Optimizing a Stock Portfolio in R

From a Diversified Portfolio Based on 4 Sample Stocks

Leon Shpaner

2020-08-26

In R Studio, we will utilize a classic stock portfolio optimization approach to examine the following 4 ticker symbols, Tejon Ranch Company (TRC), Snapchat (SNAP), AI Powered Equity ETF (AIEQ), and 2U (TWOU). These are pretty well-diversified stocks from a sample portfolio, so the data presented herein is true, significant, and will yield actionable insights.

We start by loading the required libraries below (quantmod (the stock package), quadprog (for the quadratic program), the ROI plugin and subsequent package in order to gauge the return on investment). We also load the magrittr package "to decrease development time and improve readability" and the broom package to "bridge the gap from untidy outputs of predictions and estimations to the tidy data we want to work with."

```
library(quantmod)
library(quadprog)
library(ROI.plugin.quadprog)
library(ROI)
library(ggplot2)
library(magrittr)
library(broom)
```

Next, we pick the desired beginning and ending date range for the stock prices (notice how we leave the end date blank, as we want to model this through the current date:

```
begin_date <-"2013-01-01"
end_date <- "2020-08-04"
```

Now we simply fill in the following vector (tickers) with the 4 stocks we have specified above, and tell the code to run the function getSymbols from begin_date to end_date time period previously specified above. Lastly, the argument `auto.assign = TRUE` allows the data set to automatically be assigned a name.

```
tickers<-c("TRC","SNAP","AIEQ","TWOU")
getSymbols(tickers,from = begin_date, to = end_date, auto.assign = TRUE)
```

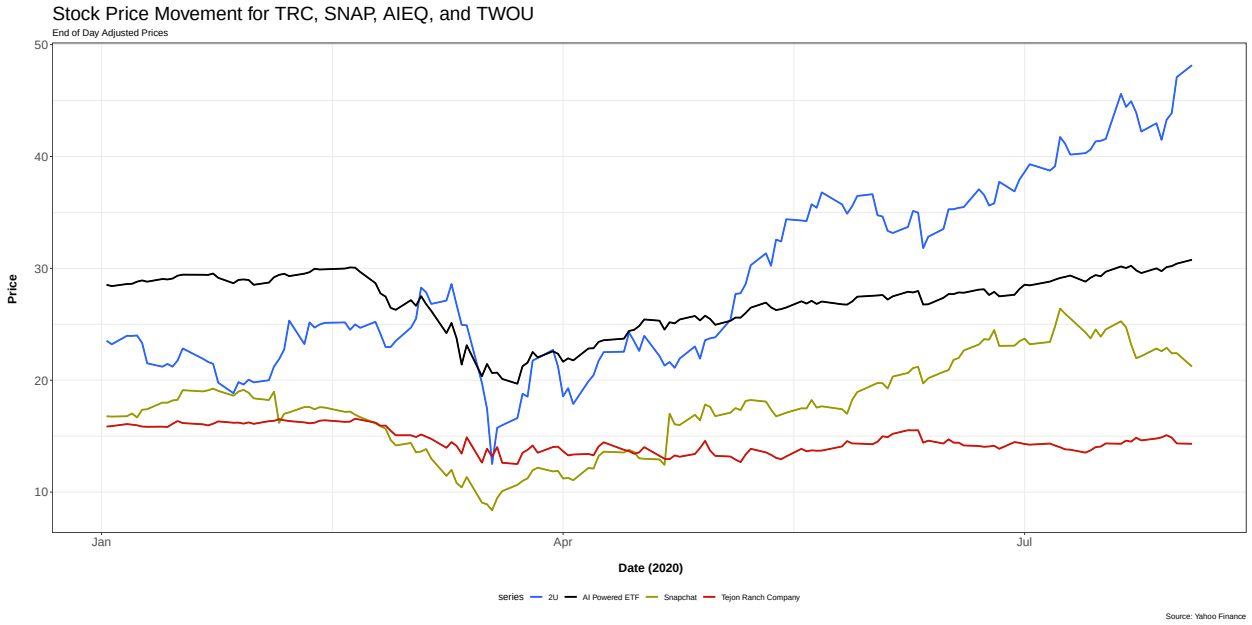
```
## [1] "TRC"  "SNAP" "AIEQ" "TWOU"
```

