ratios, which we will see right after). It measures the volatility of the excess return of an equity:

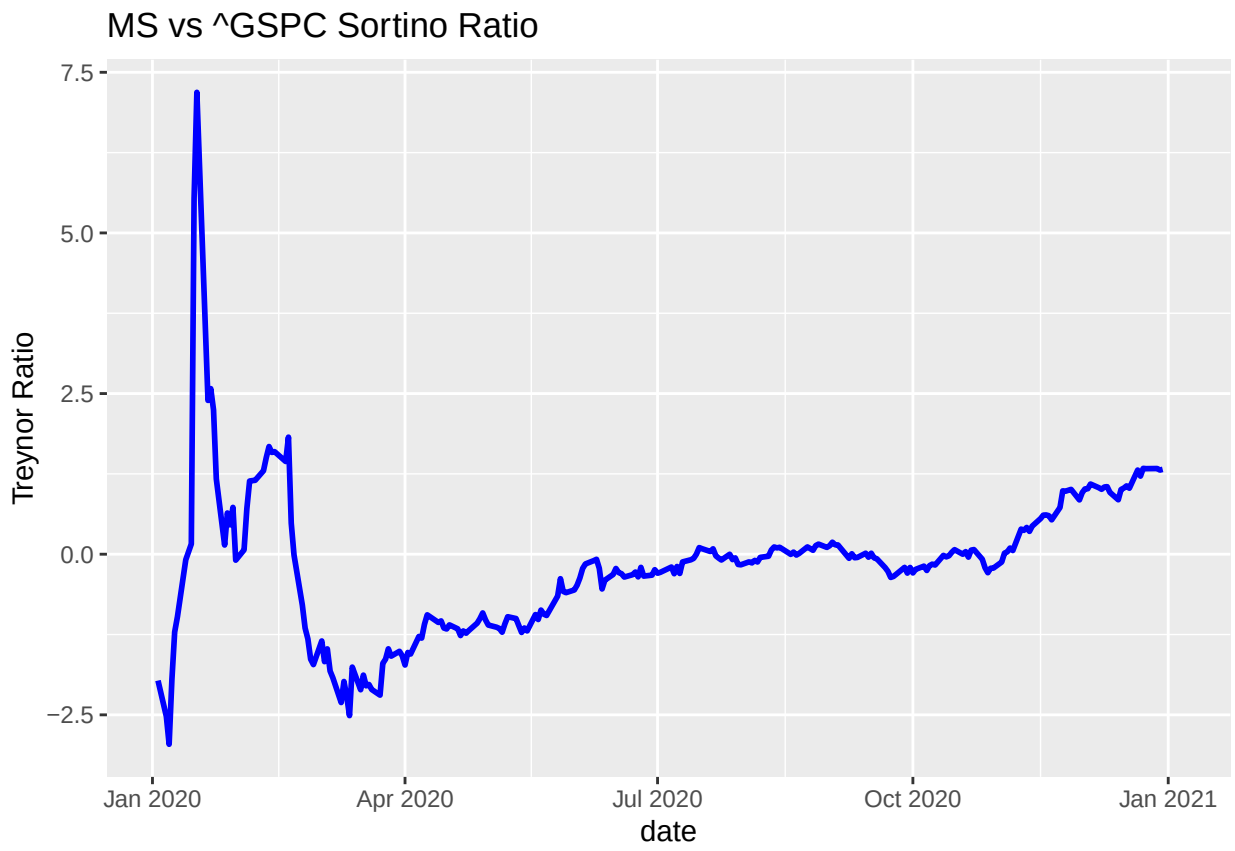
```
sharpe.ratio.cumul <- c()
for (j in 1:length(shares.daily.return)) {
  sharpe.ratio.cumul <- c(sharpe.ratio.cumul, (shares.daily.return[j] -
                                                 tbill.yield[j])/sqrt(
                                                   var(shares.list.cumul[[j]] -
                                                         tbill.yield[j])))
}
return.data <- return.data %>% add_column(return.sharpe.ratio = sharpe.ratio.cumul)
ggplot() +
  geom_line(data = return.data, aes(date, return.sharpe.ratio), color = 'blue',
            size = 1) +
  labs(title = paste0('MS vs ^GSPC Sharpe Ratio'),
       undertitle = 'Cumulative standard deviations over the year',
       x = 'date', y = 'Sharpe Ratio')
```




### Sortino Ratio

The Sortino Ratio is a twist on the above Sharpe Ratio; instead of looking at the excess return variation, Sortino Ratio looks at the excess return rate based on the equity's Down Deviation (i.e. when the returns are negative).

