FE 630: Midterm

Due on Tuesday, July 18, 2017

Professor Starer 18:15pm

Yuxuan Xia

Contents

Problem 1

1 Prove that $\nabla_{\boldsymbol{x}} S = \boldsymbol{a}$

$$S = \sum_{i=1}^{n} a_i x_i = \boldsymbol{a}^T \boldsymbol{x} \tag{1}$$

By decomposition of scalars,

$$(\nabla_x S)_\omega = \frac{\partial S}{\partial x_\omega} \tag{2}$$

$$= \sum_{i=1}^{n} \frac{\partial}{\partial x_\omega} a_i x_i \tag{3}$$

$$= a_\omega \tag{4}$$

Therefore,

$$\nabla_{\boldsymbol{x}} S = \boldsymbol{a} \tag{5}$$

2 Prove that $\nabla_{\boldsymbol{x}} \boldsymbol{a}^T \boldsymbol{x} = \nabla_{\boldsymbol{x}} \boldsymbol{x}^T \boldsymbol{a}$

$$\boldsymbol{a}^T \boldsymbol{x} = \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{x}_i \tag{6}$$

$$\boldsymbol{x}^T \boldsymbol{a} = \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{x}_i \tag{7}$$

$$\boldsymbol{x}^T \boldsymbol{a} = \boldsymbol{a}^T \boldsymbol{x} \tag{8}$$

$$\frac{\partial \boldsymbol{a}^T \boldsymbol{x}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{x}^T \boldsymbol{a}}{\partial \boldsymbol{x}} = \frac{\partial \sum_{i=1}^{n} a_i x_i}{\partial \boldsymbol{x}} \tag{9}$$

$$= \boldsymbol{a} \tag{10}$$

3 Prove that $\frac{\partial S}{\partial \boldsymbol{x}} = \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{Q}^T \boldsymbol{x}$

$$(\nabla_{\boldsymbol{x}} S)_\omega = (\nabla \boldsymbol{x} \sum_{i,j} x_i q_{i\,j} x_j)_\omega \tag{11}$$

$$= \frac{\partial}{\partial x_\omega} \sum_{i,j} x_i q_{i\,j} x_j \tag{12}$$

$$= \frac{\partial}{\partial x_\omega} [\sum_{i,j:i=j} x_i q_{i\,j} x_j + \sum_{i,j:i\neq j} x_i q_{i\,j} x_j] \tag{13}$$

$$= 2 \sum_{i,j:i=j} (x_i q_{i\,j} \delta_{i\,j} \delta_{i\,\omega}) + \sum_{i,j:i\neq j} (x_i q_{i\,j} \delta_{j\,\omega} + (x_j q_{i\,j} \delta_{i\,\omega}) \tag{14}$$

$$= \sum_j q_{\omega j} x_j + \sum_i x_i q_{i\omega} \tag{15}$$

$$= (\boldsymbol{Q}\boldsymbol{x})_\omega + (\boldsymbol{Q}^T \boldsymbol{x})_\omega \tag{16}$$

Using matrix and vector representation:

$$\frac{\partial S}{\partial \boldsymbol{x}} = \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{Q}^T \boldsymbol{x} \tag{17}$$

Problem 2

1(a) Use the constraint to write one variable (h1 or h2) in terms of the other, and substitute into the objective function to produce a new reduced objective function in terms of one variable only.
Objective function

$$\begin{aligned} f(h_1,h_2) &= a^2h_1^2 - 2abh_1h_2 + b^2h_2^2 \qquad (18)\\ &= (ah_1 - bh_2)^2 \qquad (19)\end{aligned}$$

Substitute the constraint into the objective function

$$\begin{aligned} f_r(h_1) &= [ah_1 - b(c-h_1)]^2 \qquad (20)\\ &= [(a+b)h_1 - bc]^2 \qquad (21)\\ &\geq 0 \qquad (22)\end{aligned}$$

Since the quadratic function is minimized only if it equals to zero if possible,
the optimized h_1 and h_2 satisfy

$$\hat{h}_1 = \arg\min_{h_1} f_r(h_1) = \frac{bc}{a+b} \qquad (23)$$

$$\hat{h}_2 = c - h_1 = \frac{ac}{a+b} \qquad (24)$$

1(b) Differentiate the reduced objective function with respect to the one variable and set the result equal to zero to produce an equation that can be solved for the one variable.
Differentiation

$$\frac{\partial}{\partial h_1} f_r(h_1) = 2[(a+b)h_1 - bc](a+b) = 0 \qquad (25)$$

since $a > 0$ and $b > 0$

$$(a+b)h_1 = bc \qquad (26)$$

$$h_1 = \frac{bc}{a+b} \qquad (27)$$

1(c) Solve for both h1 and h2.
Solution in 2.1(b)

$$h_1 = \frac{bc}{a+b} \qquad (28)$$

$$h_2 = c - h_1 = \frac{ac}{a+b} \qquad (29)$$

Equal to the equation 23 24 in 2.1(a)

Problem 2 continued on next page. . .

