Portfolio Optimization

In this notebook, we will first solve the problem discussed in this lecture note. Then using the same data, we will solve three different optimization problems commonly encountered in Portfolio optimization.

1. Exact problem discussed in the lecture note
2. Maximizing mean return
3. Minimizing risk
4. Simultaneously maximizing mean and minimizing risk

We will use CVXR library to solve constrained quadratic optimization problems.

Import relevant libraries.

```
suppressMessages(library(CVXR))
library(latex2exp) # To use latex symbols as plot labels
```

1. Problem discussed in the lecture note

The dataset used in this notebook can be found in the lecture notes of Prof. Shabbir Ahmed. The lecture note can be found at this link.

Read the data.

```
IBM = c(93.043, 84.585, 111.453, 99.525, 95.819, 114.708,111.515,
        113.211, 104.942, 99.827, 91.607, 107.937, 115.590)
WMT = c(51.826, 52.823, 56.477, 49.805, 50.287, 51.521, 51.531,
        48.664, 55.744, 47.916, 49.438, 51.336, 55.081)
SEHI = c(1.063, 0.938, 1.000, 0.938, 1.438, 1.700, 2.540,
         2.390, 3.120, 2.980, 1.900, 1.750, 1.800)

data = data.frame(IBM, WMT, SEHI)
data
```

```
##        IBM    WMT  SEHI
## 1   93.043 51.826 1.063
## 2   84.585 52.823 0.938
## 3  111.453 56.477 1.000
## 4   99.525 49.805 0.938
## 5   95.819 50.287 1.438
## 6  114.708 51.521 1.700
## 7  111.515 51.531 2.540
## 8  113.211 48.664 2.390
## 9  104.942 55.744 3.120
## 10  99.827 47.916 2.980
## 11  91.607 49.438 1.900
## 12 107.937 51.336 1.750
## 13 115.590 55.081 1.800
```

Calculate change in stock price

Change in stock price is calculated by subtracting a given day's stock price from following day's stock price.

```
change_in_stock_price = data[2:dim(data)[1],]-data[1:dim(data)[1]-1,]
change_in_stock_price
```

```
##        IBM    WMT   SEHI
## 2   -8.458  0.997 -0.125
## 3   26.868  3.654  0.062
## 4  -11.928 -6.672 -0.062
## 5   -3.706  0.482  0.500
## 6   18.889  1.234  0.262
## 7   -3.193  0.010  0.840
## 8    1.696 -2.867 -0.150
## 9   -8.269  7.080  0.730
## 10  -5.115 -7.828 -0.140
## 11  -8.220  1.522 -1.080
## 12  16.330  1.898 -0.150
## 13   7.653  3.745  0.050
```

Calculate rate of change of stock price

```
rate_of_return = change_in_stock_price / data[1:dim(data)[1]-1,]
rate_of_return
```

```
##            IBM           WMT        SEHI
## 2  -0.09090421  0.0192374484 -0.11759172
## 3   0.31764497  0.0691744127  0.06609808
## 4  -0.10702269 -0.1181365866 -0.06200000
## 5  -0.03723688  0.0096777432  0.53304904
## 6   0.19713209  0.0245391453  0.18219750
## 7  -0.02783590  0.0001940956  0.49411765
## 8   0.01520872 -0.0556364130 -0.05905512
## 9  -0.07304061  0.1454874240  0.30543933
## 10 -0.04874121 -0.1404276693 -0.04487179
## 11 -0.08234245  0.0317639202 -0.36241611
## 12  0.17826149  0.0383915207 -0.07894737
## 13  0.07090247  0.0729507558  0.02857143
```

Sample covariance matrix

```
C = cov(rate_of_return)
C
```

```
##              IBM         WMT        SEHI
## IBM  0.018641040 0.003598533 0.001309759
## WMT  0.003598533 0.006436938 0.004887265
## SEHI 0.001309759 0.004887265 0.068682765
```

Covariance matrix given in the paper is slightly different from this result.

