Portfolio_Optimization

December, 2018

Data Import

First, we need to pull in our results from MySQL. These were stored from our python script in a table called portfolio.

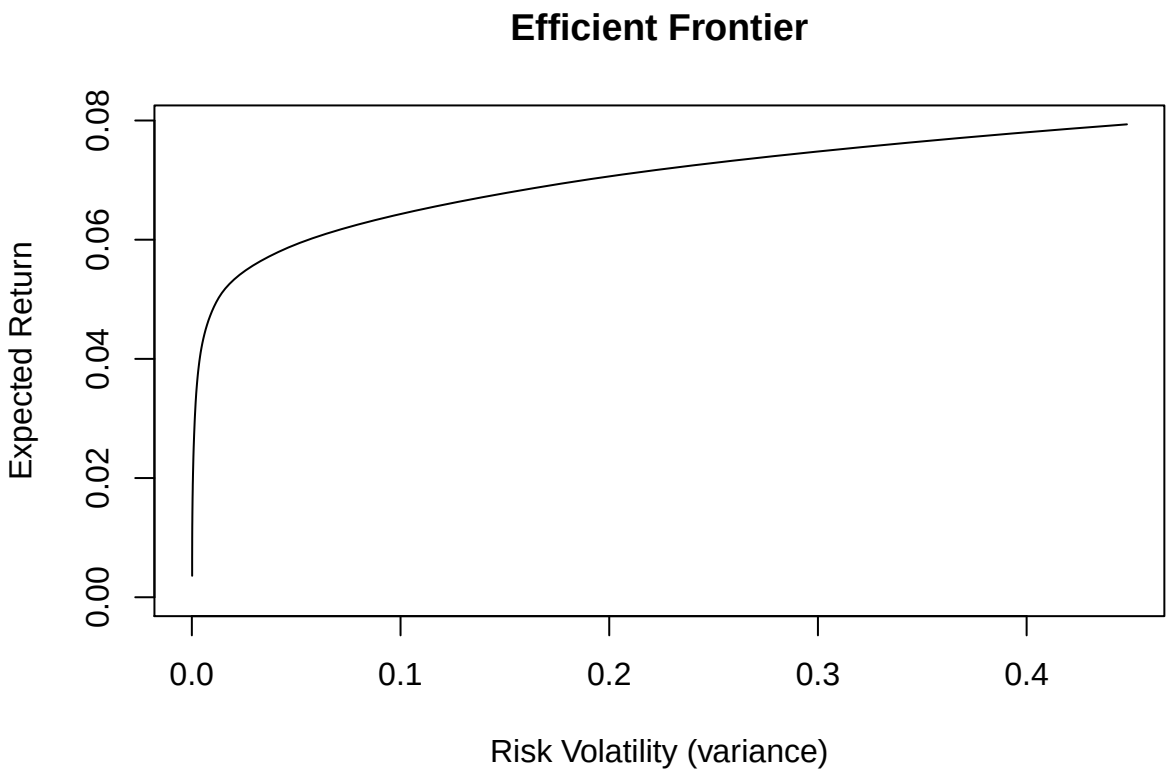
```
con <- dbConnect(MySQL(),
                 dbname = "nasdaq",
                 username = "root",
                 password = "1234")
data <- dbSendQuery(con,
                    'Select * from portfolio')
portfolio <- fetch(data)
head(portfolio)
```

```
##     expReturn      expRisk
## 1 0.003600000 0.0001392675
## 2 0.004365210 0.0001398997
## 3 0.005130419 0.0001417475
## 4 0.005895629 0.0001447402
## 5 0.006660838 0.0001488691
## 6 0.007426048 0.0001539989
```

Plotting the efficient frontier:

Next, we can plot. While efficiency frontiers are customarily plotted with standard deviation to give a bit of a smoother curve shape, for here we presented the variance since that is the calculation asked for in the assignment.

```
{plot(x = NA,
     y = NA,
     xlim = c(0, max(portfolio$expRisk)),
     ylim = c(0, max(portfolio$expReturn)),
     xlab = 'Risk Volatility (variance)',
     ylab = 'Expected Return',
     main = 'Efficient Frontier')
lines(x = portfolio$expRisk,
     y = portfolio$expReturn)}
```



Commentary:

Regarding the initial R script, interfacing with the data and performing the upper-triangle and melt of the covariance matrix was straightforward enough to complete. It was quite fascinating noticing how fast the SQL queries went when batched as compared to sent 1 by 1! With the help of 'svMisc' package and other optimizing methods, I managed to get our runtime down very reasonably, which was great for replicating the results on others separate machines.

Judging by the shape of our final curve (and how it tends to match portfolio efficiency frontiers), I was fairly confident that the gurobi optimizer correctly scaled up from the small-3x3 example to the full data set without much issue at all. The run time on Gurobi was decently lengthy, but not horrible considering the size of the dataset, and the scope of the decision area.

Overall, a sufficiently complex problem with its fair share of coding nuances to fix, but getting a large Gurobi model functioning with the data channels between R and Python has been a rewarding experience.