2(a) Write the problem in a scalar Lagrangian form.
The Lagrangian:

$$L(h,\lambda) = \frac{1}{2}(ah_1 - bh_2)^2 + \lambda(h_1 + h_2 - c) \tag{30}$$

$$= \frac{1}{2}\begin{pmatrix} h_1 & h_2 \end{pmatrix}\begin{pmatrix} a^2 & -ab \\ -ab & b^2 \end{pmatrix}\begin{pmatrix} h_1 \\ h2 \end{pmatrix} + \lambda^T \begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} h_1 \\ h2 \end{pmatrix} - c \tag{31}$$

By vector form, using the following representation

$$h = \begin{pmatrix} h_1 \\ h2 \end{pmatrix} \tag{32}$$

$$A = \begin{pmatrix} 1 & 1 \end{pmatrix} \tag{33}$$

$$Q = \begin{pmatrix} a^2 & -ab \\ -ab & b^2 \end{pmatrix} \tag{34}$$

By substitution

$$L(h,\lambda) = \frac{1}{2}h^T Q h - \lambda^T(Ah - c) \tag{35}$$

2(b) Prove the following set of simultaneous linear equations:

$$\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix}\begin{bmatrix} h \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ c \end{bmatrix} \tag{36}$$

Differentiate the equation 35

$$\nabla_h L(h,\lambda) = Qh + A^T\lambda = 0 \tag{37}$$

$$\nabla_\lambda L(h,\lambda) = Ah - c = 0 \tag{38}$$

Result

$$\begin{pmatrix} Q & A^T \end{pmatrix}\begin{pmatrix} h \\ \lambda \end{pmatrix} = 0 \tag{39}$$

$$\begin{pmatrix} A & 0 \end{pmatrix}\begin{pmatrix} h \\ \lambda \end{pmatrix} = c \tag{40}$$

By combination

$$\begin{pmatrix} Q & A^T \\ A & 0 \end{pmatrix}\begin{pmatrix} h \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ c \end{pmatrix} \tag{41}$$

Problem 3

3 Computing the expected utility of each investment, which one would the investor prefer.

The investor's utility function: $U(W) = W - 0.05W^2$

$$E[U(W)] = E(W) - 0.05E(W^2) \tag{42}$$

As for investment A

$$\begin{aligned}
E(W_A) &= \sum_i outcome_i(A)prob_i(A) && (43)\\
&= \begin{pmatrix}5 & 7 & 10\end{pmatrix}\begin{pmatrix}0.2\\0.5\\0.3\end{pmatrix} && (44)\\
&= 7.5 && (45)\\
E(W_A^2) &= \sum_i outcome_i^2(A)prob_i(A) && (46)\\
&= \begin{pmatrix}5^2 & 7^2 & 10^2\end{pmatrix}\begin{pmatrix}0.2\\0.5\\0.3\end{pmatrix} && (47)\\
&= 59.5 && (48)\\
E[U(W_A)] &= E(W_A) - 0.05E(W_A^2) && (49)\\
&= 7.5 - 0.05 * 59.5 && (50)\\
&= 4.525 && (51)
\end{aligned}$$

As for investment B

$$\begin{aligned}
E(W_B) &= \sum_i outcome_i(B)prob_i(B) && (52)\\
&= \begin{pmatrix}6 & 8 & 9\end{pmatrix}\begin{pmatrix}0.3\\0.6\\0.1\end{pmatrix} && (53)\\
&= 7.5 && (54)\\
E(W_B^2) &= \sum_i outcome_i^2(B)prob_i(B) && (55)\\
&= \begin{pmatrix}6^2 & 8^2 & 9^2\end{pmatrix}\begin{pmatrix}0.3\\0.6\\0.1\end{pmatrix} && (56)\\
&= 57.3 && (57)\\
E[U(W_B)] &= E(W_B) - 0.05E(W_B^2) && (58)\\
&= 7.5 - 0.05 * 57.3 && (59)\\
&= 4.635 && (60)
\end{aligned}$$

Since $E[U(W_B)] > E[U(W_A)]$ The investor would prefer the investment B.

Problem 4

1. What is the investor's risk premium?

The terminal value $W = \mu + x$, where μ is a fixed amount of \$5000, and x is random variable with mean zero and standard deviation \$1000.

$$E(W) = \mu \tag{61}$$
$$Var(W) = E(W^2) - E^2(W) \tag{62}$$
$$= \sigma_x^2 \tag{63}$$

Using the definition of risk premium and certainty equivalent of wealth:

$$U(\mu - y) = E[U(\mu + x)] \tag{64}$$
$$U(\cdot) = log(\cdot) \tag{65}$$

Scheme A: x satisfies Gaussian distribution

$$log(5000 - y) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma_x^2}} log(\mu + x) exp(\frac{-x^2}{2\sigma_x^2}) \mathrm{d}x \tag{66}$$
$$\approx 0.0085 \tag{67}$$

In scheme A x cannot be negative as the utility function is the log function. So the result 0.0085 ignored the tiny imaginary part and therefore, $y = 4999$.
Scheme B: x satisfies Taylor approximation of Gaussian distribution as shown in book

$$E[U(\mu + x)] = E[U(\mu) + \sum_{k=1}^{\infty} \frac{U^{(k)}(\mu)}{k!} x^k] \tag{68}$$
$$U(\mu - y) = U(\mu) + \sum_{k=1}^{\infty} \frac{U^{(k)}(\mu)}{k!} (-y)^k \tag{69}$$
$$\tag{70}$$

ignore the higher order terms in Taylor expansion:

$$\frac{U^{(2)}(\mu)}{2} \sigma_x^2 = \frac{U^{(1)}(\mu)}{1!} (-y) \tag{71}$$
$$y = -\frac{U^{(2)}(\mu)\sigma_x^2}{2U^{(1)}(\mu)} \tag{72}$$
$$= \frac{\frac{1}{\mu^2}\sigma_x^2}{2\frac{1}{\mu}} \tag{73}$$
$$= \frac{\sigma_x^2}{2\mu} \tag{74}$$
$$= \frac{1000^2}{2 \cdot 5000} \tag{75}$$
$$= 100 \tag{76}$$

Problem 4 continued on next page...

