Graphing Efficient Frontier with R

Boyie Chen

10/25/2019

Introduction

The efficient frontier depicts the relation of the expected return and the risk. Here, we have to assume that *return* is a random variable R, and $E(R)$ represent the expected return; $sd(R)$ represent the risk of return. Most of the time, the return we observed is not far away from Normal Distribution.

Import the real stock price

Let's import the real stock price of Microsoft and JP Morgan. The data are collected between 2018.10.25-2019.10.24 in form of daily price from Yahoo Finance.

```r
setwd(path)
data1 = read.csv('MSFT.csv')
data2 = read.csv('JPM.csv')

head(data1) #Microsoft daily stock price
```

```
##         Date   Open   High    Low  Close Adj.Close   Volume
## 1 2018-10-24 108.41 108.49 101.59 102.32  100.7370 63897800
## 2 2018-10-25 106.55 109.27 106.15 108.30  106.6245 61646800
## 3 2018-10-26 105.69 108.75 104.76 106.96  105.3053 55523100
## 4 2018-10-29 108.11 108.70 101.63 103.85  102.2434 55162000
## 5 2018-10-30 103.66 104.38 100.11 103.73  102.1252 65350900
## 6 2018-10-31 105.44 108.14 105.39 106.81  105.1576 51062400
```

```r
head(data2) #JP Morgan daily stock price
```

```
##         Date   Open   High    Low  Close Adj.Close   Volume
## 1 2018-10-24 104.76 105.03 102.91 103.29  100.1627 23168900
## 2 2018-10-25 104.18 105.90 103.72 104.86  101.6852 17464700
## 3 2018-10-26 104.00 104.56 102.73 103.42  100.2888 19174900
## 4 2018-10-29 104.46 106.63 103.70 104.85  101.6755 18445200
## 5 2018-10-30 105.71 106.98 104.86 106.70  103.4695 18019700
## 6 2018-10-31 108.08 110.48 107.79 109.02  105.7192 20860000
```

```r
#We use only Open Price
stock_price = cbind(data1$Open, data2$Open)
colnames(stock_price) = c('MSFT', 'JPM')
stock = data.frame(stock_price)
head(stock)
```

```
##     MSFT    JPM
## 1 108.41 104.76
## 2 106.55 104.18
## 3 105.69 104.00
## 4 108.11 104.46
## 5 103.66 105.71
## 6 105.44 108.08
```

The concept of log return

What we care about is *Return*, so we must calculate the rate of return from stock price. We use the method of log return.

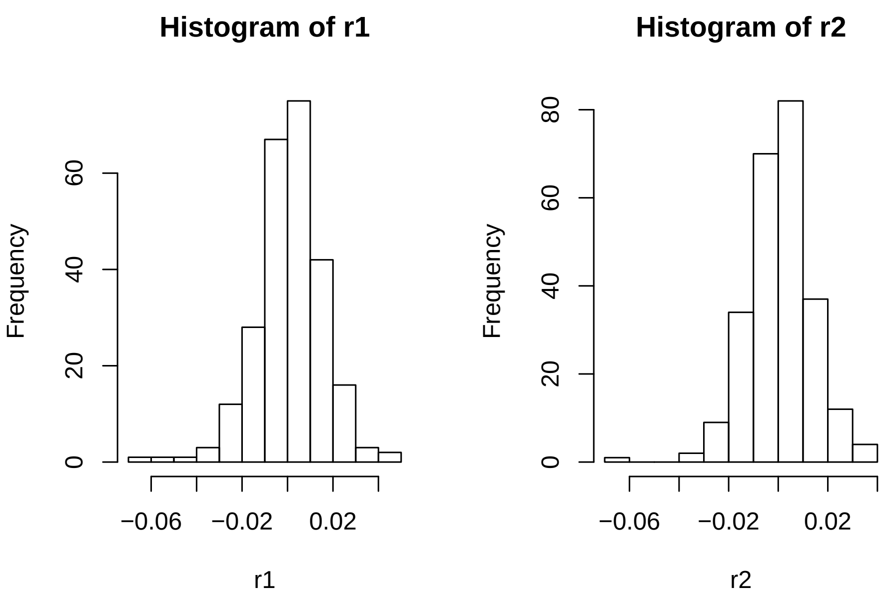
If the stock price is P_t at time period t, by definition, we have rate of return $R_t = \frac{P_t - P_{t-1}}{P_{t-1}}$ at time period t