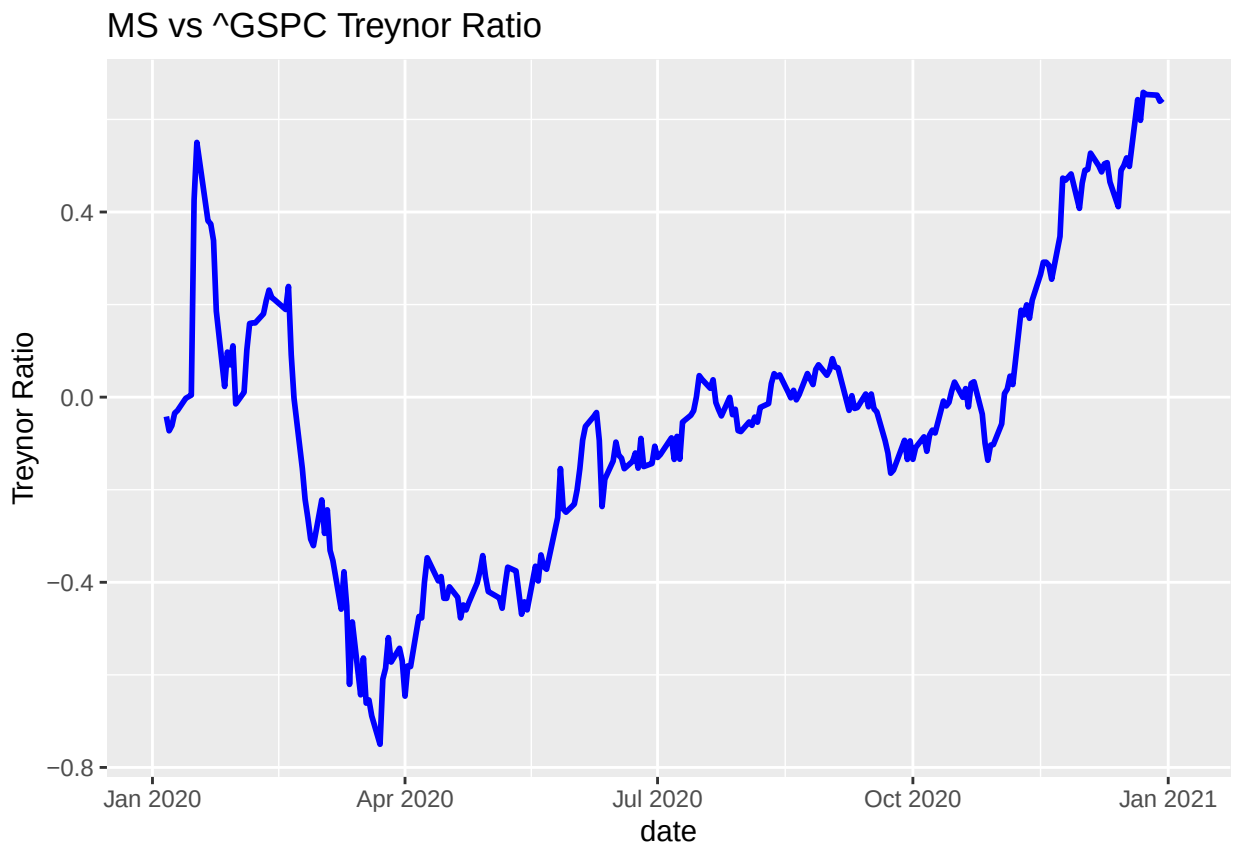
```
sortino.ratio.cumul <- c()
for (j in 1:length(shares.daily.return)) {
  sortino.ratio.cumul <- c(sortino.ratio.cumul, ((shares.daily.return[j] -
                                                    tbill.yield[j])/
                                                  shares.return.down.dev.cumul[j]))
}
return.data <- return.data %>% add_column(return.sortino.ratio = sortino.ratio.cumul)
ggplot() +
  geom_line(data = return.data, aes(date, return.sortino.ratio), color = 'blue',
            size = 1) +
  labs(title = paste0('MS vs ^GSPC Sortino Ratio'),
       undertitle = 'Cumulative standard deviations over the year',
       x = 'date', y = 'Treynor Ratio')
```