2. what is the investor's certainty equivalent of wealth?

$$Certaintyequivalent = \mu - y \tag{77}$$

Scheme A: certainty equivalent = 5000-4999 = 1
Scheme B: certainty equivalent = 5000-100 = 4900

Comparison:
Scheme A: risk premium = 4999
Scheme B: risk premium = 100
Conclusion:
The result can be extremely different using different distribution, utility function and domain of each function. Scheme A is a failure algorithm as it cannot reflect the true premium.

Problem 5

Assume that the average variance of return for an individual security is 50 units and that the average covariance between securities is 10 units.
1. What is the minimum attainable variance of an equally-weighted portfolio's return as the number of securities tends to infinity?

The equally-weighted portfolio's variance

$$\begin{aligned}
\sigma_p^2 = h^T Q h &= \frac{1}{N^2}\sum_{i,j}\sigma_{ij} && (78)\\
&= \frac{1}{N^2}\sum_{i,j:i=j}\sigma_{ij} + \frac{1}{N^2}\sum_{i,j:i\neq j}\sigma_{ij} && (79)\\
&= \frac{1}{N^2}\sum_{i}\sigma_{ii} + \frac{N(N-1)}{N^2}\sum_{i,j:i\neq j}\frac{1}{N(N-1)}\sigma_{ij} && (80)\\
&= \frac{1}{N}avg(\sigma_i^2) + \frac{N-1}{N}avg(\sigma_{ij}) && (81)\\
&= \frac{1}{N}[avg(\sigma_i^2) + avg(\sigma_{ij})] + avg(\sigma_{ij}) && (82)
\end{aligned}$$

According to the condition

$$avg(\sigma_i^2) = 50 \tag{83}$$
$$avg(\sigma_i j) = 10 \tag{84}$$

By substitution

$$\begin{aligned}
variance_{min} &= \lim_{N\to\infty}\sigma_p^2 && (85)\\
&= avg(\sigma_{ij}) && (86)\\
&= 10 && (87)
\end{aligned}$$

2. What is the variance of an equally-weighted portfolio of 5, 10, 20, 50, and 100 securities?

$$\sigma_p^2(N) = \frac{1}{N}(50-10)+10 = \frac{40}{N}+10 \tag{88}$$

Substitute N = 5,10,20,50,100 into 88

$$\sigma_p^2(5) = 18 \tag{89}$$
$$\sigma_p^2(10) = 14 \tag{90}$$
$$\sigma_p^2(20) = 12 \tag{91}$$
$$\sigma_p^2(50) = 10.8 \tag{92}$$
$$\sigma_p^2(100) = 10.4 \tag{93}$$

3. How many securities need to be held such that the variance of the portfolio is 10% more than the minimum variance?

$$\sigma_{p\ min}^2 = 10 \tag{94}$$
$$\sigma_p^2(N) = \sigma_{p\ min}^2(1+10\%) \tag{95}$$
$$\Longrightarrow\ N = 40 \tag{96}$$

Problem 6

Consider the following expected return vector , return covariance matrix Q; and portfolio weight vector h:

$$\mu = \begin{bmatrix} 1.1\% \\ 1.2\% \\ 1.3\% \end{bmatrix}, Q = \begin{bmatrix} 3\% & 0.2\% & 0.1\% \\ 0.2\% & 7\% & 0.4\% \\ 0.1\% & 0.4\% & 4\% \end{bmatrix}, h = \begin{bmatrix} 0.2 \\ 0.3 \\ 0.5 \end{bmatrix} \tag{97}$$

1. Compute the expected return of the portfolio.

Expected return

$$r_E = h^T\mu = \begin{bmatrix} 0.2 & 0.3 & 0.5 \end{bmatrix} \begin{bmatrix} 1.1\% \\ 1.2\% \\ 1.3\% \end{bmatrix} = 1.23\% \tag{98}$$

2. Compute the variance and standard deviation of return of the portfolio.

$$variance = \sigma_p^2 = h^TQh = 1.914 \times 10^{-2} \tag{99}$$
$$standarddeviation = \sigma_p = \sqrt{variance} = 1.383 \times 10^{-1} \tag{100}$$

Problem 7

Given the following data:

$$\mu = \begin{bmatrix} 1.1\% \\ 1.2\% \\ 1.3\% \end{bmatrix}, Q = \begin{bmatrix} 1\% & 0 & 0 \\ 0 & 5\% & 0 \\ 0 & 0 & 7\% \end{bmatrix} \tag{101}$$

Assume that short sales are allowed. 1. Find the composition (i.e., the weights), standard deviation, and expected return of the portfolio that satisfies the budget constraint and has minimum risk.

variance of return:

$$\sigma_p^2 = h^T Q h \tag{102}$$

budget constrain:

$$h_1 + h_2 + h_3 = 1 \tag{103}$$

Lagrangian function:

$$L(h, \lambda) = \frac{1}{2} h^T Q h + \lambda (h_1 + h_2 + h_3 - 1) \tag{104}$$

differentiation:

$$\nabla_h L(h, \lambda) = Qh + \lambda \iota = 0 \tag{105}$$

$$\frac{\partial}{\partial \lambda} L(h, \lambda) = \iota^T h - 1 = 0 \tag{106}$$

