**Part I. Markowitz Optimization:**

An efficient portfolio is a specific portfolio that make investors with the same risk aversion get the maximum return and minimum risk for that return. How to choose among a variety of assets to get the efficient portfolio is an optimal problem. One formulation of this problem is:

**with**

**and**

The above formulations simply state that we need to minimum the variance and get a specific return at meantime, and sum of weights of different assets should equal to one, those weights are not negative. This is a classic problem in finance known as mean-variance optimization and was first postulated by Harry Markowitz and requires knowledge of the financial theory, mathematics, and computing to solve.

The problem as laid out is a single-period optimization problem with rather simplistic constraints. Real world problems typically include additional assumptions/constraints such as:

* An ability to short sell – according to the above formulations, the weights of any assets cannot be negative. This means that we cannot do short sell for achieving the optimization.
* Limits on asset weights – any single asset cannot have a weight that exceeds a specific threshold. This constraints are mainly for the diversification purpose.
* Limits on sector or industry weights – this constraints are also for diversification. But it is in higher level. For example, if China put tariffs on cars, the auto industry may suffer a big loss and the stocks in this sector would dramatically drop. If we put all assets in this sector, we may loss all wealth. In order to avoiding this kind of things happening, we should set limits on sector or industry.

By running the R code, we get 12 graphs, and each two for each case. Among six cases, case 1 and case 4 are the same. Case 1-3 have different expected return but the same covariance matrix. Case 4-6 have different covariance matrix but the same expected return. So we analyze them as two different parts.

Case 1-3:

The following six graphs show the efficient frontier and the weights of different assets for case 1-3. According to the graphs, the change of expected returns does not only change the shape of the efficient frontier, but also change the weights of different assets a lot. However, we find that the change of expected return does not affect the minimum variance portfolio, and it only makes the shape of efficient frontier steeper.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

Case 4-6:

According to the following graphs, we find that the shape of efficient frontier does not change a lot, but whole efficient frontier moves to the left from case 4 to case 5, and then moves back from case 5 to case 6. In terms of the weights of different assets, change of covariance matrix does not change the weights a lot (at least not like the effects from the change of expected returns).

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

Totally speaking, we find that the model is more sensitive to the change of expected returns. And the change of expected returns would dramatically change the weights of different assets, and also the shape of efficient frontier.

**Part2:**

**Step 1 Market Equilibrium Weights**

Calculate weight of each stock by market cap in stock portfolio with the formula below.

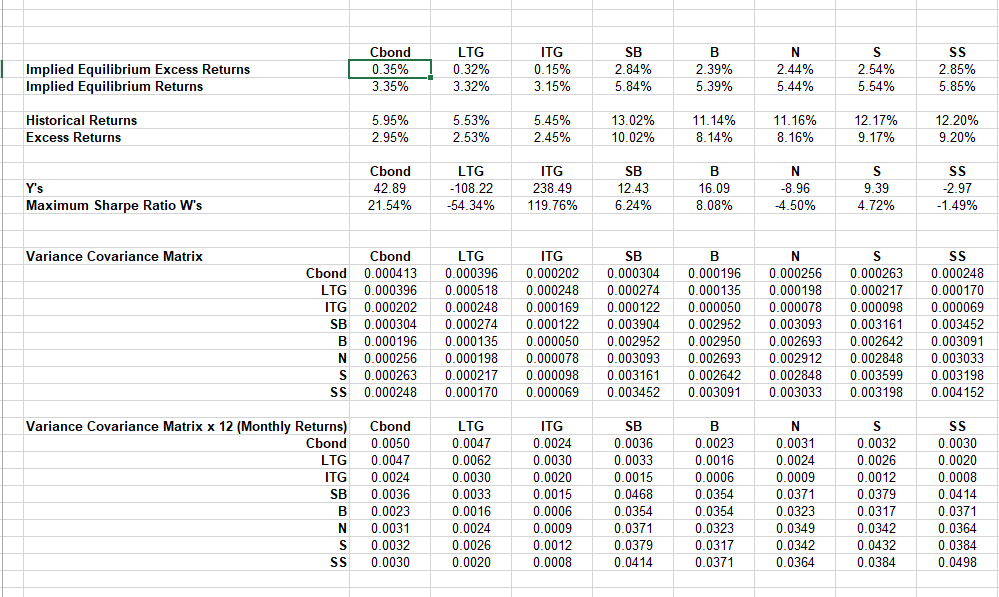


Then, we allocate 1/3 bonds(50% Cbond,25% LTG and 25%ITG) and 2/3 stocks （by market cap）for the portfolio. Then we get market equilibrium weights below:



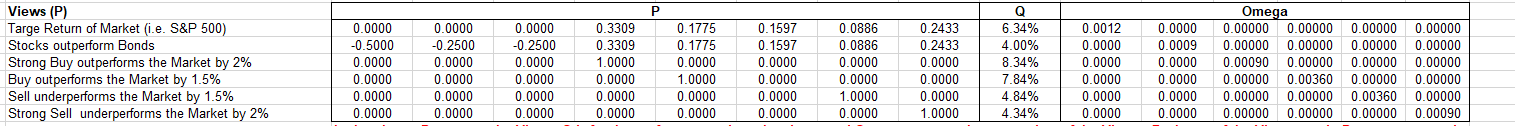
**Step 2 Weights by maxmizing sharp ratio**

First use historical return minus risk free rate, we can get excess return. Then use the excess return matrix mutiply by varience matrix to get the Y’s. The weight of maxmizing sharp ratio is equal to each kind of securites’ Y’s divided by the sum of Y’s.

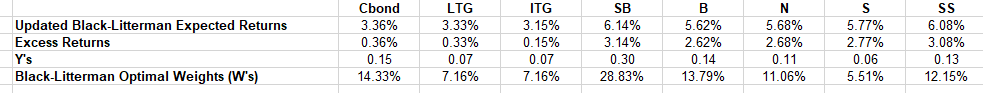


**Step 3 Black-Litterman Optimal Weights and excess returns**

Use Black-Litterman model to take views and performance given views into account when calculate excess returns and weights. P represents views , q represents the performance given views and Omega is the uncertainty matrix.



Excess return equals to the updated return minus risk free rate. Then use the updated excess return matrix multiply by the variance matrix to get the new Y’s and weights.



**Step 4 comparison of two results**

From the bar chart below, we can see that the weight of portfolio has changed a lot after considering views. What’s more, all the weights number become positive and normal while the origin weights are much more extreme and concentrate. Views and expectations are one of the strongest factors that affect the asset returns. Considering the views of the market into the portfolio construction makes the result more reasonable and also easier to execute. In a word, Black-Litterman model is a very good approach to improve Markowitz theory. But the data of market views may change anytime and sometimes hard to evaluate, this is the biggest challenge of this model.

**Python Code:**

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

In [259]:

Market\_Capitalization=[4338170.95,2327751.06,2094125.22,1162394.4,3189735.14]

Rf=0.03

Lambda=1

In [260]:

SUM\_MKT=np.sum(Market\_Capitalization)

Market\_Capitalization\_Weights=Market\_Capitalization/SUM\_MKT

In [261]:

Market\_Equilibrium\_Weights=[0.25\*2/3,0.25\*1/3,0.25\*1/3]

a=Market\_Capitalization\_Weights\*2/3

a=list(a)

Market\_Equilibrium\_Weights=Market\_Equilibrium\_Weights+a

Market\_Equilibrium\_Weights=np.asarray(Market\_Equilibrium\_Weights)

Market\_Equilibrium\_Weights

Out[261]:

array([ 0.16666667, 0.08333333, 0.08333333, 0.22056704, 0.1183506 ,

0.10647229, 0.0591 , 0.16217674])

In [262]:

Variance\_Covariance\_Matrix= np.matrix([[0.000412966199505151,0.000395768587657186,0.000202396498777172,0.000304084572915505,0.00019562014898352,0.000255722526398289,0.000263479868666769,0.000248382123204202],

[0.000395768587657186,0.000518246145254603,0.000247609148564934,0.000273612202288466,0.000134781935224511,0.000197967785313568,0.000216888577332329,0.00016997601362203],

[0.000202396498777172,0.000247609148564934,0.000168797909742308,0.000121553891868696,0.0000499931118715226,0.0000780136513372639,0.0000976988620733828,0.0000694461440652289],

[0.000304084572915505,0.000273612202288466,0.000121553891868696,0.00390409056646803,0.00295246275880029,0.00309341635661538,0.00316077650268375,0.00345216498678725],

[0.00019562014898352,0.000134781935224511,0.000049993111871523,0.00295246275880029,0.00294973957731073,0.00269271973090167,0.0026423271046472,0.00309115928976052],

[0.000255722526398289,0.000197967785313568,0.0000780136513372639,0.00309341635661538,0.00269271973090167,0.00291218603263429,0.00284787861623656,0.00303284456996285],

[0.000263479868666769,0.000216888577332329,0.0000976988620733828,0.00316077650268375,0.0026423271046472,0.00284787861623656,0.0035987334705483,0.00319774018217517],

