Team 14 Group Portfolio Assignment

Neil Patel, Esa Sun, Cody Clark

11/29/2019

Introduction:

Nonlinear portfolio optimization is a culmination of what we've learned throughout the MSBA program. The purpose of this assignment was to bring together R, MySQL, Gurobi, Python, Excel, and Solver. In this assignment, we take monthly data from the historic returns of nasdaq stocks from the years 2000 - 2009. Using this data we create a portfolio that maximizes the Expected Return given a certain risk tolerance. In this scenario, we use the Markowitz theory of portfolio optimization. According to this theory, risk is modeled using the variance and correlations of each stock. This theory states that to take on more risk, a portfolio should provide greater return. This creates an efficient frontier that models how return should increase as risk increases. In this assignment, we attempt to model that frontier using the real historical data.

1. R:

To accomplish this we must first calculate a matrix of covariance (risk) for each stock in addition to a vector of means (expected returns) for each stock between the period of January 2000 to December 2009. We also reformat the covariance matrix using the smelt() function from the reshape2 library. This provides an adjacency list of the covariance between any two stocks. It also makes further calculations simpler. You can see below how we accomplished this.

```r
nas <- read.csv(file = 'M6-NasdaqReturns.csv')
mat.nas <- data.matrix(nas[,4:123])
avgreturn <-  apply(mat.nas, 1, mean)
symbols <- nas$StockSymbol
avgreturn <- as.data.frame(cbind(symbols, avgreturn))
avgreturn[,1] <- nas$StockSymbol
rownames(mat.nas) <- nas$StockSymbol
mat.nas <- t(mat.nas)
cov_mat <- cov(mat.nas)
smelt <- melt(cov_mat)
```

2. SQL:

Incorporating lessons from Database Management we created a 'NASDAQ' database to store the outputs of our covariance and mean calculations. We will ultimately create a table 'Portfolio' which brings together both the expected returns and risks associated within the NASDAQ stocks calculated via Gurobi. The code to create the database and associated tables was executed in SQL not from R. Here we just populated those tables with the values from our covariance and avgreturn matrices. The RMySQL package was used here to make the connection to MySQL. Below is the code used in SQL and the code used in R to populate those tables in SQL.