**Treynor Ratio**

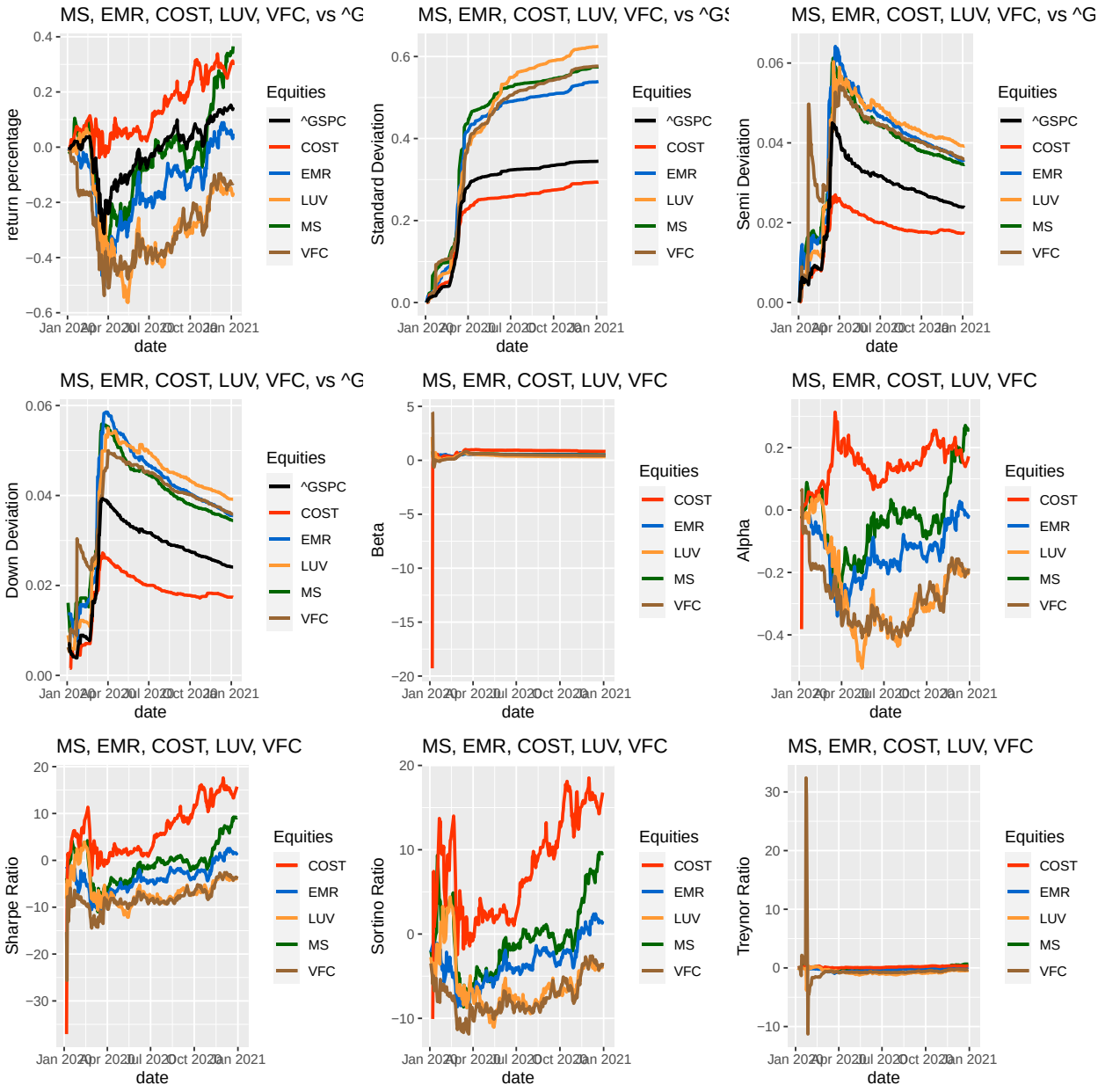
In the case of the Treynor Ratio, it is also similar to the Sharpe Ratio, but uses equity Beta to adjust the returns, compared to excess return standard deviation for Sharpe.