[0.000248382123204202,0.00016997601362203,0.0000694461440652289,0.00345216498678725,0.00309115928976052,0.00303284456996285,0.00319774018217517,0.00415177264264004]]

)

In [263]:

Variance\_Covariance\_Matrix\_12=Variance\_Covariance\_Matrix\*12

print(Variance\_Covariance\_Matrix\_12)

[[ 0.00495559 0.00474922 0.00242876 0.00364901 0.00234744 0.00306867

0.00316176 0.00298059]

[ 0.00474922 0.00621895 0.00297131 0.00328335 0.00161738 0.00237561

0.00260266 0.00203971]

[ 0.00242876 0.00297131 0.00202557 0.00145865 0.00059992 0.00093616

0.00117239 0.00083335]

[ 0.00364901 0.00328335 0.00145865 0.04684909 0.03542955 0.037121

0.03792932 0.04142598]

[ 0.00234744 0.00161738 0.00059992 0.03542955 0.03539687 0.03231264

0.03170793 0.03709391]

[ 0.00306867 0.00237561 0.00093616 0.037121 0.03231264 0.03494623

0.03417454 0.03639413]

[ 0.00316176 0.00260266 0.00117239 0.03792932 0.03170793 0.03417454

0.0431848 0.03837288]

[ 0.00298059 0.00203971 0.00083335 0.04142598 0.03709391 0.03639413

0.03837288 0.04982127]]

In [264]:

*#Variance\_Covariance\_Matrix = pd.read\_csv("/Users/jane/Desktop/Book2.csv",header=-1)*

In [265]:

trans\_weights=np.matrix.transpose(Market\_Equilibrium\_Weights)

trans\_weights

MMULT=np.matmul(Variance\_Covariance\_Matrix\_12,trans\_weights)

*#print(MMULT)*

Implied\_Equilibrium\_Excess\_Returns=np.transpose(MMULT\*Lambda)

print(Implied\_Equilibrium\_Excess\_Returns)

[[ 0.00350374]

[ 0.00321056]

[ 0.00151804]

[ 0.02844212]

[ 0.02390996]

[ 0.02444212]

[ 0.02537421]

[ 0.02848613]]

In [266]:

Implied\_Equilibrium\_Returns=Implied\_Equilibrium\_Excess\_Returns+Rf

print(Implied\_Equilibrium\_Returns)

[[ 0.03350374]

[ 0.03321056]

[ 0.03151804]

[ 0.05844212]

[ 0.05390996]

[ 0.05444212]

[ 0.05537421]

[ 0.05848613]]

In [267]:

Historical\_Returns=[0.0595249784482758,0.0552711206896553,0.0544693577586207,0.130160964946643,0.111389192314337,0.111565419466273,0.121709482758621,0.121984358077008]

Historical\_Returns=np.matrix(Historical\_Returns)

Excess\_Returns=Historical\_Returns-Rf

Excess\_Returns=Excess\_Returns.transpose()

inverse = np.linalg.inv(Variance\_Covariance\_Matrix)

Y=np.matmul(inverse,Excess\_Returns).transpose()

SUM\_Y=Y.sum()

Maximum\_Sharpe\_Ratio\_W=Y/SUM\_Y

Maximum\_Sharpe\_Ratio\_W

Out[267]:

matrix([[ 0.2153751 , -0.54342113, 1.19758959, 0.06240986, 0.08080607,

-0.04498861, 0.04716242, -0.01493331]])

In [268]:

P=[[0.0000,0.0000,0.0000,0.3309,0.1775,0.1597,0.0886,0.2433],

[-0.5000,-0.2500,-0.2500,0.3309,0.1775,0.1597,0.0886,0.2433],

[0.0000,0.0000,0.0000,1.0000,0.0000,0.0000,0.0000,0.0000],

[0.0000,0.0000,0.0000,0.0000,1.0000,0.0000,0.0000,0.0000],

[0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,1.0000,0.0000],

[0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,1.0000]]

In [269]:

Q=np.matrix([0.0634,0.04,0.0834,0.0784,0.0484,0.0434])

Q=np.transpose(Q)

Omega=[[0.0012,0.0000,0.00000,0.00000,0.00000,0.00000],

[0.0000,0.0009,0.00000,0.00000,0.00000,0.00000],

[0.0000,0.0000,0.00090,0.00000,0.00000,0.00000],

[0.0000,0.0000,0.00000,0.00360,0.00000,0.00000],

[0.0000,0.0000,0.00000,0.00000,0.00360,0.00000],

[0.0000,0.0000,0.00000,0.00000,0.00000,0.00090]]