Mean return for each stock

```r
means = colMeans(rate_of_return)
means
```

```
##         IBM         WMT        SEHI
## 0.026002150 0.008101316 0.073715909
```

Optimization problem of the book

Notation: C: Covariance matrix

Problem:

$$\min x^T C x$$

s.t.

$$e^T x \leq 1000$$

$$\bar{r}^T x \geq 50$$

$$x \geq 0$$

The solver solves problems of the following kind:

$$\min(x^T C x + q^T x)$$

s.t.

$$Gx \leq h$$

$$Ax = b$$

So we convert our constraints to a form understandable by the solver. We don't have any equality constraints. We modify all inequality constraints to less than equal to type. The matrix G takes following form.

$$\begin{pmatrix} 1 & 1 & 1 \\ -0.026 & -0.008 & -0.073 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \leq \begin{pmatrix} 1000 \\ -50 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

```r
G = matrix(c(rep(1,3), -means, -array(diag(3))), ncol = 3, byrow = T)
G
```

```
##             [,1]         [,2]        [,3]
## [1,]  1.00000000  1.000000000  1.00000000
## [2,] -0.02600215 -0.008101316 -0.07371591
## [3,] -1.00000000  0.000000000  0.00000000
## [4,]  0.00000000 -1.000000000  0.00000000
## [5,]  0.00000000  0.000000000 -1.00000000
```

```r
h = matrix(c(1000, -50, 0, 0, 0), ncol = 1)
h
```

```
##      [,1]
## [1,] 1000
## [2,]  -50
## [3,]    0
## [4,]    0
## [5,]    0
```

```
n = 3
x = Variable(n)
objective_1 = Minimize(quad_form(x,C))
constraints_1 = list(G %*% x <= h)
prob_1 = Problem(objective_1, constraints_1)
prob_1_sol = solve(prob_1)
print(paste0("Variance is: ",prob_1_sol$value))
```

```
## [1] "Variance is: 22634.4184565445"
```

```
cat("Solution (x) is: \n", prob_1_sol$getValue(x))
```

```
## Solution (x) is:
##  497.0455 1.466578e-20 502.9545
```

Obtained variance is higher than that of the paper. This is because the covariance matrix given in the paper is slightly different that sample covariance matrix.

2. Maximizing mean return

m : mean

$e = [1, 1, 1]^T$

Problem:

$$\max x^T m$$

s.t.

$$e^T x = 1$$
$$x^T C x \leq \sigma_0^2$$
$$x \geq 0$$

We have taken $\sigma_0^2 = 130$.

```
e = matrix(rep(1,3), ncol = 1)
x = Variable(n)
objective_2 = Maximize(t(x) %*% matrix(means, ncol = 1))
constraints_2 = list(t(e)%*%x == 1,
                     quad_form(x,C) <= 130,
                     -diag(3) %*% x <= matrix(rep(0,3), ncol = 1))
prob_2 = Problem(objective_2, constraints_2)
prob_2_sol = solve(prob_2)
print(paste0("Mean return is: ", prob_2_sol$value))
```

```
## [1] "Mean return is: 0.073715909403193"
```

```r
cat("Solution (x) is: \n", prob_2_sol$getValue(x))
```

```
## Solution (x) is:
##  5.045331e-10 9.863929e-10 1
```

3. Minimizing risk

$e = [1, 1, 1]^T$

$m = \text{mean}$

Problem:

$$\min x^T C x$$

s.t.

$$e^T x = 1$$

$$x^T m = m_0$$

$$x \geq 0$$

We have taken $m_0 = 0.05$. We have taken this peculiar value because solution doesn't exist for all values of m_0. For $m_0 = 1, 5, 10, 50, etc.$ no solution exists.

```r
m_0 = 0.05
x = Variable(n)
objective_3 = Minimize(quad_form(x,C))
constraints_3 = list(t(e) %*% x == 1,
                     t(x) %*% matrix(means, ncol = 1) == m_0,
                     -diag(3) %*% x <= matrix(rep(0,3), ncol = 1))
prob_3 = Problem(objective_3, constraints_3)
prob_3_sol = solve(prob_3)
print(paste0("Minimum variance is: ", prob_3_sol$value))
```

```
## [1] "Minimum variance is: 0.0226344184565446"
```

```r
cat("Solution (x) is: \n", prob_3_sol$getValue(x))
```

```
## Solution (x) is:
##  0.4970455 -3.024006e-24 0.5029545
```

4. Simultaneously maximizing mean return and minimizing risk

We have taken $\lambda = 5$. λ value can be changed to control risk and obtain different results.