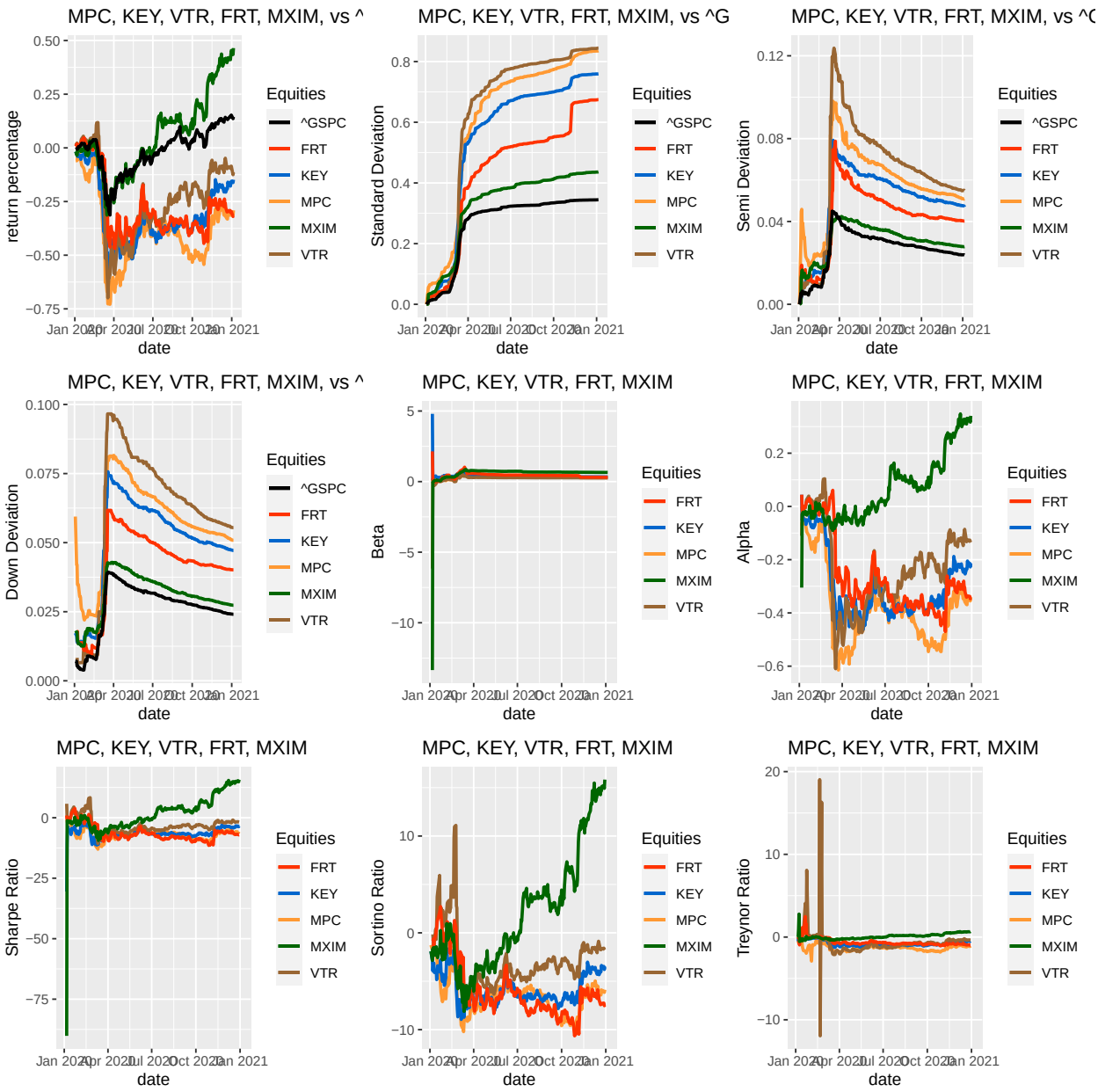
```
treynor.ratio.cumul <- c()
for (j in 1:length(shares.daily.return)) {
  treynor.ratio.cumul <- c(treynor.ratio.cumul, (shares.daily.return[j] -
                                                   tbill.yield[j])/beta.cumul[j])
}
return.data <- return.data %>% add_column(return.treynor.ratio = treynor.ratio.cumul)
ggplot() +
  geom_line(data = return.data, aes(date, return.treynor.ratio), color = 'blue',
            size = 1) +
  labs(title = paste0('MS vs ^GSPC Treynor Ratio'),
       undertitle = 'Cumulative standard deviations over the year',
       x = 'date', y = 'Treynor Ratio')
```