If we consider compounding, i.e. our t and $t-1$ is really close to each other, then we can see the relation of P_{t+h} and P_t as : $P_{t+h} = \lim_{h \to 0} P_t(1+\frac{R}{h})^h = P_t e^R$ where R is the continuously compounded rate over the period. Thus, we can first take log, and then take the difference to the Price. Then $log(P_t) - log(P_{t-1}) = log(\frac{P_t}{P_{t-1}}) = log(e^R) = R$

Calculating log return in R

We see r_1, r_2 as two random variables that give us the rate of return of MSFT and JPM respectively.

```
#calculating log return
r1 = diff(log(stock$MSFT), 1)
r2 = diff(log(stock$JPM), 1)
par(mfrow = c(1,2))
hist(r1)
hist(r2)
```



So far, we can depict our 2 assets with the following moments.

```
#gaining parameters
mean(r1)*251 #annualized rate of return of stock of Microsoft
```

```
## [1] 0.2513554
```

```
mean(r2)*251 #of JP Morgan
```

```
## [1] 0.178959
```

```
sd(r1)*251 #we measure the asset's risk by its standard devition
```

```
## [1] 3.743166
```

```
sd(r2)*251
```

```
## [1] 3.245695
```

```r
cov(r1, r2) #how the 2 assets are coorelated to each other
```

```
## [1] 9.738555e-05
```

A portfolio that lower the risk

Now we consider a Portfolio P which is a weighted combination of the 2 assets. $P = \omega r_1 + (1-\omega) r_2$ The further question will be: What expected rate of return will P give us? Since P is also a radom variable, we know: $E(P) = \omega E(r_1) + (1-\omega)E(r_2)$

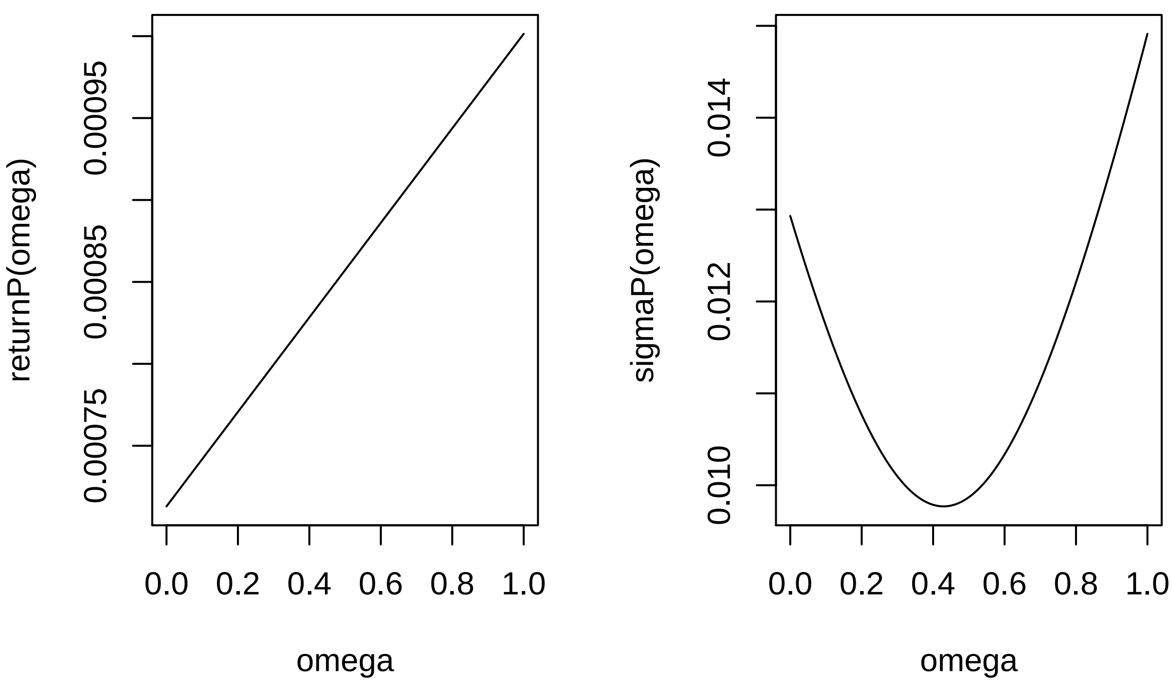
```r
#setting function(let P consisting of MSFT & JPM)
returnP = function(omega_r1){
  retP = omega_r1*mean(r1) + (1-omega_r1)*mean(r2)
  return(retP)
}
```