$e = [1, 1, 1]^T$

Problem:

$$\max x^T m - \lambda x^T C x$$

s.t.

$$e^T x = 1$$

$$x \geq 0$$

```r
lamda = 5
x = Variable(n)
objective_4 = Maximize(t(x) %*% matrix(means, ncol = 1) - lamda*quad_form(x,C))
constraints_4 = list(t(e) %*% x == 1,
                     -diag(3) %*% x <= matrix(rep(0,3), ncol = 1))
prob_4 = Problem(objective_4, constraints_4)
prob_4_sol = solve(prob_4)
print(paste0("Maximum value is: ", prob_4_sol$value))
```

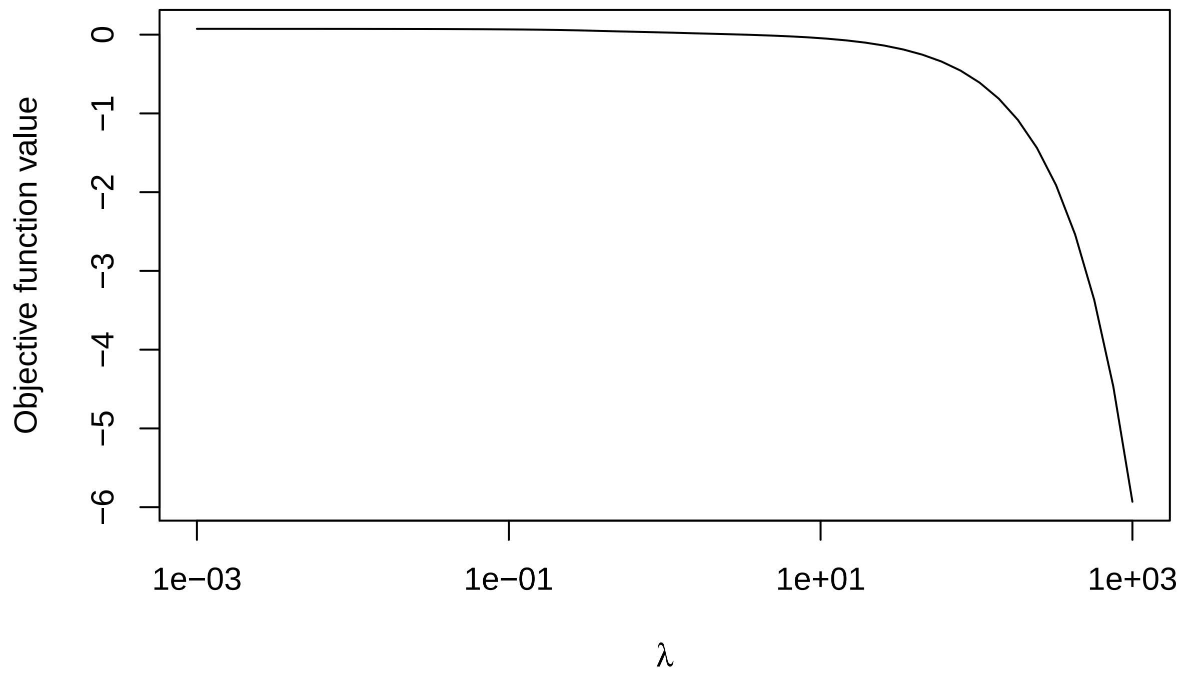
```
## [1] "Maximum value is: -0.0128161176512607"
```

```r
cat("Solution (x) is: \n", prob_4_sol$getValue(x))
```

