Lecture 9 CAPM-implied MVF

Douglas Chung

National Chengchi University
19 November 2021

1 CAPM and portfolio optimizations

In this JupyterNotebook, we will construct the MVF implied by CAPM.

[1]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
import pandas_datareader as pdr

factor = pdr.get_data_famafrench('F-F_Research_Data_Factors', start='1-1-1926')
asset = pdr.get_data_famafrench('10_Industry_Portfolios', start='1-1-1926')

N = asset[0].shape[1]

df = asset[0].join(factor[0])
df = df/100
df
```

[1]:

Date	NoDur	Durbl	Manuf	Enrgy	HiTec	Telcm	Shops	Hlth
1926-07	0.0145	0.1555	0.0469	-0.0118	0.0290	0.0083	0.0011	0.0177
1926-08	0.0397	0.0368	0.0281	0.0347	0.0266	0.0217	-0.0071	0.0425
1926-09	0.0114	0.0480	0.0115	-0.0339	-0.0038	0.0241	0.0021	0.0069
1926-10	-0.0124	-0.0823	-0.0363	-0.0078	-0.0458	-0.0011	-0.0229	-0.0057
1926-11	0.0520	-0.0019	0.0410	0.0001	0.0471	0.0163	0.0643	0.0542
...	...	...	...	...	...	...	...	...
2021-03	0.0721	0.0059	0.0754	0.0227	0.0074	0.0159	0.0554	0.0013
2021-04	0.0336	0.0437	0.0254	0.0071	0.0655	0.0315	0.0706	0.0289
2021-05	0.0193	-0.0526	0.0270	0.0613	-0.0087	-0.0083	-0.0221	0.0002
2021-06	-0.0074	0.0566	-0.0080	0.0550	0.0694	-0.0012	0.0284	0.0428
2021-07	0.0020	-0.0069	0.0134	-0.0852	0.0341	0.0028	0.0026	0.0315

Date	Utils	Other	Mkt-RF	SMB	HML	RF
1926-07	0.0704	0.0213	0.0296	-0.0238	-0.0273	0.0022

```
1926-08 -0.0169  0.0435  0.0264 -0.0147  0.0414  0.0025
1926-09  0.0204  0.0029  0.0036 -0.0139  0.0012  0.0023
1926-10 -0.0263 -0.0284 -0.0324 -0.0013  0.0065  0.0032
1926-11  0.0371  0.0211  0.0253 -0.0016 -0.0038  0.0031
...         ...     ...     ...     ...     ...     ...
2021-03  0.1035  0.0563  0.0308 -0.0241  0.0741  0.0000
2021-04  0.0398  0.0582  0.0493 -0.0311 -0.0093  0.0000
2021-05 -0.0117  0.0294  0.0029 -0.0028  0.0704  0.0000
2021-06 -0.0142 -0.0246  0.0279  0.0180 -0.0776  0.0000
2021-07  0.0294 -0.0069  0.0120 -0.0396 -0.0170  0.0000

[1141 rows x 14 columns]
```

1.0.1 sample GMVP

[2]:
```
from numpy.linalg import inv
import math
R = df.iloc[:,0:N]
ER = R.mean()
SD = R.std()
COV = R.cov()
RX = R.subtract(df.RF, axis=0)
ERX = RX.mean()