By combination and representing in vector form:

$$\begin{pmatrix} Q & \iota \\ \iota^T & 0 \end{pmatrix} \begin{pmatrix} h \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} \tag{107}$$

$$\begin{pmatrix} h \\ \lambda \end{pmatrix} = \begin{pmatrix} Q & \iota \\ \iota^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} \tag{108}$$

$$= \begin{pmatrix} 0.74 \\ 0.15 \\ 0.11 \\ -0.01 \end{pmatrix} \tag{109}$$

composition:

$$h = \begin{pmatrix} 0.74 \\ 0.15 \\ 0.11 \end{pmatrix} \tag{110}$$

expected return:

$$r_E = h^T \mu = 1.136\% \tag{111}$$

variance:

$$\sigma^2 = h^T Q h = 0.007447 \tag{112}$$

standard deviation:

$$\sigma_p = \sqrt{variance} = 0.0863 \tag{113}$$

2. Plot the expected return and standard deviation (i.e., map out the efficient frontier) by maximizing the objective $O = \tau \mu^T h - \frac{1}{2} h^T Q h$ subject to the constraint $h^T \iota = 1$ for the range $\tau = 0, 0.01, 0.02, \cdots 0.99, 1$.

Problem 7 continued on next page...

Lagrangian function:

$$L(h,\lambda) = \frac{1}{2}h^TQh - \tau\mu^Th + \lambda(\iota^Th - 1) \tag{114}$$

Differential:

$$\nabla_h L(h,\lambda) = Qh - (\tau\mu^T)^T + (\lambda\iota^T)^T = Qh - \tau\mu + \lambda\iota \tag{115}$$

$$\frac{\partial}{\partial\lambda}L(h,\lambda) = h^T\iota - 1 \tag{116}$$

simultaneous linear equations:

$$\begin{pmatrix} Q & \iota \\ \iota^T & 0 \end{pmatrix}\begin{pmatrix} h \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} \tag{117}$$

$$\begin{pmatrix} h \\ \lambda \end{pmatrix} = \begin{pmatrix} Q & \iota \\ \iota^T & 0 \end{pmatrix}^{-1}\begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} \tag{118}$$

for each portfolio:
expected return

$$r_p = h^T\mu \tag{119}$$

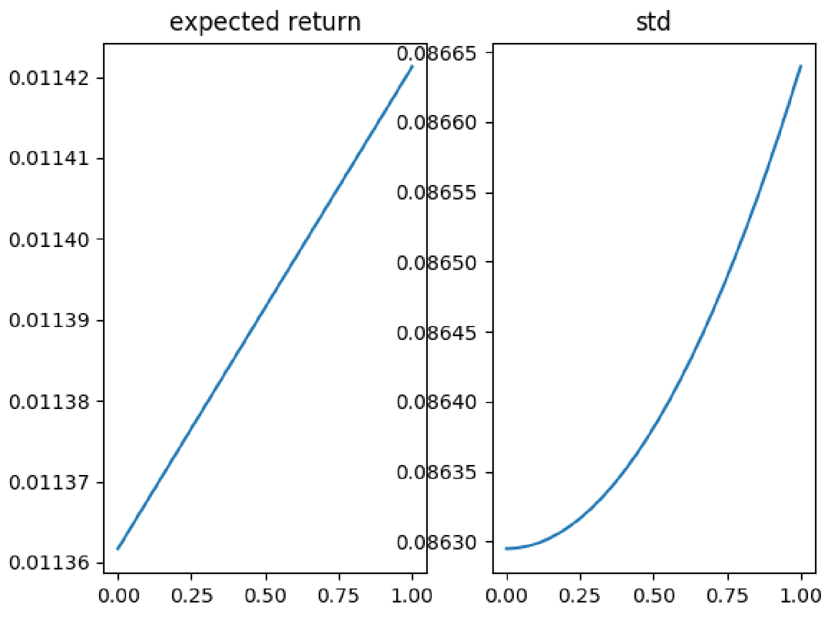
variance

$$\sigma_p^2 = h^TQ \tag{120}$$

standard deviation

$$\sigma_p = \sqrt{h^TQh} \tag{121}$$

Calculate composition h through 118 and substitute into 119 , 121
Using different τ, one can obtain two set of key-value pairs (τ, r_p),(τ, σ_p)



Listing 1: Python Script for Question 7

```
import numpy as np
import matplotlib.pyplot as plt

# 7.1
M = np.array([[.01, 0, 0, 1], [0, .05, 0, 1], [0, 0, .07, 1], [1, 1, 1, 0]])
b = np.array([[0], [0], [0], [1]])
hlambda = np.matmul(np.linalg.inv(M), b)
h = hlambda[0:-1, :]
print("h=", h)
Q = np.array([[.01, 0, 0], [0, .05, 0], [0, 0, .07]])
var = np.matrix(h.T) * np.matrix(Q) * np.matrix(h)
print("variance=", var[0, 0])
std = np.sqrt(var)
print("std=", std[0, 0])
mu = [1.1, 1.2, 1.3]
re = np.matmul(h.T, mu)
print("expected return = ", re[0], "%")

# 7.2
M = M
mu = [.011, .012, .013]
Q = Q
M_inv = np.linalg.inv(M)


def b(coef):
    mu = [.011, .012, .013]
    b = [i * coef for i in mu]
    b.append(1)
    return np.asarray(b).T


x_coef = [i * 0.01 for i in range(101)]
h_list = []
r_list = []
std_list = []
for coef in x_coef:
    hlambda = np.matmul(M_inv, b(coef=coef))
    h = hlambda[0:-1]
    h_list.append(h)
    re = np.matmul(h, mu)
    r_list.append(re)
    var = np.matrix(h) * np.matrix(Q) * np.matrix(h).T
    std_list.append(np.sqrt(var[0, 0]))

plt.subplot(121)
plt.title("expected return")
plt.plot(x_coef, r_list)

plt.subplot(122)
plt.title("std")
plt.plot(x_coef, std_list)
```

Problem 7 continued on next page...

```
plt.show()
```

Problem 8

Consider again the return vector and covariance matrix Q of the risky securities given in Question 6. Assume that short sales of these risky securities are allowed.
1. Suppose that a risk-free investment with a return of 0.5% is available. Find the tangency portfolio, its expected return, and the standard deviation of its return.