Also, we can calculate the risk of P measured by $\sqrt{Var(P)}$. $Var(P) = \omega^2 Var(r_1) + (1-\omega)^2 Var(r_2) + 2\omega(1-\omega)Cov(r_1, r_2)$

```r
sigmaP = function(omega_r1){
  varP = omega_r1^2*var(r1) + (1-omega_r1)^2*var(r2)
  + 2*omega_r1*(1-omega_r1)*cov(r1, r2)
  return(sqrt(varP))
}
```

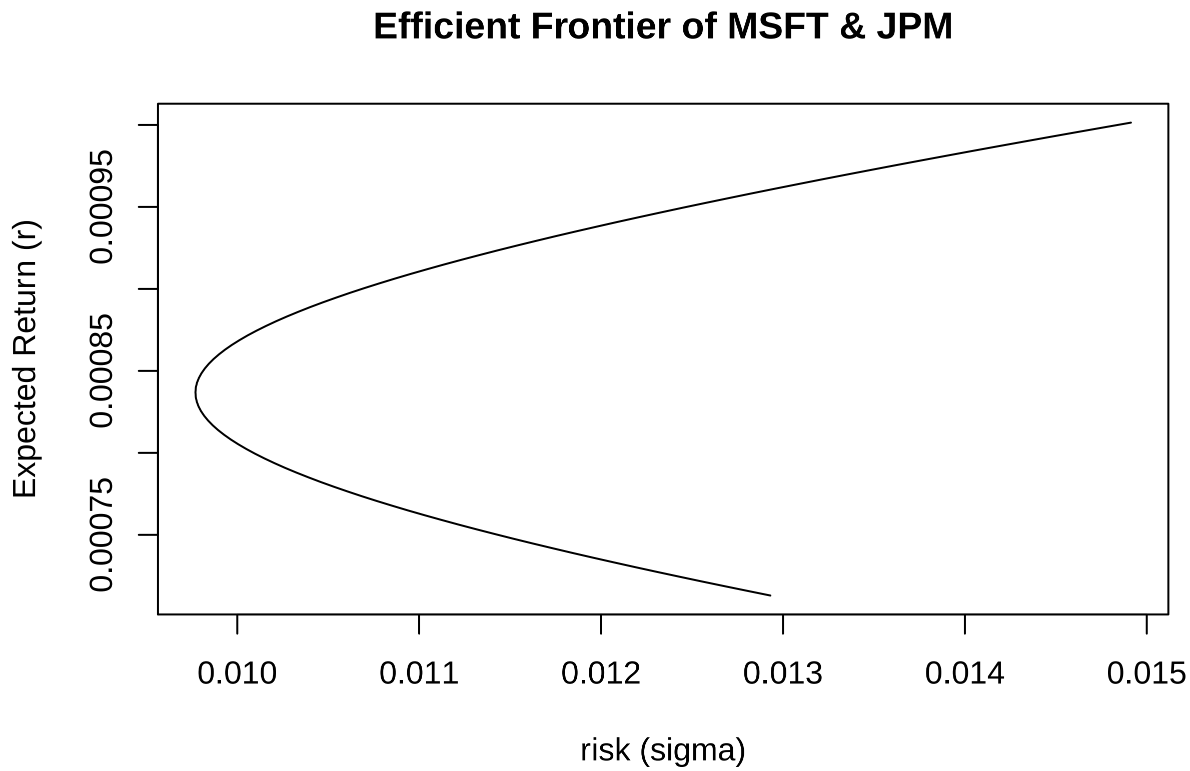
With certain ω, we can know what $E(P)$ & $\sqrt{Var(P)}$ are.

```r
omega = seq(0, 1, 0.01)
par(mfrow = c(1,2))
plot(omega, returnP(omega), type = 'l')
plot(omega, sigmaP(omega), type='l')
```



Thus, we can find the relation between risk and expected return.

