Lecture 6: Modern Portfolio Theory

Douglas Chung

National Chengchi University
22 October 2021

1 Optimal Risky Portfolios (N Assets)

1.1 Step 1: simulate asset returns

We simulate returns for 3 assets using the normal distribution.

[1]:
```
import numpy as np
import pandas as pd

mu_A = 0.05
sig_A = 0.15

mu_B = 0.08
sig_B = 0.20

mu_C = 0.12
sig_C = 0.225

A = pd.DataFrame(np.random.normal(mu_A, sig_A, size=(100, 1)), columns=list('A'))
B = pd.DataFrame(np.random.normal(mu_B, sig_B, size=(100, 1)), columns=list('B'))
C = pd.DataFrame(np.random.normal(mu_C, sig_C, size=(100, 1)), columns=list('C'))

df = pd.concat([A, B, C], axis=1)
df.head()
```

[1]:

	A	B	C
0	0.296695	0.259181	0.208324
1	0.128018	0.150919	0.102826
2	-0.068420	-0.020775	0.282342
3	0.069268	0.779666	-0.189741
4	-0.076177	0.314679	-0.390004

1.2 Step 2: compute sample statistics

We compute sample mean and sample variance-covariance matrix of the simulated data.

[2]:
```
R = df.mean()
COV = df.cov()

print('Sample mean:')
print(R)
print('Sample variance-covariance matrix:')
print(COV)
```

```
Sample mean:
A    0.058477
B    0.062581
C    0.131937
dtype: float64
Sample variance-covariance matrix:
          A         B         C
A  0.016387 -0.000300  0.003821
B -0.000300  0.045290 -0.000696
C  0.003821 -0.000696  0.049543
```

1.3 Step 3: compute the minimum variance portfolio (MVP)

The matrix formula for MVP is:

$$\mathbf{w}_{\text{GMVP}} = \frac{\Sigma^{-1}\mathbf{1}}{\mathbf{1}^{\intercal}\Sigma^{-1}\mathbf{1}}$$

[3]:
```
from numpy.linalg import inv
import math

ONE = np.ones(3)
InvCOV = inv(COV)
W_GMVP = np.dot(InvCOV, ONE)/np.dot(ONE.T, np.dot(InvCOV, ONE))
ret_GMVP = np.dot(W_GMVP, R)
sig_GMVP = math.sqrt(np.dot(W_GMVP, np.dot(COV, W_GMVP)))

print('GMVP:')
print(W_GMVP)
print('ret_GMVP:')
print(ret_GMVP)
print('sig_GMVP:')
print(sig_GMVP)
```

```
GMVP:
[0.59815404 0.23541957 0.16642639]
ret_GMVP:
0.07166892143253469
sig_GMVP:
0.10181798390675448
```

1.4 Step 4: compute the maximum Sharpe ratio portfolio (MSRP)

The matrix formula for MSRP is:

$$\mathbf{w}_{\text{MSRP}} = \frac{\Sigma^{-1}[\mathbb{E}[\text{R}] - r_f \mathbf{1}]}{\mathbf{1}^{\intercal}\Sigma^{-1}[\mathbb{E}[\text{R}] - r_f \mathbf{1}]}$$

[4]:
```
r_f = 0.01

ONE = np.ones(3)
ER = R - r_f
InvCOV = inv(COV)
W_MSRP = np.dot(InvCOV, ER)/np.dot(ONE.T, np.dot(InvCOV, ER))
ret_MSRP = np.dot(W_MSRP, R)
sig_MSRP =  math.sqrt(np.dot(W_MSRP, np.dot(COV, W_MSRP)))

SR_MSRP = (ret_MSRP - r_f)/sig_MSRP

print('MSRP:')
print(W_MSRP)
print('ret_MSRP:')
print(ret_MSRP)
print('sig_MSRP:')
print(sig_MSRP)
```

```
MSRP:
[0.41130462 0.20380574 0.38488964]
ret_MSRP:
0.08758760673477341
sig_MSRP:
0.11420562499580156
```

1.5 Step 5: trace out the entire minimum variance frontier

Portfolio expected returns along the MVF:

$$\mathbb{E}[r_{u,\text{MVF}}] = u\mathbb{E}[r_{\text{GMVP}}] + (1-u)\mathbb{E}[r_{\text{MSRP}}]$$

Portfolio risk along the MVF:

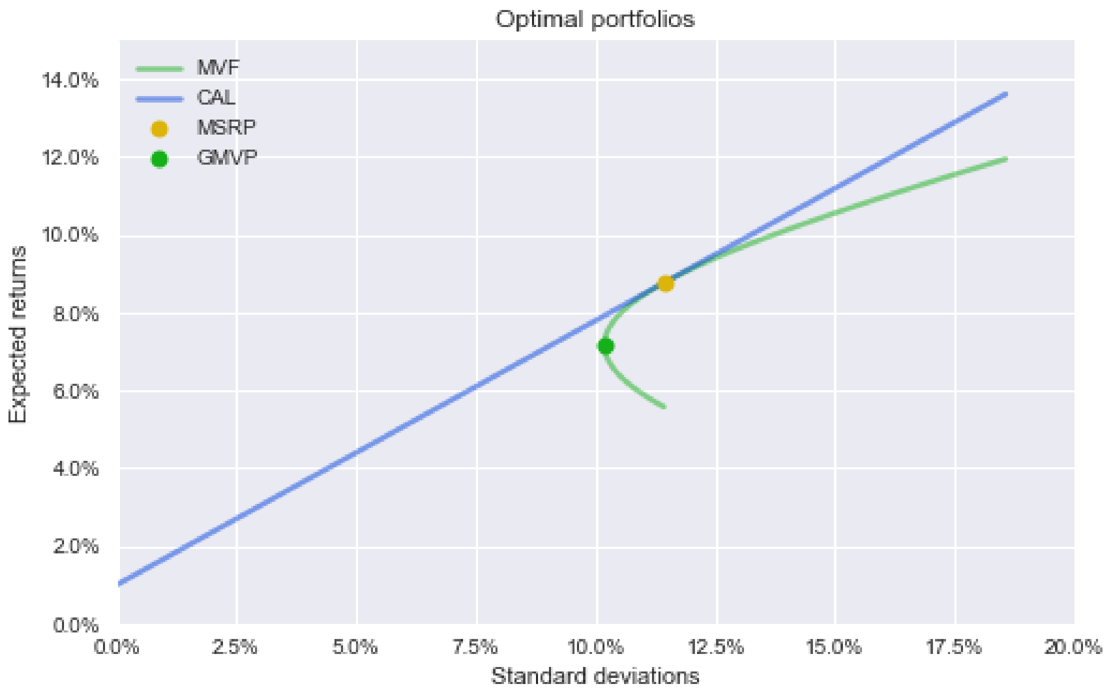
$$\sigma_{u,\text{MVF}} = \sqrt{u^2\text{Var}[r_{\text{GMVP}}] + (1-u)^2\text{Var}[r_{\text{MSRP}}] + 2u(1-u)\text{Cov}[r_{\text{GMVP}}, r_{\text{MSRP}}]}$$

[17]:
```
W = np.linspace(-2, 2, 100, endpoint=True)
ret_MVF = []
sig_MVF = []
for u in W:
    ret_MVF.append(u*ret_GMVP + (1-u)*ret_MSRP)
    sig_MVF.append(math.sqrt(u**2 * sig_GMVP**2 + (1-u)**2 * sig_MSRP**2 +␣
 ↪2*u*(1-u)*np.dot(W_GMVP, np.dot(COV, W_MSRP))))

CAL_x = np.linspace(0, np.max(sig_MVF), 50, endpoint=True)
CAL_y = r_f + SR_MSRP*CAL_x

import matplotlib.pyplot as plt
import matplotlib.ticker as mtick

plt.style.use('seaborn')
fig = plt.figure(figsize=(8, 5))
ax = fig.add_subplot(1,1,1)
plt.plot(sig_MVF, ret_MVF, color='xkcd:green', linewidth=2.5, alpha=0.5)
plt.plot(CAL_x, CAL_y, color='xkcd:blue', linewidth=2.5, alpha=0.5)
plt.plot(sig_MSRP, ret_MSRP, 'o', color='xkcd:gold', markersize=8)
plt.plot(sig_GMVP, ret_GMVP, 'o', color='xkcd:green', markersize=8)
plt.ylabel('Expected returns')
plt.xlabel('Standard deviations')
ax.yaxis.set_major_formatter(mtick.PercentFormatter(1.0))
ax.xaxis.set_major_formatter(mtick.PercentFormatter(1.0))
plt.legend(['MVF', 'CAL', 'MSRP', 'GMVP'], loc='best')
plt.title('Optimal portfolios')
plt.xlim([0, 0.2])
plt.ylim([0, 0.15])
plt.show()
```


Optimal portfolios
MVF
CAL
MSRP
GMVP
Expected returns
Standard deviations
0.0%
2.0%
4.0%
6.0%
8.0%
10.0%
12.0%
14.0%
0.0%
2.5%
5.0%
7.5%
10.0%
12.5%
15.0%
17.5%
20.0%