Lagrangian:

$$L = \frac{1}{2}h^TQh + \lambda_1(h^T\mu - \mu_p) + \lambda_2(h^T\iota - 1) \tag{122}$$

efficient frontier:

$$\frac{\sigma^2}{1/C} - \frac{(\mu_p - B/C)^2}{D/C} = 1 \tag{123}$$

which is a hyperbola in (σ, μ)-space where

$$\sigma_p^2 = h^TQh \tag{124}$$
$$A = \mu^TQ^{-1}\mu = 9.473 \times 10^{-3} \tag{125}$$
$$B = \mu^TQ^{-1}\iota = 7.931 \times 10^{-1} \tag{126}$$
$$C = \iota^TQ^{-1}\iota = 6.683 \times 10^{1} \tag{127}$$
$$D = AC - B^2 = 3.937 \times 10^{-3} \tag{128}$$
$$Sharpe\ ratio = \frac{r_p - r_f}{\sigma} \tag{129}$$

by definition of tangency portfolio, the constrained minimization of multivariate function:

$$\min -\frac{r_p - r_f}{\sigma} \tag{130}$$

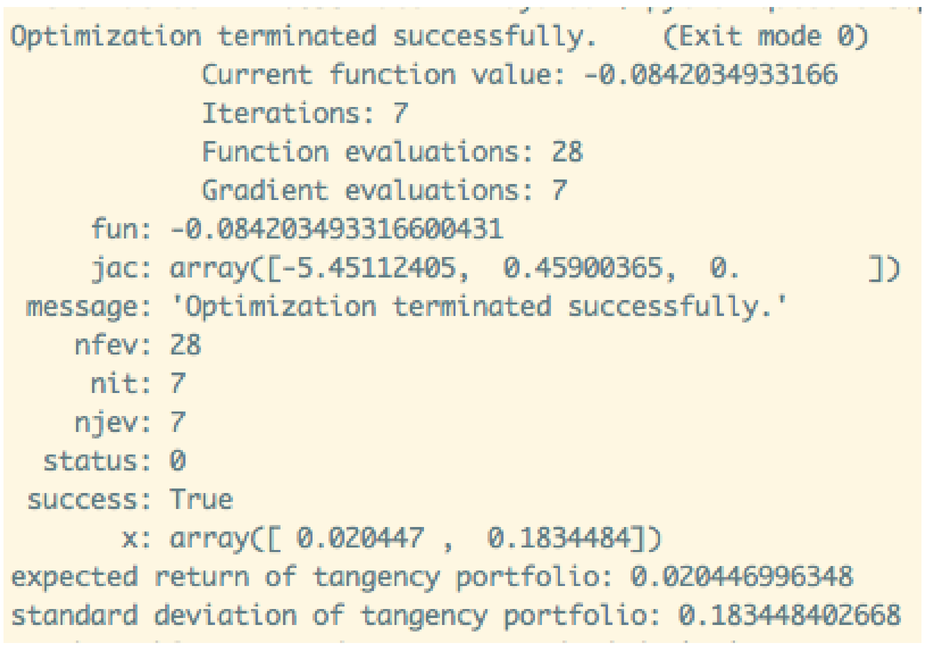
subject to:

$$efficient\ frontier(\mu_p, \sigma; A, B, C, D) \tag{131}$$
$$= \frac{\sigma^2}{1/C} - \frac{(\mu_p - B/C)^2}{D/C} - 1 = 0 \tag{132}$$
$$r_p \geq r_f \tag{133}$$
$$\sigma > 0 \tag{134}$$

using Scipy.optimize.minimize(see ”question8.py”)
result:

$$maximized\ Sharpe\ ratio = 0.0842 \tag{135}$$
$$optimized\ return = 0.0204 \tag{136}$$
$$optimized\ standard\ deviation = 0.1834 \tag{137}$$

Problem 8 continued on next page. . .



```
Optimization terminated successfully.    (Exit mode 0)
            Current function value: -0.0842034933166
            Iterations: 7
            Function evaluations: 28
            Gradient evaluations: 7
     fun: -0.084203493316600431
     jac: array([-5.45112405,  0.45900365,  0.        ])
 message: 'Optimization terminated successfully.'
    nfev: 28
     nit: 7
    njev: 7
  status: 0
 success: True
       x: array([ 0.020447 ,  0.1834484])
expected return of tangency portfolio: 0.020446996348
standard deviation of tangency portfolio: 0.183448402668
```

2. Find and tabulate the compositions, expected returns, and standard deviations of the following combinations:

expected return:

$$r_e = h^T\mu = \begin{pmatrix} h_1 & h_2 \end{pmatrix}\begin{pmatrix} r_f \\ r_p \end{pmatrix} = h_1 r_f + h_2 r_p \tag{138}$$

variance:

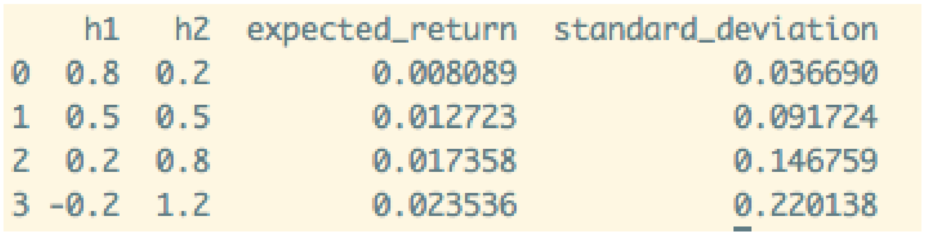
$$var = h^T Q h = \sigma^2 h_2^2 \tag{139}$$

standard deviation:

$$std = \sqrt{var} = \sigma h_2 \tag{140}$$

substitute different pairs of (h_1, h_2) into 138,140

index	h_1	h_2	expected return	standard deviation
0	0.8	0.2	0.008089	0.036690
1	0.5	0.5	0.012723	0.091724
2	0.2	0.8	0.017358	0.146759
3	-0.2	1.2	0.023536	0.220138



```
    h1   h2  expected_return  standard_deviation
0  0.8  0.2         0.008089            0.036690
1  0.5  0.5         0.012723            0.091724
2  0.2  0.8         0.017358            0.146759
3 -0.2  1.2         0.023536            0.220138
```

Listing 2: Python Script for Question 8

```
import numpy as np

mu = np.array([1.1, 1.2, 1.3]).T * 0.01
Q = np.array([[3, 0.2, 0.1], [0.2, 7, 0.4], [0.1, 0.4, 4]]) * 0.01
l = np.array([1, 1, 1]).T
A = np.matmul(np.matmul(mu.T, np.linalg.inv(Q)), mu)
B = np.matmul(np.matmul(mu.T, np.linalg.inv(Q)), l)
C = np.matmul(np.matmul(l.T, np.linalg.inv(Q)), l)
D = A * C - B * B
rf = 0.005


def efficientFrontier(re, sigma, A, B, C, D):
    return sigma * sigma / (1 / C) - pow((re - B / C), 2) / (D / C) - 1


def sharpeRatio(re, rf, sigma):
    return (re - rf) / sigma


# 8.1
from scipy.optimize import minimize

SR = lambda x: -sharpeRatio(x[0], rf, x[1])
cons = ({'type': 'eq',
         'fun': lambda x: np.array([efficientFrontier(x[0], x[1], A, B, C, D)]),
         'jac': lambda x: np.array([-2 * (x[0] - B / C) / (D / C), 2 * C * x[1]])},
        {'type': 'ineq',
         'fun': lambda x: np.array([x[0] - rf]),
         'jac': lambda x: np.array([1, 0])},
        {'type': 'ineq',
         'fun': lambda x: np.array([x[1]]),
         'jac': lambda x: np.array([0, 1])}
        )
res = minimize(SR, [0, 0.5], constraints=cons, options={'disp': True})
print(res)

optimized_sharpe_ratio = -res.fun
optimized_return = res.x[0]
optimized_sigma = res.x[1]
print("expected return of tangency portfolio:", optimized_return)
print("standard deviation of tangency portfolio:", optimized_sigma)

# 8.2
import pandas as pd

mu_pair = np.array([rf, optimized_return]).T


def expectedReturn(h, mu):
    return np.matmul(h.T, mu)
```

```python
def standardDeviation(h, sigma):
    return sigma * h[1]


df = pd.DataFrame(columns=['h1', 'h2', 'expected_return', 'standard_deviation'])
h1_list = [.8, .5, .2, -.2]
for i in range(np.size(h1_list)):
    h = np.array([h1_list[i], 1 - h1_list[i]]).T
    df.loc[i] = [h[0], h[1], expectedReturn(h, mu_pair), standardDeviation(h, optimized_sigma)]

print(df)
```

Problem 9

1. Download daily price data for the first ten companies alphabetically by ticker in the NASDAQ 100 index from the beginning of January 2015 to the end of December 2016. Place the adjusted closing prices into a matrix in which each column represents a company, and each row represents a date in ascending order. Show your code for downloading and arranging this data. Do NOT print the price data itself.

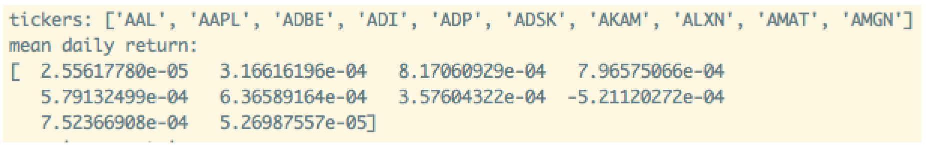
Since Yahoo API is not available this time, alternatively using quandl source.
See "question9.py" #9.1

2. From the price data matrix, use matrix manipulation techniques (with no loops) to compute a simple returns matrix (i.e., not logarithmic returns matrix) in which each column represents a company, and each row represents a day. The entry in the ijth position should therefore be the return on day i of security j: Show your code for doing these computations. Do NOT print the returns data itself.

See "question9.py" #9.2

3. Use a single built-in command to compute the mean daily return for all ten stocks simultaneously. Show your code, and DO print the mean return of each of the stocks.