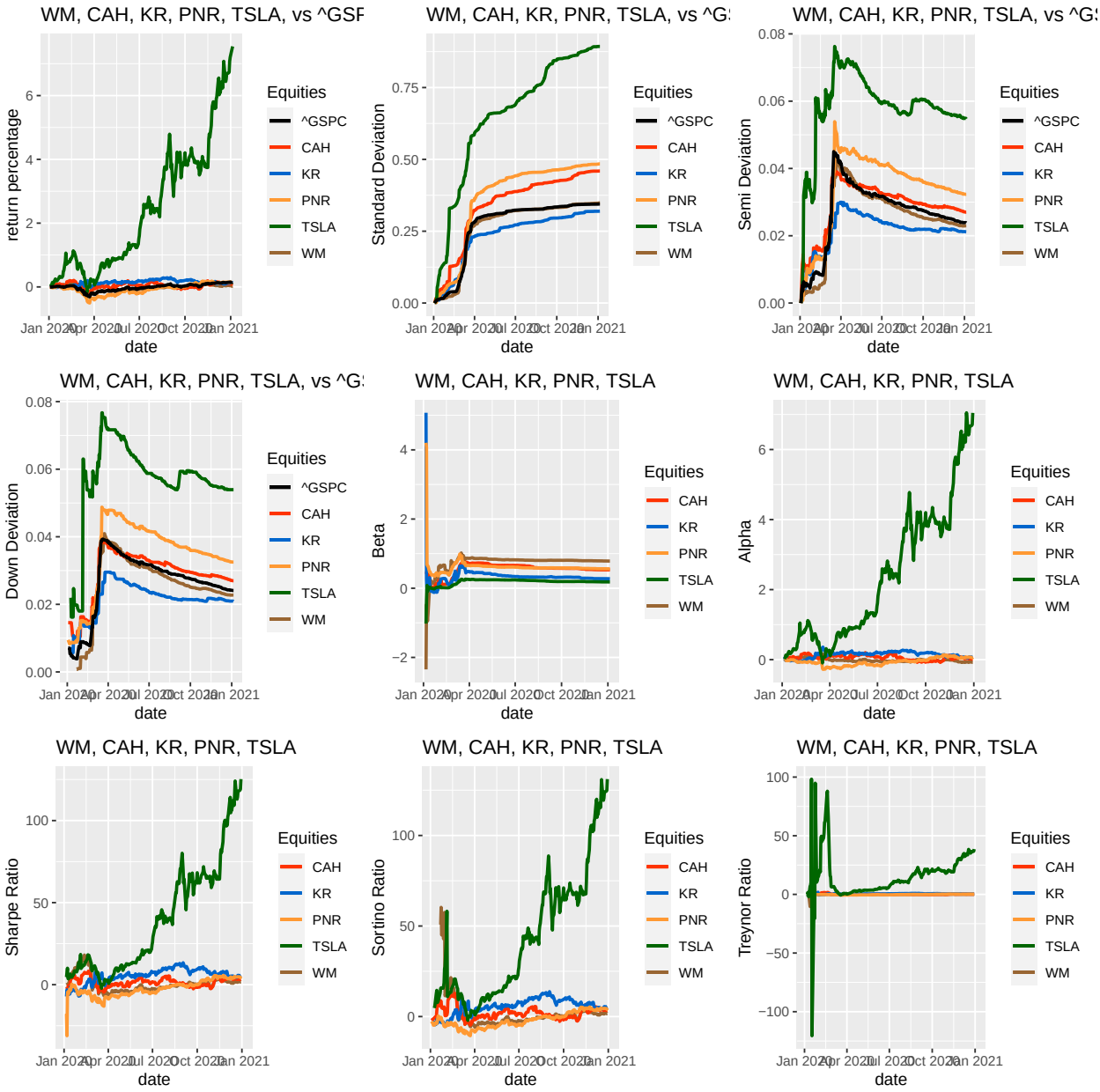


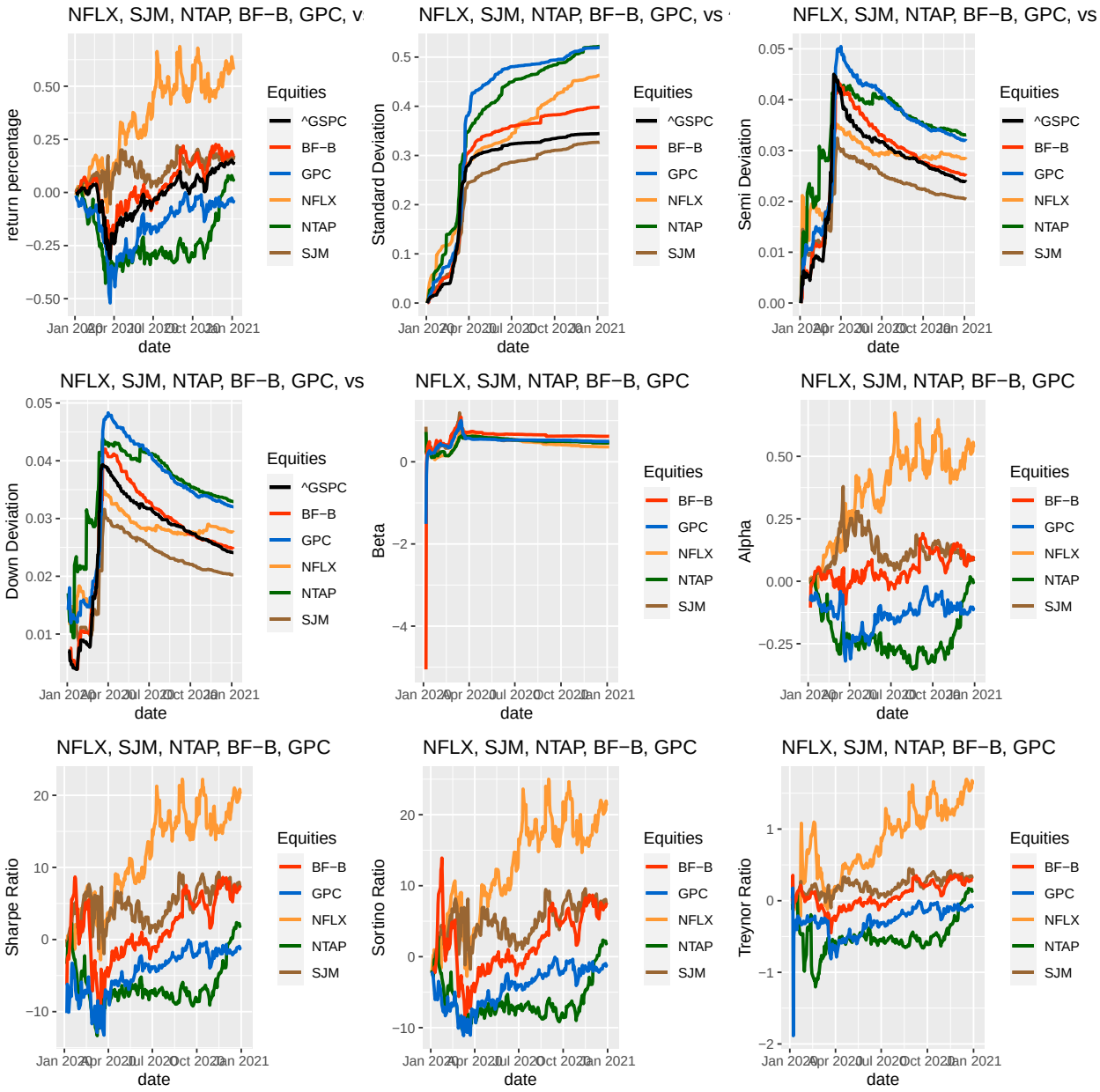

### Second Analysis (Other Single Random Sample Equities)

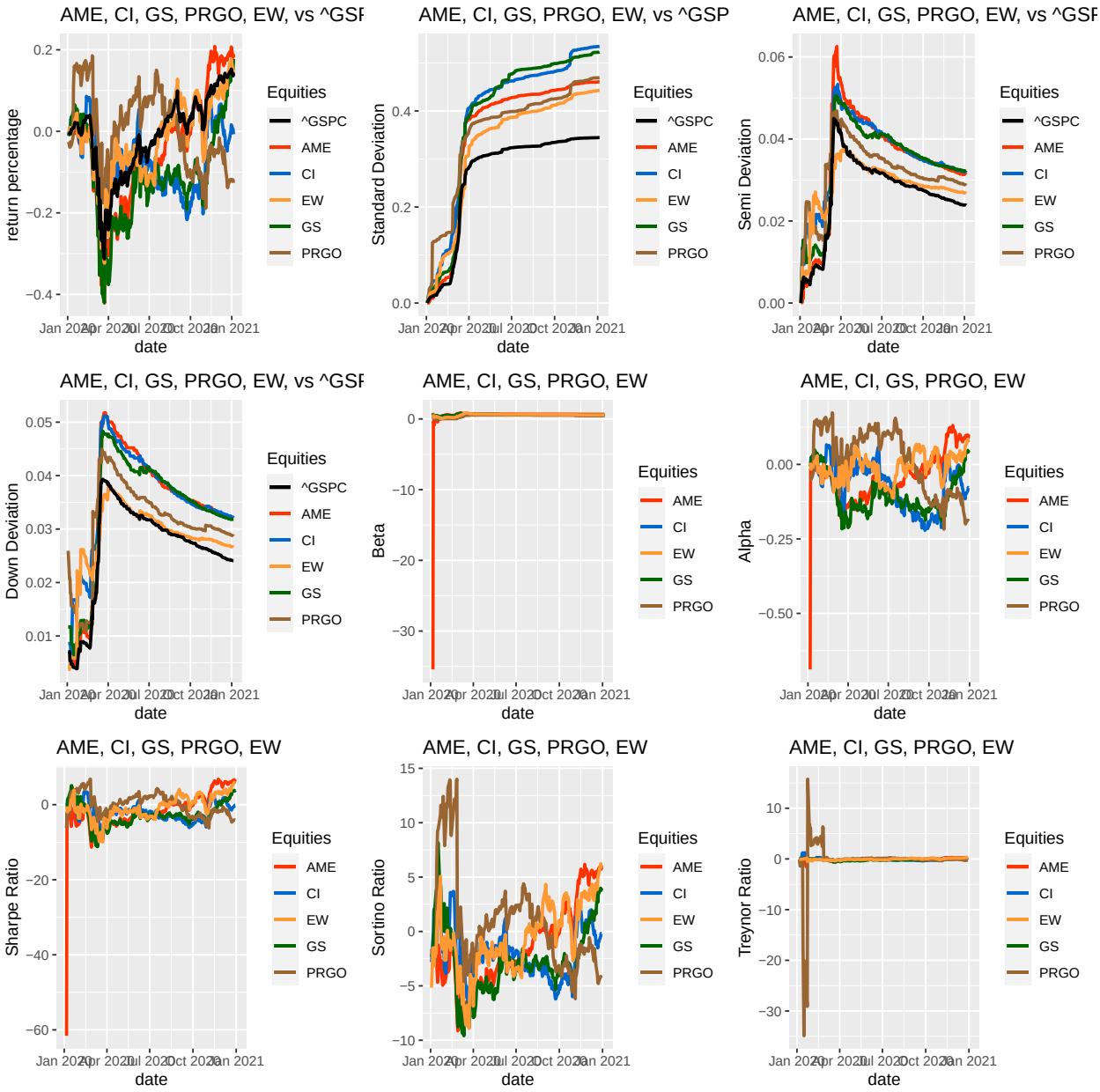
We know apply this to other Equities chosen from our Random Sampling Method. To save time and space, Equities will be evaluated and plotted 5 at a time. Again, in an effort to streamline the report, and since we are basically reapplying previous code through multiple different equities at the same time, the code will be hidden on the report for this section (note that it is however still available in the core code file on github).