1.6 Step 6: generate random portfolios

[6]:
```
W_RAND = pd.DataFrame(np.random.uniform(0, 1, size=(100, 3)),␣
↪columns=list('ABC'))
W_RAND = W_RAND.divide(W_RAND.sum(axis=1), axis=0)
W_RAND.head()
```

[6]:

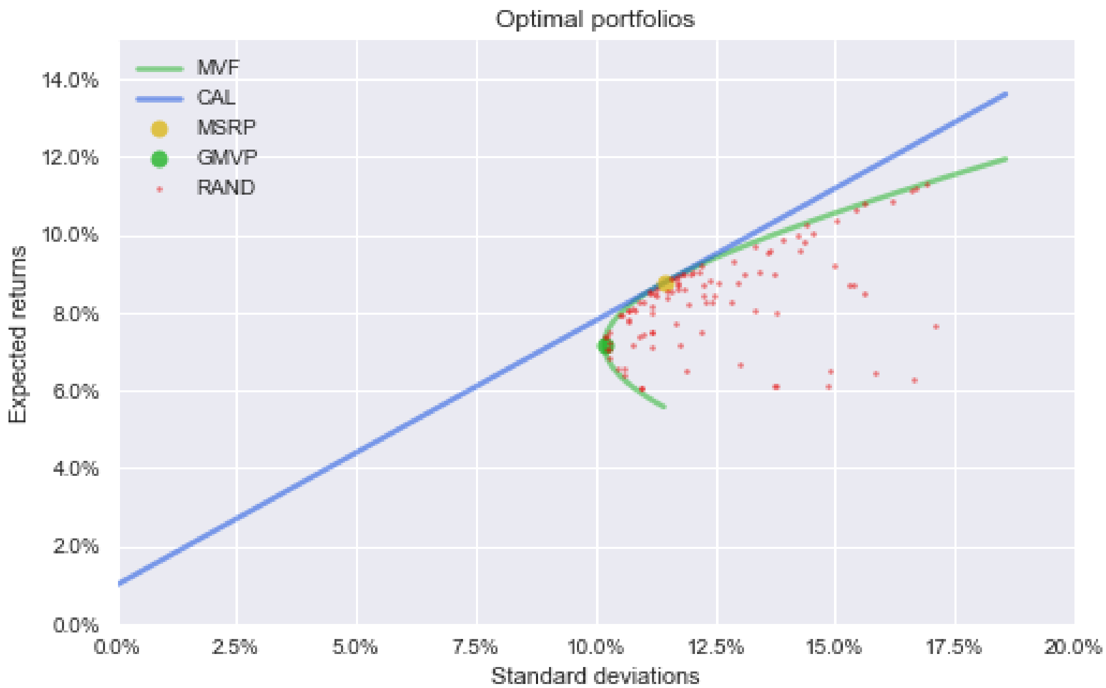
	A	B	C
0	0.387411	0.408411	0.204178
1	0.658744	0.329669	0.011587
2	0.537214	0.114096	0.348690
3	0.628863	0.119335	0.251802
4	0.303298	0.326451	0.370252

[18]:
```
ret_RAND = []
sig_RAND = []

plt.style.use('seaborn')
fig = plt.figure(figsize=(8, 5))
ax = fig.add_subplot(1,1,1)
plt.plot(sig_MVF, ret_MVF, color='xkcd:green', linewidth=2.5, alpha=0.5)
plt.plot(CAL_x, CAL_y, color='xkcd:blue', linewidth=2.5, alpha=0.5)
plt.plot(sig_MSRP, ret_MSRP, 'o', color='xkcd:gold', markersize=8, alpha=0.75)
plt.plot(sig_GMVP, ret_GMVP, 'o', color='xkcd:green', markersize=8, alpha=0.75)

for row in range(0,len(W_RAND)):
    W_TMP = W_RAND.iloc[row, :]
    ret_RAND = (np.dot(W_TMP, R))
    sig_RAND = (math.sqrt(np.dot(W_TMP, np.dot(COV, W_TMP))))
    plt.plot(sig_RAND, ret_RAND, 'o', color='xkcd:red', markersize=2.5, alpha=0.
↪5)

plt.ylabel('Expected returns')
plt.xlabel('Standard deviations')
ax.yaxis.set_major_formatter(mtick.PercentFormatter(1.0))
ax.xaxis.set_major_formatter(mtick.PercentFormatter(1.0))
plt.legend(['MVF', 'CAL', 'MSRP', 'GMVP', 'RAND'], loc='best')
plt.title('Optimal portfolios')
plt.xlim([0, 0.2])
plt.ylim([0, 0.15])
plt.show()
```



References

[1] Harry Markowitz (1952) "Portfolio Selection", *The Journal of Finance*, 7(1): 77–91.