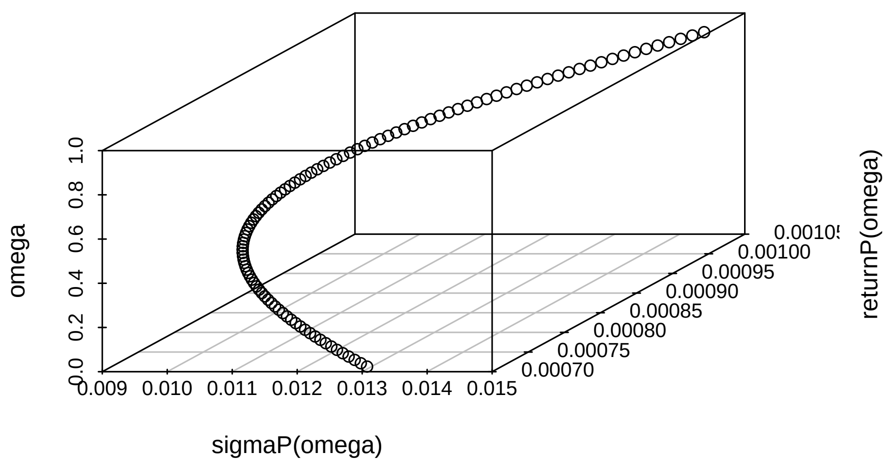
```r
#plotting sigma-return scatter plot
plot(sigmaP(omega), returnP(omega), type = 'l',
     main="Efficient Frontier of MSFT & JPM",
     ylab="Expected Return (r)", xlab = "risk (sigma)")
```



Note that the section of negative slope are not efficient.

Another way to look at the ω is to get the 3D scatter plot.

```r
#3D Plot
library(scatterplot3d)
scatterplot3d(sigmaP(omega),returnP(omega),omega)
```



Simulate Future Stock Price (Optional)

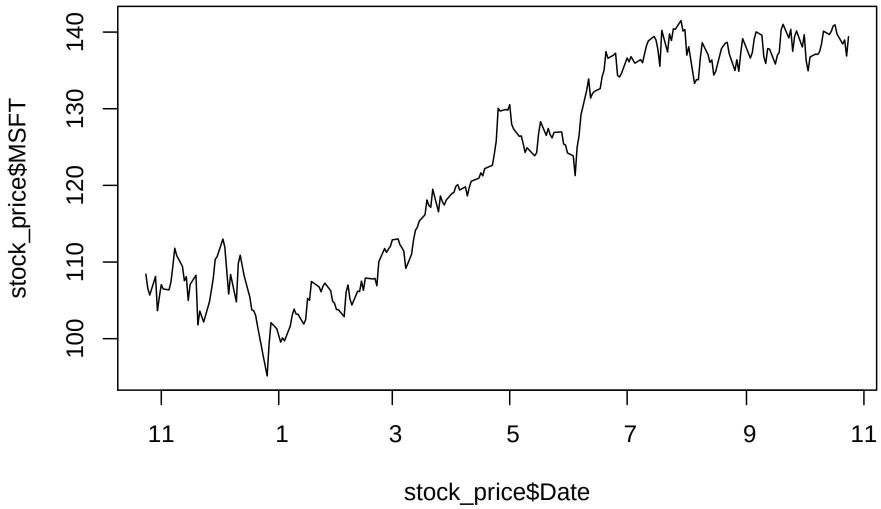
If the rate of return is truly a r.v., we can use its moments to predict/estimate future rate of return. And then we can predict/estimate the future stock price if other things remain the same.

We can draw the graph of stock price.

```
# 模擬股價波動
stock_price = cbind.data.frame(data1$Date, data1$Open, data2$Open)
colnames(stock_price) = c('Date', 'MSFT', 'JPM')

#Treat vlb as 'date' type
?as.Date
stock_price$Date = as.Date(stock_price$Date)

# 兩張股價的折線圖
plot(stock_price$Date, stock_price$MSFT, type = 'l')
```




```
plot(stock_price$Date, stock_price$JPM, type = 'l')
```

With the moments, we can find the possible future rate of return.

```r
#parameters(daily frequency)
mean(r1) #Expected rate of return of MSFT
```

```
## [1] 0.001001416
```

```r
sd(r1) #standard deviation of rate of return of MSFT
```

```
## [1] 0.01491301
```

```r
set.seed(1234)
stock_price$MSFT[1] #MSFT 第一天的股價，我們的預測要從這裡開始
```

```
## [1] 108.41
```