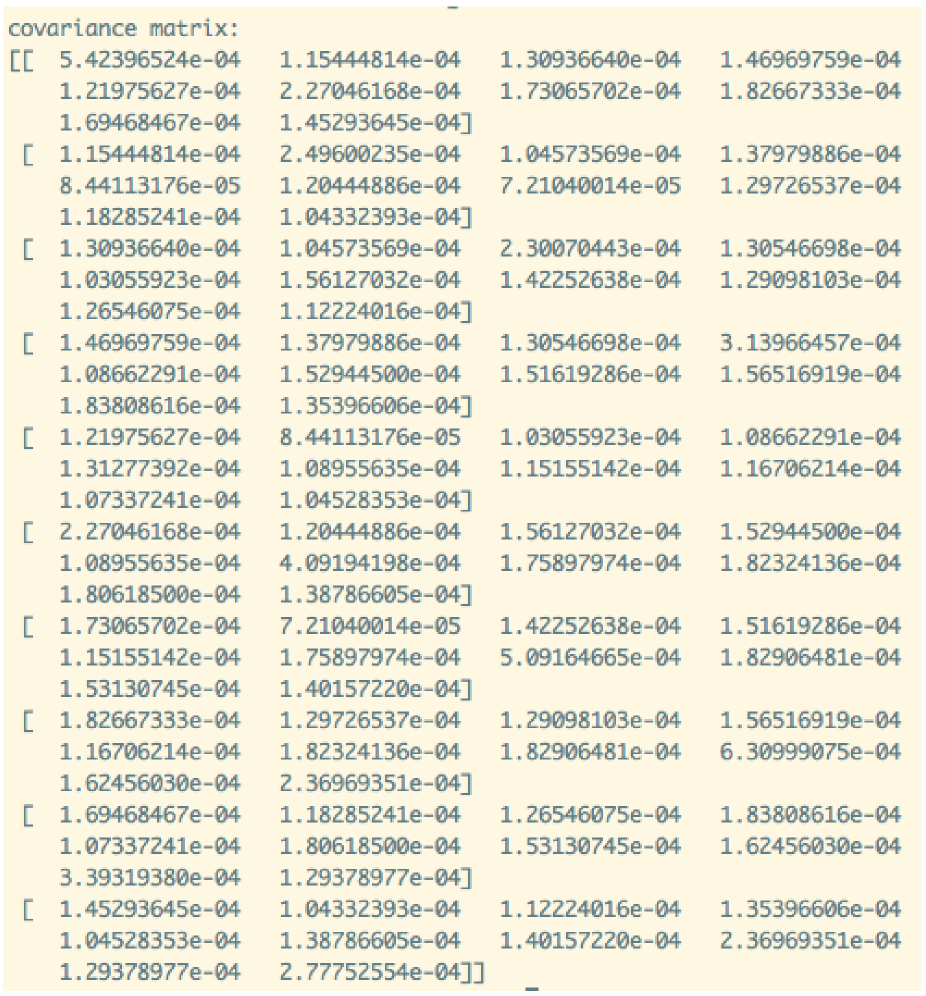
See "question9.py" #9.3



```
tickers: ['AAL', 'AAPL', 'ADBE', 'ADI', 'ADP', 'ADSK', 'AKAM', 'ALXN', 'AMAT', 'AMGN']
mean daily return:
[  2.55617780e-05   3.16616196e-04   8.17060929e-04   7.96575066e-04
   5.79132499e-04   6.36589164e-04   3.57604322e-04  -5.21120272e-04
   7.52366908e-04   5.26987557e-05]
```

4. Use your language's help system to find a single built-in command that computes the covariance matrix from a matrix of data points. Use that single built-in command to compute the covariance matrix of the returns data. Show your code, and DO print the resulting 10 × 10 covariance matrix.

See "question9.py" #9.4



```
covariance matrix:
[[  5.42396524e-04   1.15444814e-04   1.30936640e-04   1.46969759e-04
    1.21975627e-04   2.27046168e-04   1.73065702e-04   1.82667333e-04
    1.69468467e-04   1.45293645e-04]
 [  1.15444814e-04   2.49600235e-04   1.04573569e-04   1.37979886e-04
    8.44113176e-05   1.20444886e-04   7.21040014e-05   1.29726537e-04
    1.18285241e-04   1.04332393e-04]
 [  1.30936640e-04   1.04573569e-04   2.30070443e-04   1.30546698e-04
    1.03055923e-04   1.56127032e-04   1.42252638e-04   1.29098103e-04
    1.26546075e-04   1.12224016e-04]
 [  1.46969759e-04   1.37979886e-04   1.30546698e-04   3.13966457e-04
    1.08662291e-04   1.52944500e-04   1.51619286e-04   1.56516919e-04
    1.83808616e-04   1.35396606e-04]
 [  1.21975627e-04   8.44113176e-05   1.03055923e-04   1.08662291e-04
    1.31277392e-04   1.08955635e-04   1.15155142e-04   1.16706214e-04
    1.07337241e-04   1.04528353e-04]
 [  2.27046168e-04   1.20444886e-04   1.56127032e-04   1.52944500e-04
    1.08955635e-04   4.09194198e-04   1.75897974e-04   1.82324136e-04
    1.80618500e-04   1.38786605e-04]
 [  1.73065702e-04   7.21040014e-05   1.42252638e-04   1.51619286e-04
    1.15155142e-04   1.75897974e-04   5.09164665e-04   1.82906481e-04
    1.53130745e-04   1.40157220e-04]
 [  1.82667333e-04   1.29726537e-04   1.29098103e-04   1.56516919e-04
    1.16706214e-04   1.82324136e-04   1.82906481e-04   6.30999075e-04
    1.62456030e-04   2.36969351e-04]
 [  1.69468467e-04   1.18285241e-04   1.26546075e-04   1.83808616e-04
    1.07337241e-04   1.80618500e-04   1.53130745e-04   1.62456030e-04
    3.39319380e-04   1.29378977e-04]
 [  1.45293645e-04   1.04332393e-04   1.12224016e-04   1.35396606e-04
    1.04528353e-04   1.38786605e-04   1.40157220e-04   2.36969351e-04
    1.29378977e-04   2.77752554e-04]]
```

Problem 9 continued on next page. . .