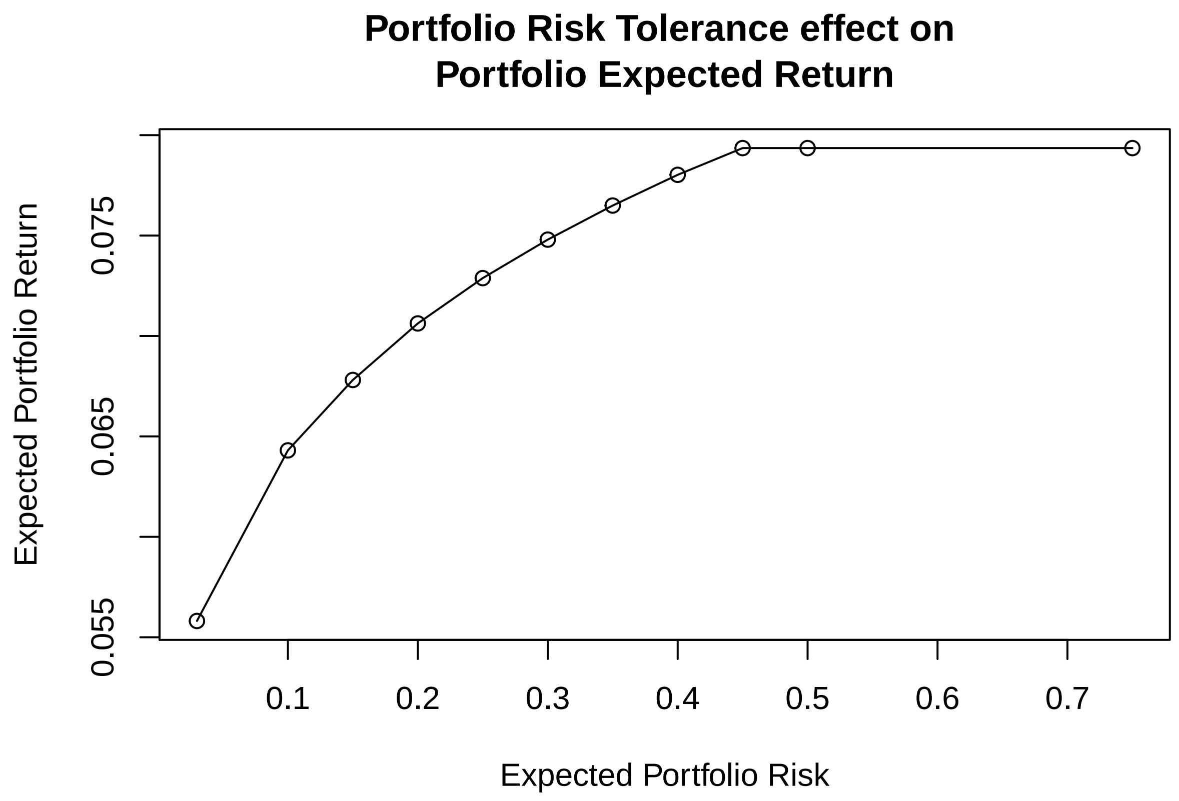
```
###########
### SQL ###
###########

create database nasdaq;
use nasdaq;
create table cov (
stock1 varchar(10),
stock2 varchar(10),
covariance double
);
create table r (
stock varchar(10),
meanReturn double
);
create table portfolio (
expReturn double,
expRisk double
);

##############
### R CODE ###
##############
db <- RMySQL:: dbConnect(drv = RMySQL::MySQL(), dbname = "nasdaq", username = "root", password = "root"
beginsql <-  sprintf("insert into cov (stock1, stock2, covariance) values")
sql <- c()
for (i in 1:nrow(smelt)) {
  sql[i] <- sprintf("('%s', '%s', %s), " ,
                    smelt[i,1], smelt[i,2], smelt[i,3])
  if (i %% 1000 == 0 | i == nrow(smelt)) {
    # dump batch
    sql[i] <- sprintf("('%s', '%s', %s);" ,
                      smelt[i,1], smelt[i,2], smelt[i,3],2)
    if(i == 1000) {
      dbSendQuery(db, paste(beginsql, paste(sql, collapse = '')))
    } else {
      j <- lastDump + 1
      dbSendQuery(db, paste(beginsql, paste(sql[j:i], collapse = '')))
    }
    lastDump <- i
  }
}
beginsql2 <- sprintf("insert into r (stock, meanReturn) values")
sql2 <- c()
for (i in 1:nrow(avgreturn)) {
  sql2[i] <- sprintf("('%s', %s), " ,
                     avgreturn[i,1], avgreturn[i,2])
  if(i == nrow(avgreturn)) {
    sql2[i] <- sprintf("('%s', %s); " ,
                       avgreturn[i,1], avgreturn[i,2])
    dbSendQuery(db, paste(beginsql2, paste(sql2, collapse = '')))
  }
}
```

3. Python:

In python, we bring the data in and then, utilizing Gurobi, create a series of objective values of expected returns given varying levels of maximum risks ranging from 0.03 to 0.75. After about .45 risk, the expected return seemed to taper off, so there was not much need to test many risk levels after that value. We sent those risks and associated expected returns to the nasdaq database and table called porftolio. The associated code for python is contained within the portfolio.py file.

4. W-R-apping Up:

Finally, we import the portfolio table back into R and create a plot of the efficient frontier (see below). As expected, the Expected Return tapers off and creates a smooth curve. This models the tradeoff between risk and return per the Markowitz theory.


Portfolio Risk Tolerance effect on
Portfolio Expected Return
Expected Portfolio Return
Expected Portfolio Risk