Portfolio Optimization using R and Plotly

Published April 3, 2016 by Riddhiman in Business Intelligence, Data Visualization, R

In this post we'll focus on showcasing Plotly's **WebGL** capabilities by charting financial portfolios using an R package called **PortfolioAnalytics**. The package is a generic portfolo optimization framework developed by folks at the **University of Washington** and *Brian Peterson* (of the PerformanceAnalytics fame).

You can see the vignette here

Let's pull in some data first.

```
library(PortfolioAnalytics)
library(quantmod)
library(PerformanceAnalytics)
library(zoo)
library(plotly)

# Get data
getSymbols(c("MSFT", "SBUX", "IBM", "AAPL", "^GSPC", "AMZN"))

# Assign to dataframe
# Get adjusted prices
prices.data <- merge.zoo(MSFT[,6], SBUX[,6], IBM[,6], AAPL[,6], GSPC[,6], AMZN[,6])

# Calculate returns
returns.data <- CalculateReturns(prices.data)
returns.data <- na.omit(returns.data)

# Set names
colnames(returns.data) <- c("MSFT", "SBUX", "IBM", "AAPL", "^GSPC", "AMZN")

# Save mean return vector and sample covariance matrix
meanReturns <- colMeans(returns.data)
covMat <- cov(returns.data)
```

Now that we have some data, let's get started by creating a portfolio specification. This can be done by using `portfolio.spec()`

```
# Start with the names of the assets
port <- portfolio.spec(assets = c("MSFT", "SBUX", "IBM", "AAPL", "^GSPC", "AMZN"))
```

Now for some constraints. Let's use the following:

- Box constraints
- Leverage (weight sum)

```
# Box
port <- add.constraint(port, type = "box", min = 0.05, max = 0.8)

# Leverage
port <- add.constraint(portfolio = port, type = "full_investment")
```

Let's use the built-in **random** solver. This essentially creates a set of feasible portfolios that satisfy all the constraints we have specified. For a full list of supported constraints see here

```r
# Generate random portfolios
rportfolios <- random_portfolios(port, permutations = 500000, rp_method = "sample")
```

Now let's add some objectives and optimize. For simplicity's sake let's do some mean-variance optimization.

```r
# Get minimum variance portfolio
minvar.port <- add.objective(port, type = "risk", name = "var")

# Optimize
minvar.opt <- optimize.portfolio(returns.data, minvar.port, optimize_method = "random",
                                 rp = rportfolios)

# Generate maximum return portfolio
maxret.port <- add.objective(port, type = "return", name = "mean")

# Optimize
maxret.opt <- optimize.portfolio(returns.data, maxret.port, optimize_method = "random",
                                 rp = rportfolios)

# Generate vector of returns
minret <- 0.06/100
maxret <- maxret.opt$weights %*% meanReturns

vec <- seq(minret, maxret, length.out = 100)
```

Now that we have the minimum variance as well as the maximum return portfolios, we can build out the efficient frontier. Let's add a weight concentration objective as well to ensure we don't get highly concentrated portfolios.

Note:

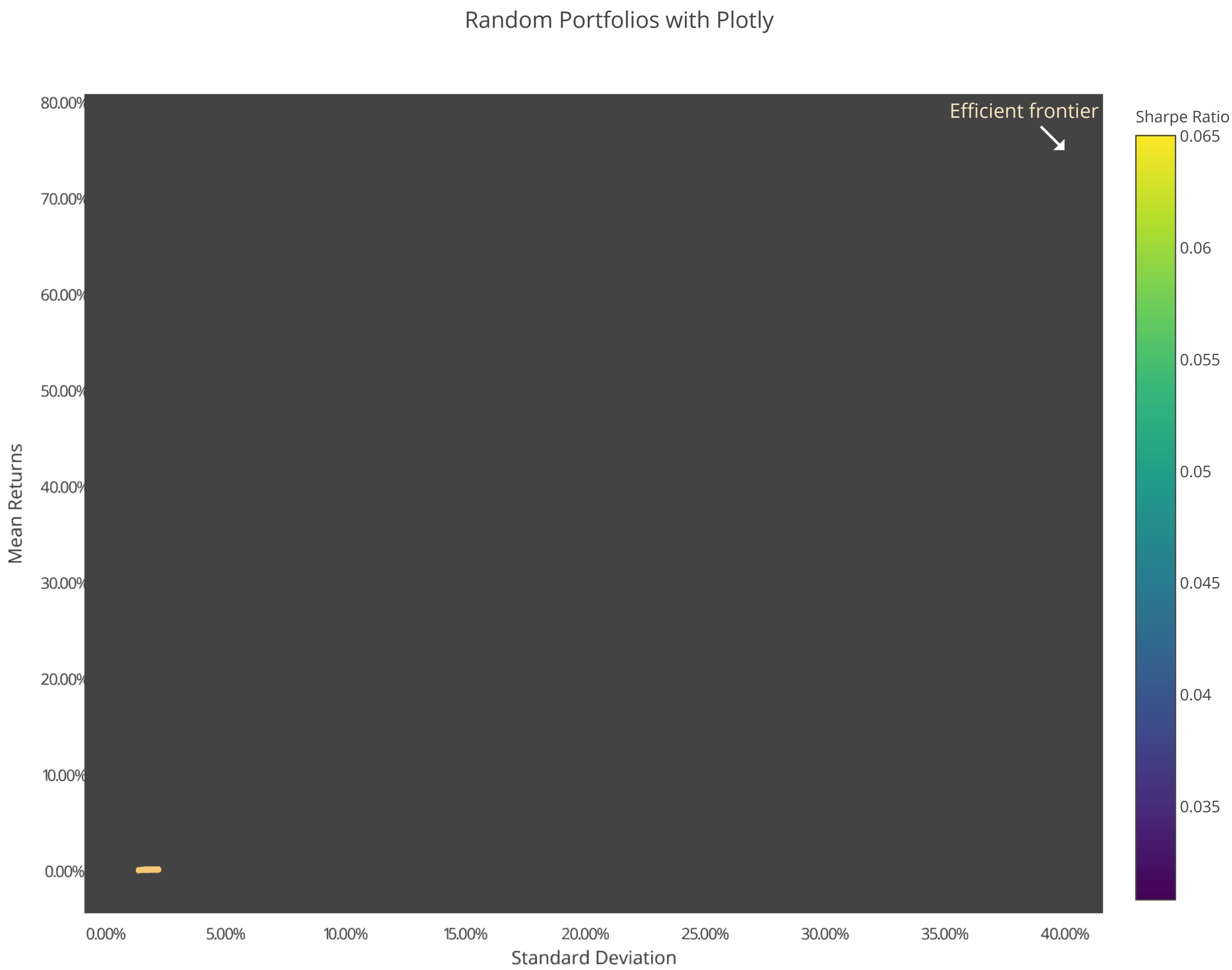
- `random_portfolios()` ignores any *diversification* constraints. Hence, we didn't add it previously.
- Using the **random** solver for each portfolio in the loop below would be very compute intensive. We'll use the **ROI** (R Optmization Infrastructure) solver instead.

```r
eff.frontier <- data.frame(Risk = rep(NA, length(vec)),
                           Return = rep(NA, length(vec)),
                           SharpeRatio = rep(NA, length(vec)))

frontier.weights <- mat.or.vec(nr = length(vec), nc = ncol(returns.data))
colnames(frontier.weights) <- colnames(returns.data)

for(i in 1:length(vec)){
  eff.port <- add.constraint(port, type = "return", name = "mean", return_target = vec[i])
  eff.port <- add.objective(eff.port, type = "risk", name = "var")
  # eff.port <- add.objective(eff.port, type = "weight_concentration", name = "HHI",
  #                           conc_aversion = 0.001)

  eff.port <- optimize.portfolio(returns.data, eff.port, optimize_method = "ROI")

  eff.frontier$Risk[i] <- sqrt(t(eff.port$weights) %*% covMat %*% eff.port$weights)

  eff.frontier$Return[i] <- eff.port$weights %*% meanReturns

  eff.frontier$Sharperatio[i] <- eff.port$Return[i] / eff.port$Risk[i]

  frontier.weights[i,] = eff.port$weights

  print(paste(round(i/length(vec) * 100, 0), "% done..."))
}
```