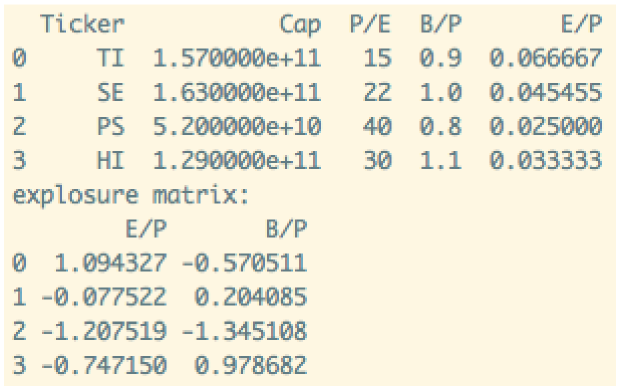
Listing 3: Python Script for Question 9

```python
import quandl
import numpy as np
import pandas as pd
import os.path

# quandl.ApiConfig.api_key = "pgZTGbk8X-D7SdwAS_wN"

# 9.1
nasdaqDF = pd.read_csv("nasdaq100list.csv")
symbols = nasdaqDF.ix[:, 'Symbol'].sort_values().tolist()
tickers = symbols[0:10]
print("tickers:", tickers)
if os.path.isfile("stockDataFrame.csv"):
    df = pd.read_csv("stockDataFrame.csv")
    df.index = df.Date
    df = df.ix[:, 1:]
else:
    closeList = []
    for ticker in tickers:
        stockDF = quandl.Dataset('WIKI/' + ticker).data(
            params={'start_date': '2015-01-01', 'end_date': '2016-12-31'}).to_pandas()
        stockClose = stockDF.ix[:, 'Adj. Close']
        closeList.append(stockClose)

    df = pd.concat(closeList, axis=1)
    df.columns = tickers

# 9.2
closeMatrix = df.as_matrix()
returnMatrix = (closeMatrix[1:, :] - closeMatrix[:-1, :]) / closeMatrix[:-1, :]

# 9.3
meanDailyReturn = returnMatrix.mean(axis=0)
print("mean daily return:")
print(meanDailyReturn)

# 9.4
covariance = np.cov(returnMatrix.T)
print("covariance matrix:")
print(covariance)
```

Problem 10

1. Compute the exposures of the four companies to the earnings yield factor (i.e, E/P, the inverse of P/E) and the book-to-price (B/P) factor. Show your computations, and list the resulting X matrix.

See "question10.py" #10.1



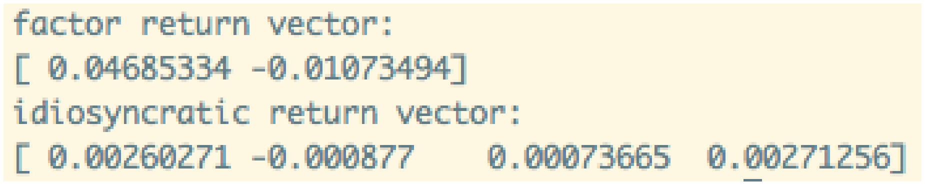
```
  Ticker           Cap  P/E  B/P       E/P
0     TI  1.570000e+11   15  0.9  0.066667
1     SE  1.630000e+11   22  1.0  0.045455
2     PS  5.200000e+10   40  0.8  0.025000
3     HI  1.290000e+11   30  1.1  0.033333
explosure matrix:
        E/P       B/P
0  1.094327 -0.570511
1 -0.077522  0.204085
2 -1.207519 -1.345108
3 -0.747150  0.978682
```

2. Suppose that at one snapshot, the excess return vector is

$$r = \begin{bmatrix} 0.0600 \\ -0.0067 \\ -0.0414 \\ -0.0428 \end{bmatrix}$$

Compute:
(a) An estimate of the factor return vector, f.
(b) An estimate of the idiosyncratic return vector ϵ:

Using pseudo inverse technique to calculate linear regression
See "question10.py" #10.2



```
factor return vector:
[ 0.04685334 -0.01073494]
idiosyncratic return vector:
[ 0.00260271 -0.000877    0.00073665  0.00271256]
```

Listing 4: Python Script for Question 10

```python
import numpy as np
import pandas as pd

data = [['TI', 1.57e11, 15, .9], ['SE', 1.63e11, 22, 1], ['PS', 0.52e11, 40, .8], ['HI', 1.29e11
dataset = pd.DataFrame(data=data, columns=['Ticker', 'Cap', 'P/E', 'B/P'])
ep = 1 / dataset.ix[:, 'P/E']
dataset.ix[:, 'E/P'] = ep
print(dataset)


# 10.1
def exposure(v, c):
    cmean = 0
```

Problem 10 continued on next page...

```python
    for i in range(len(v)):
        cmean += v[i] * c[i]
    cmean /= np.sum(c)
    return (v - cmean) / np.std(v,ddof=1)


ep_exposure = exposure(dataset.ix[:, 'E/P'], dataset.ix[:, 'Cap'])
bp_exposure = exposure(dataset.ix[:, 'B/P'], dataset.ix[:, 'Cap'])
X = pd.concat([ep_exposure, bp_exposure], axis=1)
print("explosure matrix:")
print(X)

# 10.2
r = np.array([0.06, -.0067, -.0414, -.0428])
# using pseudo inverse to compute factor returns
# which is equalvalent to least square regression
f = np.matmul(np.linalg.pinv(X), r)
print("factor return vector:")
print(f)

epsilon = r - np.matmul(X, f)
print("idiosyncratic return vector:")
print(epsilon)
```