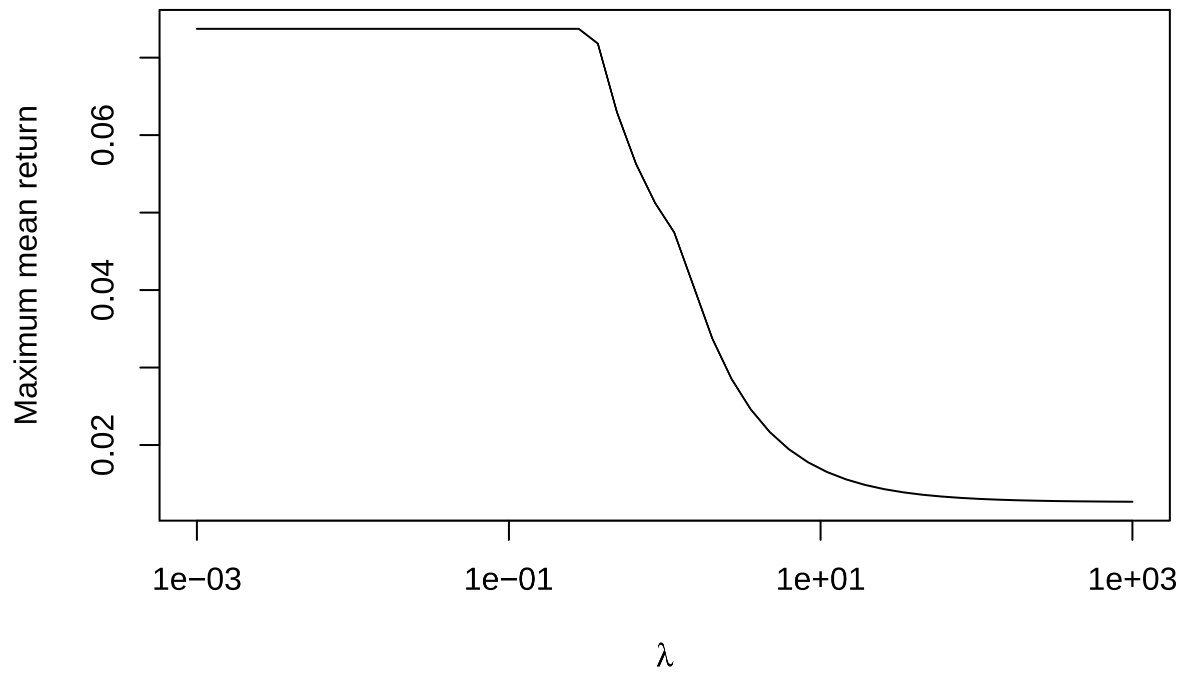
```
## Solution (x) is:
##  0.2641051 0.6087802 0.1271147
```