ONE = np.ones(N)
InvCOV = inv(COV)
W_GMVP = np.dot(InvCOV, ONE)/np.dot(ONE.T, np.dot(InvCOV, ONE))
ret_GMVP = np.dot(W_GMVP.T, ER)
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:
[ 0.73355657 -0.07842116 -0.0968838   0.16638474 -0.09693082  0.53677273
 -0.03668449  0.09494554  0.11102221 -0.33376154]
ret_GMVP:
0.008931041516669467
sig_GMVP:
0.03768935422468202
```

1.0.2 Sample MSRP

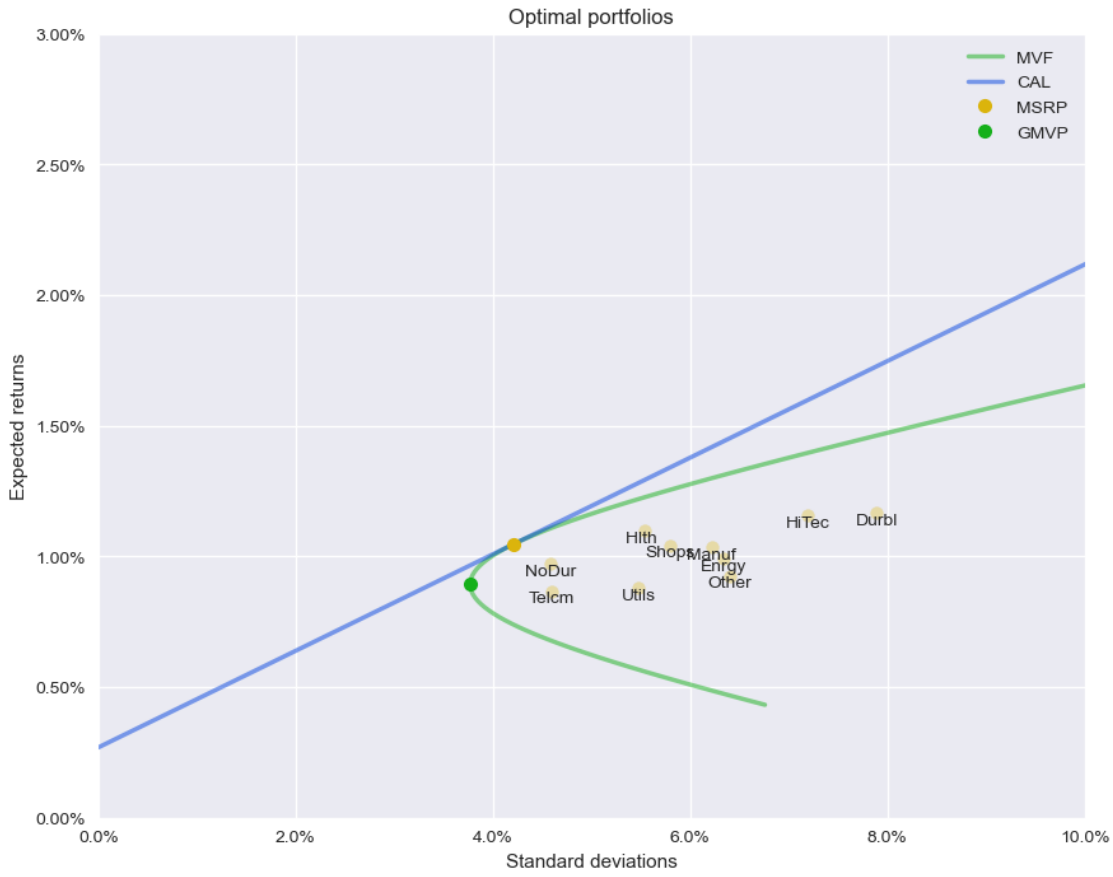
[3]:
```
ONE = np.ones(N)
InvCOV = inv(COV)
W_MSRP = np.dot(InvCOV, ERX)/np.dot(ONE.T, np.dot(InvCOV, ERX))
ret_MSRP = np.dot(W_MSRP, ER)
sig_MSRP =  math.sqrt(np.dot(W_MSRP, np.dot(COV, W_MSRP)))

rf = df.RF.mean()
SR_MSRP = (ret_MSRP - rf)/sig_MSRP

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

```
MSRP:
[ 0.73773798  0.11021237 -0.181763    0.22236982  0.10468745  0.29730098
  0.01255563  0.34421546  0.01431749 -0.66163418]
ret_MSRP:
0.010467390687530404
sig_MSRP:
0.04207300815985203
SR_MSRP:
0.18486692817995645
```

1.0.3 Sample MVF

[11]:
```
W = np.linspace(-4, 4, 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 = rf + SR_MSRP*CAL_x

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

plt.style.use('seaborn')
```

```
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(1,1,1)
plt.plot(sig_MVF, ret_MVF, color='xkcd:green', linewidth=2.5, alpha=0.5,␣
 ↪label='MVF')
plt.plot(CAL_x, CAL_y, color='xkcd:blue', linewidth=2.5, alpha=0.5, label='CAL')
plt.plot(sig_MSRP, ret_MSRP, 'o', color='xkcd:gold', markersize=8, label='MSRP')
plt.plot(sig_GMVP, ret_GMVP, 'o', color='xkcd:green', markersize=8, label='GMVP')


ax.scatter(SD, ER, color='xkcd:beige')
n = ER.index
for i, txt in enumerate(n):
    ax.annotate(txt, (SD[i], ER[i]),
                horizontalalignment='center',
                verticalalignment='top')

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(loc='best')
plt.title('Optimal portfolios')
plt.xlim([0, 0.1])
plt.ylim([0, 0.03])
plt.show()
```