MS, EMR, COST, LUV, VFC, vs ^G
return percentage
date
Equities
^GSPC
COST
EMR
LUV
MS
VFC
MS, EMR, COST, LUV, VFC, vs ^GS
Standard Deviation
MS, EMR, COST, LUV, VFC, vs ^G
Semi Deviation
MS, EMR, COST, LUV, VFC, vs ^G
Down Deviation
MS, EMR, COST, LUV, VFC
Beta
MS, EMR, COST, LUV, VFC
Alpha
MS, EMR, COST, LUV, VFC
Sharpe Ratio
MS, EMR, COST, LUV, VFC
Sortino Ratio
MS, EMR, COST, LUV, VFC
Treynor Ratio


MPC, KEY, VTR, FRT, MXIM, vs ^
return percentage
0.50
0.25
0.00
−0.25
−0.50
−0.75
Jan 2020
Apr 2020
Jul 2020
Oct 2020
Jan 2021
date
Equities
^GSPC
FRT
KEY
MPC
MXIM
VTR
MPC, KEY, VTR, FRT, MXIM, vs ^G
Standard Deviation
0.8
0.6
0.4
0.2
0.0
MPC, KEY, VTR, FRT, MXIM, vs ^C
Semi Deviation
0.12
0.08
0.04
0.00
MPC, KEY, VTR, FRT, MXIM, vs ^
Down Deviation
0.100
0.075
0.050
0.025
0.000
MPC, KEY, VTR, FRT, MXIM
Beta
5
0
−5
−10
Alpha
0.2
0.0
−0.2
−0.4
−0.6
MPC, KEY, VTR, FRT, MXIM
Sharpe Ratio
0
−25
−50
−75
MPC, KEY, VTR, FRT, MXIM
Sortino Ratio
10
0
−10
MPC, KEY, VTR, FRT, MXIM
Treynor Ratio
20
10
0
−10


WM, CAH, KR, PNR, TSLA, vs ^GSP
return percentage
date
Equities
^GSPC
CAH
KR
PNR
TSLA
WM
WM, CAH, KR, PNR, TSLA, vs ^GS
Standard Deviation
WM, CAH, KR, PNR, TSLA, vs ^GS
Semi Deviation
WM, CAH, KR, PNR, TSLA, vs ^GS
Down Deviation
WM, CAH, KR, PNR, TSLA
Beta
WM, CAH, KR, PNR, TSLA
Alpha
WM, CAH, KR, PNR, TSLA
Sharpe Ratio
WM, CAH, KR, PNR, TSLA
Sortino Ratio
WM, CAH, KR, PNR, TSLA
Treynor Ratio


NFLX, SJM, NTAP, BF−B, GPC, v
return percentage
0.50
0.25
0.00
−0.25
−0.50
Jan 2020
Apr 2020
Jul 2020
Oct 2020
Jan 2021
date
Equities
^GSPC
BF−B
GPC
NFLX
NTAP
SJM
NFLX, SJM, NTAP, BF−B, GPC, vs
Standard Deviation
0.5
0.4
0.3
0.2
0.1
0.0
NFLX, SJM, NTAP, BF−B, GPC, vs
Semi Deviation
0.05
0.04
0.03
0.02
0.01
0.00
NFLX, SJM, NTAP, BF−B, GPC, vs
Down Deviation
NFLX, SJM, NTAP, BF−B, GPC
Beta
0
−2
−4
NFLX, SJM, NTAP, BF−B, GPC
Alpha
0.50
0.25
0.00
−0.25
NFLX, SJM, NTAP, BF−B, GPC
Sharpe Ratio
20
10
0
−10
NFLX, SJM, NTAP, BF−B, GPC
Sortino Ratio
NFLX, SJM, NTAP, BF−B, GPC
Treynor Ratio
1
0
−1
−2




At this point, we see that some equities have outpaced the index, some have not, some have but only before *or* after the March 2020 COVID Crash. However, just picking a stock at random is simple pure gambling (as we would be picking a stock at random out 501 possible choices here). Instead, what if we decided to pick 50 stocks at random in an attempt to diversify the investment portfolio ? (This will be done in a subsequent report, to be distributed at a later date).