Solving for different values of λ

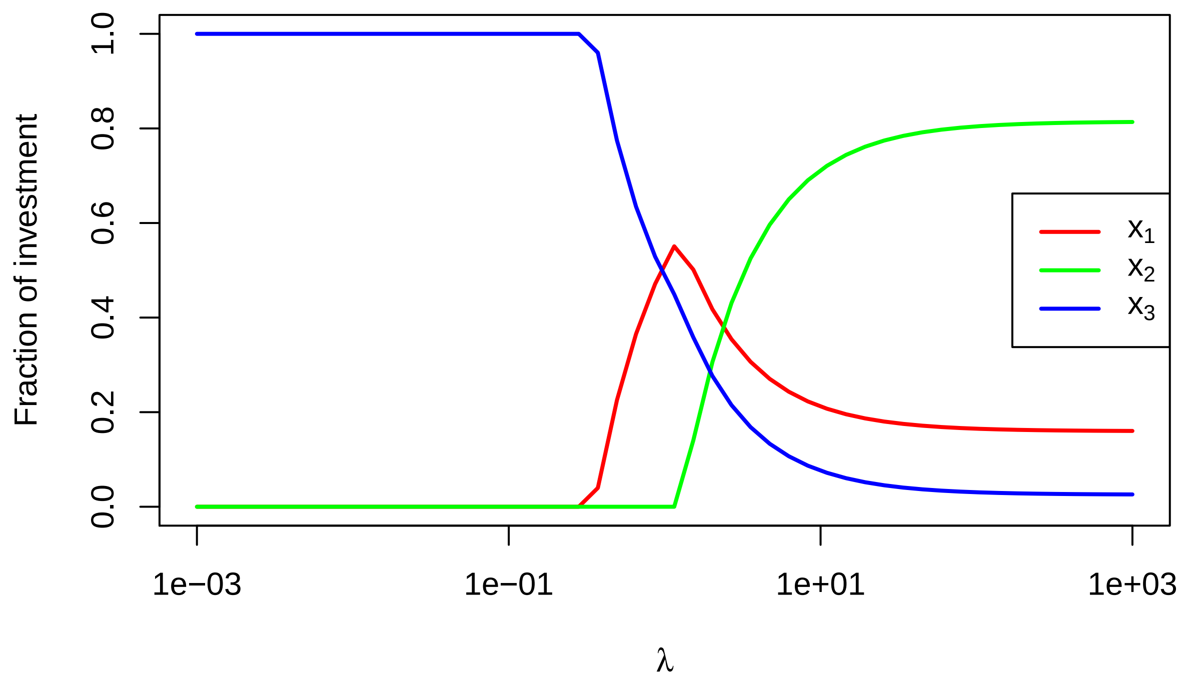
```r
lambda = 10^seq(-3,3,length.out = 50)
value = array(rep(NaN, 50))
x_values = matrix(rep(NaN, 50*3), ncol = 3)
i = 1
for (lam in lambda){
  x = Variable(n)
  objective = Maximize(t(x) %*% matrix(means, ncol = 1) - lam*quad_form(x,C))
  constraints = list(t(e) %*% x == 1,
                     -diag(3) %*% x <= matrix(rep(0,3), ncol = 1))
  problem = Problem(objective, constraints)
  sol = solve(problem)
  value[i] = sol$value
  x_values[i,] = sol$getValue(x)
  i = i + 1
}

plot(lambda, value, log = "x", type = "l",
     xlab = TeX("$\\lambda"), ylab = "Objective function value")
```



```r
mean_returns = x_values %*% matrix(means, ncol = 1)
plot(lambda, mean_returns, log = "x", type = "l",
     xlab = TeX("$\\lambda"), ylab = "Maximum mean return")
```



```
plot(lambda, x_values[,1], log = "x", type = "l", col = "red",ylim = c(0,1), lwd = 2,
     xlab = TeX("$\\lambda"), ylab = "Fraction of investment")
points(lambda, x_values[,2], type = "l", col = "green", lwd = 2)
points(lambda, x_values[,3], type = "l", col = "blue", lwd = 2)
legend("right", legend = c(TeX("$x_1$"), TeX("$x_2$"), TeX("$x_3")),
       col = c("red", "green", "blue"), lty = c(1,1,1), lwd = 2)
```



```r
sessionInfo()
```

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 7 x64 (build 7601) Service Pack 1
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United Kingdom.1252
## [2] LC_CTYPE=English_United Kingdom.1252
## [3] LC_MONETARY=English_United Kingdom.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United Kingdom.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] latex2exp_0.4.0 CVXR_1.0-1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.4.6    lattice_0.20-41 gmp_0.6-0       digest_0.6.25
##  [5] R6_2.4.1        grid_4.0.2      magrittr_1.5    evaluate_0.14
##  [9] rlang_0.4.5     stringi_1.4.6   Rmpfr_0.8-1     Matrix_1.2-18
## [13] rmarkdown_2.1   osqp_0.6.0.3    tools_4.0.2     bit64_0.9-7
## [17] stringr_1.4.0   bit_1.1-15.2    xfun_0.13       yaml_2.2.1
## [21] compiler_4.0.2  htmltools_0.4.0 ECOSolveR_0.5.3 knitr_1.28
```