[5]:
```
BETA = []
AvgR = []
SSR = []
for c in range(N):
    tmp_df = df.iloc[:,c]
    tmp_df = tmp_df.subtract(df.RF, axis=0)
    reg_df = pd.DataFrame([tmp_df, df['Mkt-RF']], index=['R', 'MKT']).T
    reg = smf.ols('R ~ 1 + MKT', data=reg_df).fit()
    BETA.append(reg.params[1])
    tmp_avg = reg_df.mean()
    AvgR.append(tmp_avg[0])
    SSR.append(reg.ssr/reg.nobs)
B = np.matmul(np.array([BETA]).T, np.array([BETA]))
SIG = B * df['Mkt-RF'].var() + np.diag(SSR)
```

1.0.4 Sample variance-covariance matrix

N variances

$\frac{N(N-1)}{2}$ covariances

[6]:
```
COV
```

[6]:

	NoDur	Durbl	Manuf	Enrgy	HiTec	Telcm	Shops	\
NoDur	0.002101	0.002635	0.002416	0.001791	0.002420	0.001447	0.002293	
Durbl	0.002635	0.006223	0.004198	0.003045	0.004345	0.002285	0.003604	
Manuf	0.002416	0.004198	0.003866	0.002864	0.003841	0.001973	0.003054	
Enrgy	0.001791	0.003045	0.002864	0.004022	0.002745	0.001535	0.002143	
HiTec	0.002420	0.004345	0.003841	0.002745	0.005165	0.002256	0.003317	
Telcm	0.001447	0.002285	0.001973	0.001535	0.002256	0.002115	0.001815	
Shops	0.002293	0.003604	0.003054	0.002143	0.003317	0.001815	0.003364	
Hlth	0.002010	0.002741	0.002606	0.001955	0.002866	0.001543	0.002389	
Utils	0.001770	0.002563	0.002351	0.002046	0.002410	0.001574	0.002053	
Other	0.002467	0.004021	0.003618	0.002810	0.003677	0.002089	0.003067	

	Hlth	Utils	Other
NoDur	0.002010	0.001770	0.002467
Durbl	0.002741	0.002563	0.004021
Manuf	0.002606	0.002351	0.003618
Enrgy	0.001955	0.002046	0.002810
HiTec	0.002866	0.002410	0.003677
Telcm	0.001543	0.001574	0.002089
Shops	0.002389	0.002053	0.003067
Hlth	0.003062	0.001861	0.002612
Utils	0.001861	0.003001	0.002503
Other	0.002612	0.002503	0.004103

1.0.5 Variance-covariance matrix under single factor model

[7]:
```
SIG_F = pd.DataFrame(SIG, index=COV.index, columns= COV.columns)
SIG_F
```

[7]:

	NoDur	Durbl	Manuf	Enrgy	HiTec	Telcm	Shops	\
NoDur	0.002101	0.002719	0.002405	0.001943	0.002633	0.001432	0.002076	
Durbl	0.002719	0.006252	0.004043	0.003265	0.004425	0.002406	0.003489	
Manuf	0.002405	0.004043	0.003884	0.002888	0.003914	0.002129	0.003087	
Enrgy	0.001943	0.003265	0.002888	0.004029	0.003161	0.001719	0.002493	
HiTec	0.002633	0.004425	0.003914	0.003161	0.005187	0.002330	0.003379	
Telcm	0.001432	0.002406	0.002129	0.001719	0.002330	0.002115	0.001837	
Shops	0.002076	0.003489	0.003087	0.002493	0.003379	0.001837	0.003376	
Hlth	0.001795	0.003016	0.002668	0.002155	0.002920	0.001588	0.002303	
Utils	0.001643	0.002762	0.002443	0.001973	0.002674	0.001454	0.002109	
Other	0.002413	0.004056	0.003588	0.002898	0.003927	0.002136	0.003097	