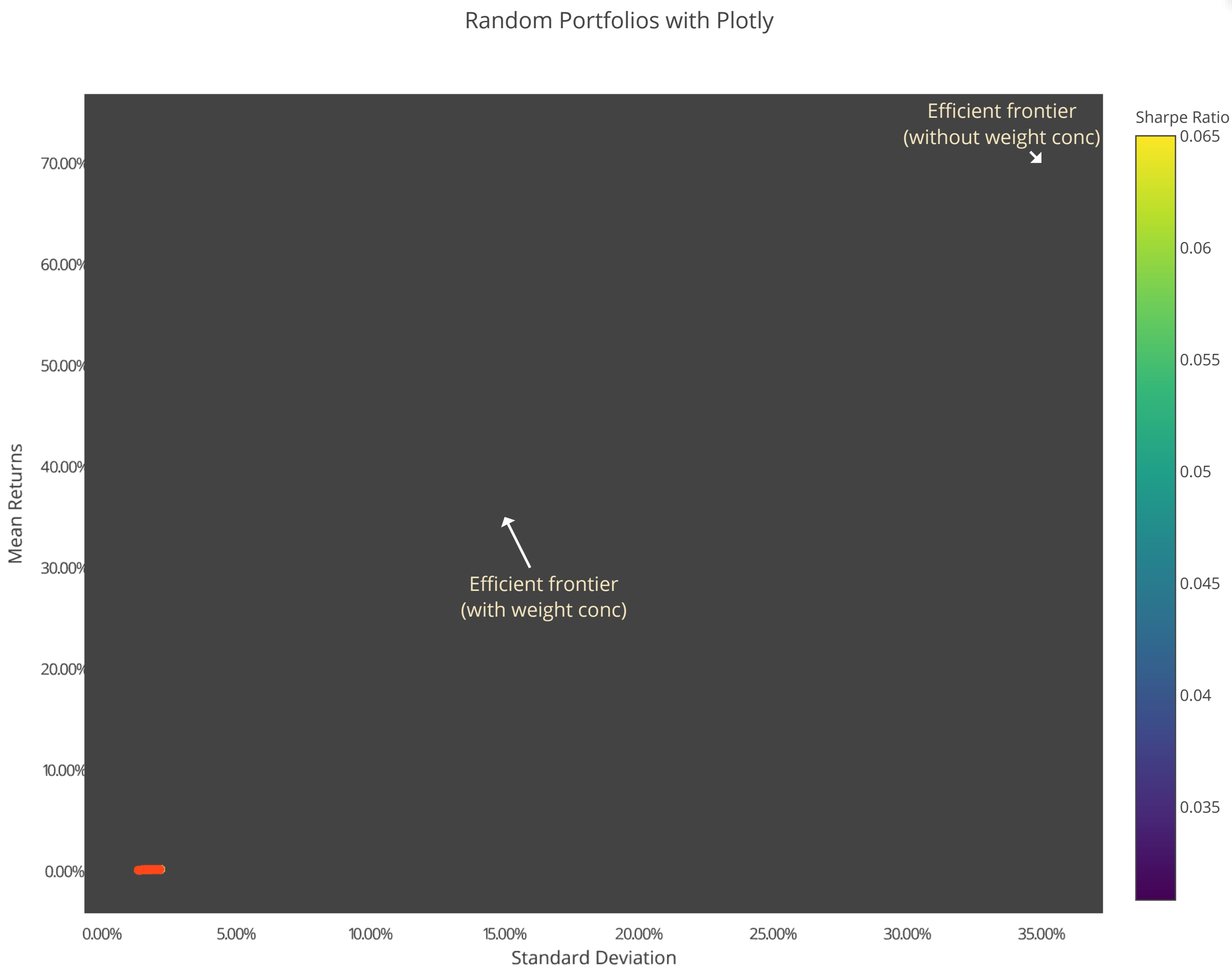
Now lets plot !

```r
feasible.sd <- apply(rportfolios, 1, function(x){
  return(sqrt(matrix(x, nrow = 1) %*% covMat %*% matrix(x, ncol = 1)))
```

```
})

feasible.means <- apply(rportfolios, 1, function(x){
  return(x %*% meanReturns)
})

feasible.sr <- feasible.means / feasible.sd

p <- plot_ly(x = feasible.sd, y = feasible.means, color = feasible.sr,
        mode = "markers", type = "scattergl", showlegend = F,

        marker = list(size = 3, opacity = 0.5,
                      colorbar = list(title = "Sharpe Ratio"))) %>%

  add_trace(data = eff.frontier, x = Risk, y = Return, mode = "markers",
            type = "scattergl", showlegend = F,
            marker = list(color = "#F7C873", size = 5)) %>%

  layout(title = "Random Portfolios with Plotly",
         yaxis = list(title = "Mean Returns", tickformat = ".2%"),
         xaxis = list(title = "Standard Deviation", tickformat = ".2%"),
         plot_bgcolor = "#434343",
         paper_bgcolor = "#F8F8F8",
         annotations = list(
           list(x = 0.4, y = 0.75,
                ax = -30, ay = -30,
                text = "Efficient frontier",
                font = list(color = "#F6E7C1", size = 15),
                arrowcolor = "white")
         ))
```