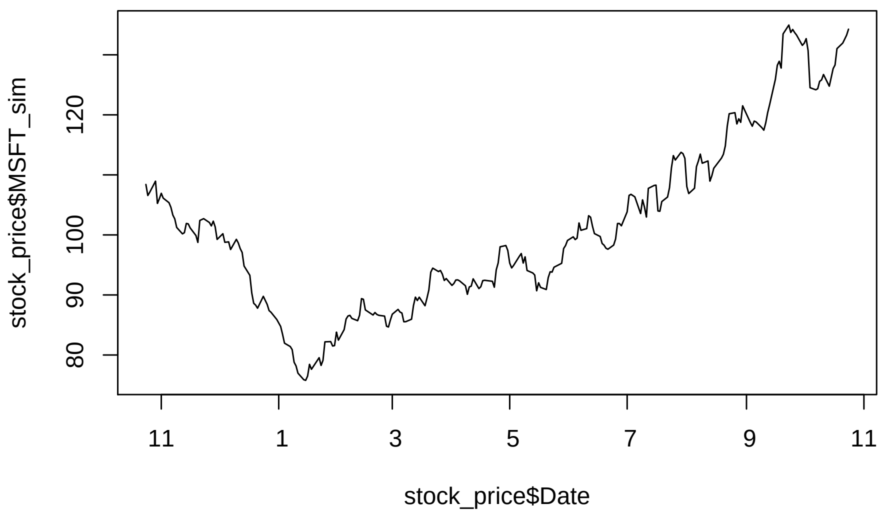
```r
days = 252-1 # 總共有 251 天要預測，也就是 compound 251 次
changes = rnorm(days, mean = mean(r1), sd = sd(r1)) # 這 251 天每天的 rate of return
```

And then, we can use the rate of return and `cumprod()` to find the predicted stock price.

```r
MSFT_sim = cumprod(c(stock_price$MSFT[1], changes+1)) # 放入起始股價以及每日的成長率，用 cumulative product
?cumprod
length(MSFT_sim) # 共有 252 天的價格
```

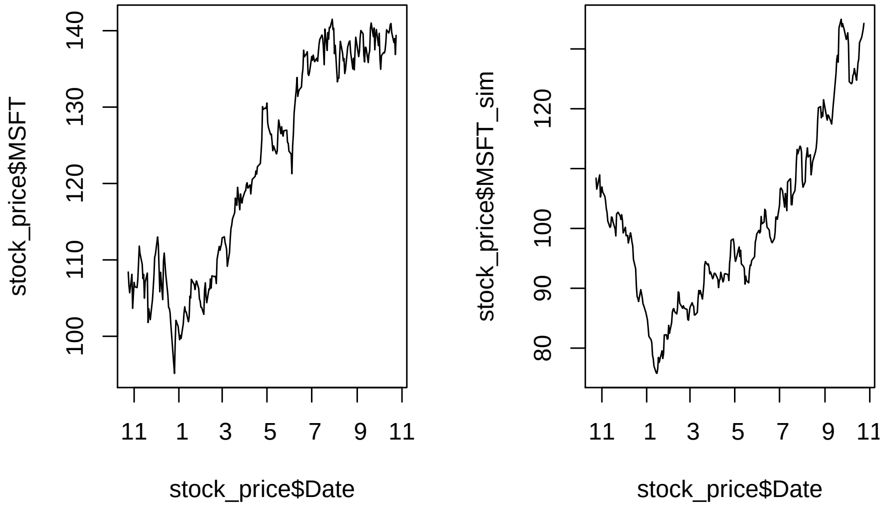
```
## [1] 252
```

```r
stock_price = cbind.data.frame(stock_price, MSFT_sim)
plot(stock_price$Date, stock_price$MSFT_sim, type = 'l')
```



Are we able to tell the difference between the real stock price of MSFT and the simulative version?

```r
# 兩張圖一起看
par(mfrow = c(1,2))
plot(stock_price$Date, stock_price$MSFT, type = 'l')
plot(stock_price$Date, stock_price$MSFT_sim, type = 'l')
```



But the above is only one possible future QQ. Though the stock price at the last day seems not too different

```
stock_price$MSFT[252] - stock_price$MSFT_sim[252]
```

```
## [1] 5.117742
```

Let's use Simulation! We simulate 10000 possible future stock prices.

```
runs = 10000

#simulates future movements and returns the closing price on day 252
generate_path = function(){
  days = 252-1
  changes = rnorm(days, mean=mean(r1), sd=sd(r1))
  sample_path = cumprod(c(stock_price$MSFT[1], changes+1))
  closing_price = sample_path[days+1] #+1 because we add the opening price
  return(closing_price)
}
```

And check whether the stock price on last day would be close to each other or not.

```
closing_sim = replicate(runs,generate_path())
mean(closing_sim)
```

```
## [1] 139.0924
```

This is the first step of simulation, actually, we can change the expected rate of return over time, and we can also add disturbance randomly.

When in doubt, Monte Carlo.

May Monte Carlo Simulation be with you!