	Hlth	Utils	Other
NoDur	0.001795	0.001643	0.002413
Durbl	0.003016	0.002762	0.004056
Manuf	0.002668	0.002443	0.003588

```
Enrgy  0.002155  0.001973  0.002898
HiTec  0.002920  0.002674  0.003927
Telcm  0.001588  0.001454  0.002136
Shops  0.002303  0.002109  0.003097
Hlth   0.003066  0.001823  0.002677
Utils  0.001823  0.002999  0.002451
Other  0.002677  0.002451  0.004114
```

1.0.6 GMVP under single factor model

[8]:
```
ONE = np.ones(N)
InvSIG = inv(SIG_F)
W_GMVP_F = np.dot(InvSIG, ONE)/np.dot(ONE.T, np.dot(InvSIG, ONE))
ret_GMVP_F = np.dot(W_GMVP_F.T, ER)
sig_GMVP_F = math.sqrt(np.dot(W_GMVP_F, np.dot(SIG, W_GMVP_F)))

print('GMVP_F:')
print(W_GMVP_F)
print('ret_GMVP_F:')
print(ret_GMVP_F)
print('sig_GMVP_F:')
print(sig_GMVP_F)
```

```
GMVP_F:
[ 0.68592104 -0.15366159 -0.29470703  0.092872   -0.23519847  0.50877519
  0.1208152   0.22021438  0.23917428 -0.184205  ]
ret_GMVP_F:
0.008504525011678702
sig_GMVP_F:
0.03462254240244776
```

1.0.7 MSRP under single factor model

[9]:
```
ONE = np.ones(N)
InvSIG = inv(SIG_F)
W_MSRP_F = np.dot(InvSIG, ERX)/np.dot(ONE.T, np.dot(InvSIG, ERX))
ret_MSRP_F = np.dot(W_MSRP_F, ER)
sig_MSRP_F =  math.sqrt(np.dot(W_MSRP_F, np.dot(SIG, W_MSRP_F)))

SR_MSRP_F = (ret_MSRP_F - rf)/sig_MSRP_F

print('MSRP_F:')
print(W_MSRP_F)
print('ret_MSRP_F:')
print(ret_MSRP_F)
print('sig_MSRP_F:')
```

```
print(sig_MSRP_F)
print('SR_MSRP_F:')
print(SR_MSRP_F)
```

```
MSRP_F:
[ 0.68818828 -0.01241665 -0.24988741  0.08963765  0.01783771  0.29202128
  0.23475821  0.44069177  0.09588786 -0.59671869]
ret_MSRP_F:
0.010181296622307573
sig_MSRP_F:
0.039298503177652065
SR_MSRP_F:
0.1906386530477231
```

1.0.8 Comparison

[10]:
```
W = np.linspace(-4, 4, 100, endpoint=True)
ret_MVF_F = []
sig_MVF_F = []
for u in W:
    ret_MVF_F.append(u*ret_GMVP_F + (1-u)*ret_MSRP_F)
    sig_MVF_F.append(math.sqrt(u**2 * sig_GMVP_F**2 + (1-u)**2 * sig_MSRP_F**2 +␣
 ↪2*u*(1-u)*np.dot(W_GMVP_F, np.dot(SIG, W_MSRP_F))))

CAL_x_F = np.linspace(0, np.max(sig_MVF_F), 50, endpoint=True)
CAL_y_F = rf + SR_MSRP_F*CAL_x_F

fig = plt.figure(figsize=(10, 8))
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)

plt.plot(sig_MVF_F, ret_MVF_F, color='xkcd:violet', linewidth=2.5, alpha=0.5)
plt.plot(CAL_x_F, CAL_y_F, color='xkcd:gray', linewidth=2.5, alpha=0.5)
plt.plot(sig_MSRP_F, ret_MSRP_F, 'o', color='xkcd:orange', markersize=8)
plt.plot(sig_GMVP_F, ret_GMVP_F, 'o', color='xkcd:red', 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','MVF_F', 'CAL_F', 'MSRP_F', 'GMVP_F'],␣
 ↪loc='best')
plt.title('Optimal portfolios')
```

```
plt.xlim([0, 0.1])
plt.ylim([0, 0.03])
plt.show()
```