Adding a visual component visa vie line plot for stock prices will certainly add value to this analysis. For example, from the graph below, we can immediately see that 2U outperforms all other stocks in the long run.

```
options(repr.plot.width = 1, repr.plot.height = 8)
knitr::opts_chunk$set(fig.width = 25, fig.height = 8)
start = as.Date("2020-01-01")
end = as.Date("2020-08-04")
getSymbols(c("TRC", "SNAP", "AIEQ", "TWOU"), src = "yahoo", from = start, to = end)
```

```
## [1] "TRC"  "SNAP" "AIEQ" "TWOU"
```

```
stocks = as.xts(data.frame(TRC = TRC[, "TRC.Adjusted"], SNAP = SNAP[, "SNAP.Adjusted"],
                           AIEQ = AIEQ[, "AIEQ.Adjusted"], TWOU = TWOU[, "TWOU.Adjusted"]))
names(stocks) = c("Tejon Ranch Company", "Snapchat", "AI Powered ETF", "2U")
index(stocks) = as.Date(index(stocks))
stocks_series = tidy(stocks) %>%
  ggplot(aes(x=index,y=value, color=series)) + geom_line()+
  theme_bw()+
  theme(legend.position="bottom")+
  theme(axis.text = element_text(size = 14),
        plot.title = element_text(size = 22),
        axis.title = element_text(size = 14,face = "bold"))+
  labs(title = "Stock Price Movement for TRC, SNAP, AIEQ, and TWOU",
       element_text(size=22, face = "bold"),
            subtitle = "End of Day Adjusted Prices",
            caption = " Source: Yahoo Finance") +
  xlab("\n Date (2020)") + ylab("Price \n")+
  scale_color_manual(values = c("#2a63ff", "#000000", "#999900", "#cd1409",
                                alpha = 1, linetype = 1)) + geom_line(size=1)
stocks_series
```



This will ultimately allow us to plug the value of our stocks at their closing period into the object close, using the cbind syntax, binding the data into columns. Then we load the close prices into the vector of returns, thereby looking at returns as a function of close prices.

```
close<-cbind(TRC$TRC.Close, SNAP$SNAP.Close, AIEQ$AIEQ.Close, TWOU$TWOU.Close)
View(close) #look at close to make sure your stock data was input correctly
close<-close[complete.cases(close), ]
returns <- (close/lag(close) - 1)[-1]
head(returns)
```

```
##               TRC.Close   SNAP.Close    AIEQ.Close    TWOU.Close
## 2020-01-03  0.002522068 -0.001787902 -0.0041841353 -0.0135997444
## 2020-01-06  0.011320755  0.001791104  0.0066526963  0.0327445081
## 2020-01-07 -0.003109391  0.014898688  0.0006956522 -0.0004171882
```

```
## 2020-01-08 -0.003743044 -0.021139224  0.0066040667  0.0016694909
## 2020-01-09 -0.006887915  0.041391782  0.0037983772 -0.0279166667
## 2020-01-10 -0.001261034  0.002880127 -0.0037840041 -0.0780111444
```

Taking this a step further, we examine the average returns(mu), covariance matrix (cov), standard deviation (s), and minimum value (min) of our returns vector as shown below:

```
mu<-colMeans(returns)
s<-cov(returns)
s
```

```
##               TRC.Close   SNAP.Close   AIEQ.Close   TWOU.Close
## TRC.Close  0.0009320552 0.0005692155 0.0005085904 0.0006068748
## SNAP.Close 0.0005692155 0.0027714389 0.0006990460 0.0013600576
## AIEQ.Close 0.0005085904 0.0006990460 0.0005714617 0.0007277273
## TWOU.Close 0.0006068748 0.0013600576 0.0007277273 0.0034830722
```

```
mu
```

```
##     TRC.Close    SNAP.Close    AIEQ.Close    TWOU.Close
## -0.0002256408  0.0029149139  0.0007681058  0.0066618018
```

```
min(mu)
```

```
## [1] -0.0002256408
```

```
min(returns)
```

```
## [1] -0.2835051
```

As in any optimization problem, we must define the constraints. As shown immediately above, our minimum portfolio return is -0.6493151. Now, since we are only looking at 4 stocks, our ensuing script needs to reflect that of a 2x4 matrix. We set these up as follows, binding into rows this (with rbind), and plugging in the value for our minimum portfolio return:

```
sum1<-matrix(c(1,1,1,1),ncol=4)
constr<-rbind(mu,sum1)
constr
```

```
##        TRC.Close  SNAP.Close   AIEQ.Close  TWOU.Close
## mu -0.0002256408 0.002914914 0.0007681058 0.006661802
##     1.0000000000 1.000000000 1.0000000000 1.000000000
```

```
minreturn <- -0.6493151
```

As shown above, the vector of ones is used to sum up the allocations (our decisions). The average return of the porfolio, mu, will need to be greater than or equal to the minimum return value of -0.6493151 that we have just calculated.

Finally, we proceed to plug in the parameters for the quadratic program by placing them into the QP data frame, thereby allowing us to run the solution and ensuing ROI as follows:

```
QP <- OP(Q_objective(Q=2*s,
                     L=c(0,0,0,0)),
         L_constraint(L=constr,
                      dir=c(">=","=="),
                      rhs=c(minreturn,1)),
         maximum=FALSE)

QP
```

```
## ROI Optimization Problem:
```

```
##
## Minimize a quadratic objective function of length 4 with
## - 4 continuous objective variables,
##
## subject to
## - 2 constraints of type linear.
## - 0 lower and 0 upper non-standard variable bounds.
```

```
# run and view answer!
sol <- ROI_solve(QP, solver = "quadprog")
sol$solution
```

```
## [1] 1.292754e-01 3.310385e-18 8.707246e-01 0.000000e+00
```

```
sol$solution%*%mu
```

```
##              [,1]
## [1,] 0.0006396389
```