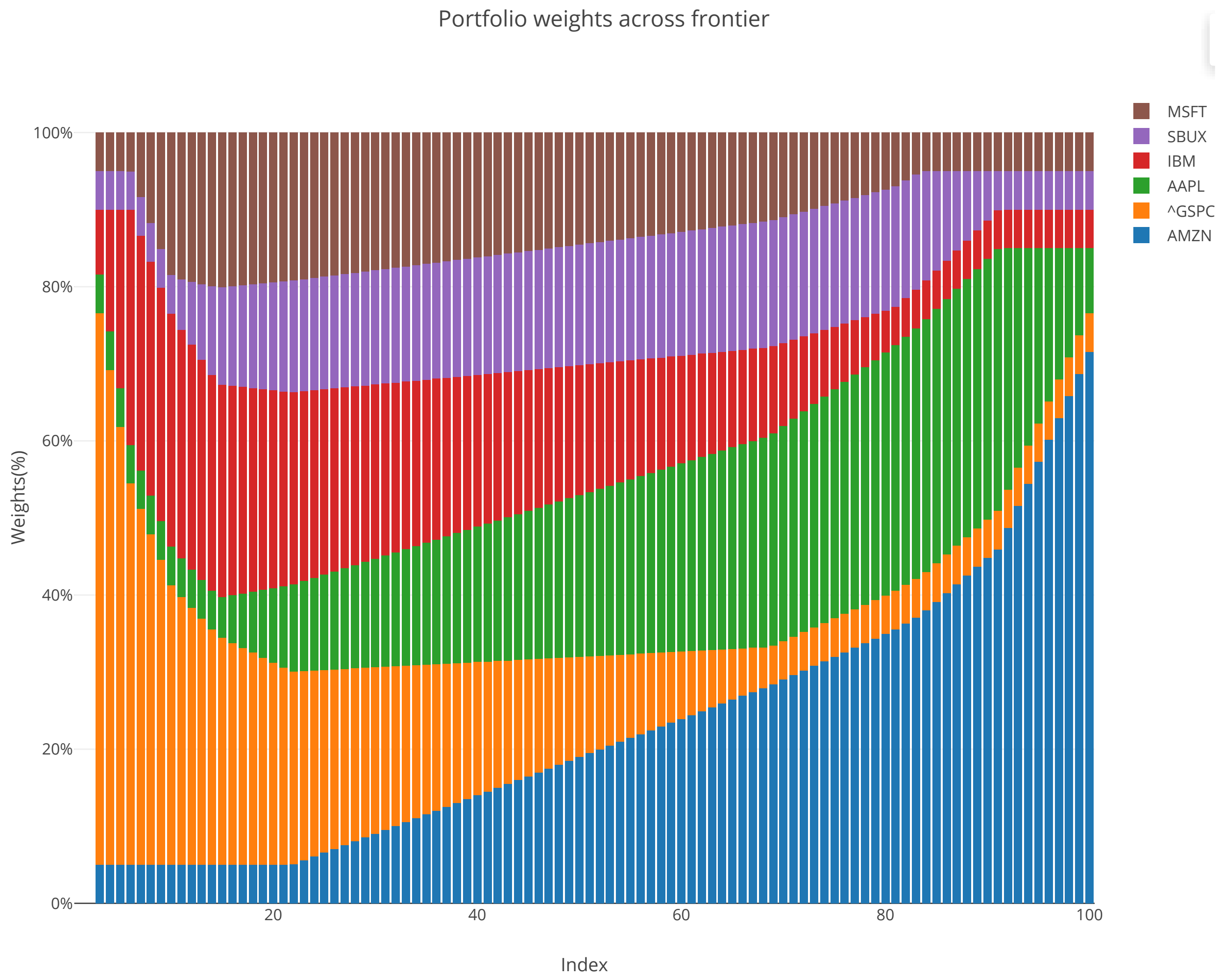


The chart above is plotting **42,749** data points ! Also, you'll notice that since the portfolios on the frontier(beige dots) have an added weight concentration objective, thefrontier seems sub optimal. Below is a comparison.



Let's also plot the weights to check how diversified our optimal portfolios are. We'll use a barchart for this.

```
frontier.weights.melt <- reshape2::melt(frontier.weights)

q <- plot_ly(frontier.weights.melt, x = Var1, y = value, group = Var2, type = "bar") %>%
  layout(title = "Portfolio weights across frontier", barmode = "stack",
         xaxis = list(title = "Index"),
         yaxis = list(title = "Weights(%)", tickformat = ".0%"))
```



44

[Riddhiman](#)

Tags: [Data Visualization](#), [Optimization](#), [Plotly](#), [R](#)

Post navigation

[Previous Post Eight Advantages of Python Over Matlab](#)
[Next Post Tufte style visualizations in R using Plotly](#)
Search for: Enter keyword and hit ente Search

Recent Posts

- [County-Level Choropleth in Plotly and R](#)
- [7 Interactive Bioinformatics Plots made in Python and R](#)
- [Creating interactive SVG tables in R](#)
- [Building rich analytic web apps has never been easier](#)
- [PLOTCON NYC: Edward Tufte, Dash workshop, Charts for React.js, and Discount for R-Bloggers](#)

Business Intelligence

- [Funnel Chart](#)
- [Lego Mini-Series](#)
- [All OSS used in Plotly Cloud and Plotly On-Premise](#)
- [News and Updates Surrounding plotly for R](#)
- [Native support for candlestick charts in Plotly and R](#)
- [Plotcon May 2017 – Speakers and Topics](#)
- [15 Python and R Charts with Interactive Controls: Buttons, Dropdowns, and Sliders](#)

Blog roll

[R-Bloggers](#)