In [270]:

b1=np.matmul(Variance\_Covariance\_Matrix\_12\*0.0025,np.transpose(P))

b1

Out[270]:

matrix([[ 7.79876208e-06, -2.88196906e-06, 9.12253719e-06,

5.86860447e-06, 7.90439606e-06, 7.45146370e-06],

[ 6.19947056e-06, -5.48097296e-06, 8.20836607e-06,

4.04345806e-06, 6.50665732e-06, 5.09928041e-06],

[ 2.61321319e-06, -3.54578723e-06, 3.64661676e-06,

1.49979336e-06, 2.93096586e-06, 2.08338432e-06],

[ 1.02897025e-04, 9.53720109e-05, 1.17122717e-04,

8.85738828e-05, 9.48232951e-05, 1.03564950e-04],

[ 8.75029584e-05, 8.31828433e-05, 8.85738828e-05,

8.84921873e-05, 7.92698131e-05, 9.27347787e-05],

[ 8.87057539e-05, 8.28000552e-05, 9.28024907e-05,

8.07815919e-05, 8.54363585e-05, 9.09853371e-05],

[ 9.19973458e-05, 8.56857420e-05, 9.48232951e-05,

7.92698131e-05, 1.07962004e-04, 9.59322055e-05],

[ 1.04063805e-04, 9.85424073e-05, 1.03564950e-04,

9.27347787e-05, 9.59322055e-05, 1.24553179e-04]])

In [271]:

b2=np.linalg.inv((np.matmul(np.matmul(P,Variance\_Covariance\_Matrix\_12\*0.0025),np.transpose(P))+Omega))

b2

Out[271]:

matrix([[ 787.25225682, -56.98269892, -65.08598908, -13.87382471,

-14.56553142, -65.28185926],

[ -56.98269892, 1029.14541609, -79.20308103, -17.51197142,

-17.84868745, -81.93399435],

[ -65.08598908, -79.20308103, 1009.20085813, -18.36028903,

-19.79034583, -84.26962205],

[ -13.87382471, -17.51197142, -18.36028903, 272.8536547 ,

-4.11378232, -19.36206168],

[ -14.56553142, -17.84868745, -19.79034583, -4.11378232,

271.57160373, -19.8591827 ],

[ -65.28185926, -81.93399435, -84.26962205, -19.36206168,

-19.8591827 , 1002.67660417]])

In [272]:

b3=Q-np.matmul(P,Implied\_Equilibrium\_Returns)

In [273]:

b4=np.matmul(b2,b3)

np.transpose(np.matmul(b1,b4))

Updated\_Black\_Litterman\_Expected\_Returns=np.transpose(Implied\_Equilibrium\_Returns)+np.transpose(np.matmul(b1,b4))

In [274]:

Updated\_Black\_Litterman\_Expected\_Returns

Out[274]:

matrix([[ 0.03358137, 0.03325853, 0.03152352, 0.0614123 , 0.0561927 ,

0.05679685, 0.05767069, 0.0607946 ]])

In [275]:

Excess\_Returns\_2=Updated\_Black\_Litterman\_Expected\_Returns-Rf

In [276]:

inverse\_2 = np.linalg.inv(Variance\_Covariance\_Matrix\_12)

Y\_2=np.matmul(inverse\_2,np.transpose(Excess\_Returns\_2)).transpose()

print(Y\_2)

SUM\_Y\_2=Y\_2.sum()

Black\_Litterman\_Optimal\_Weights=Y\_2/SUM\_Y\_2

Black\_Litterman\_Optimal\_Weights

[[ 0.14761214 0.07380607 0.07380607 0.29710351 0.14206507 0.11393341

0.05680137 0.12522529]]

Out[276]:

matrix([[ 0.14326367, 0.07163183, 0.07163183, 0.2883512 , 0.13788 ,

0.11057707, 0.05512807, 0.12153631]])

In [283]:

print(('Maximum Sharpe Ratio Weights'),Maximum\_Sharpe\_Ratio\_W)

print(('Black-Litterman Optimal Weights'),Black\_Litterman\_Optimal\_Weights)

Maximum Sharpe Ratio Weights [[ 0.2153751 -0.54342113 1.19758959 0.06240986 0.08080607 -0.04498861

0.04716242 -0.01493331]]

Black-Litterman Optimal Weights [[ 0.14326367 0.07163183 0.07163183 0.2883512 0.13788 0.11057707

0.05512807 0.12153631]]

**R code:**

variance\_covariance\_12<-c(0.0050, 0.0047, 0.0024, 0.0036, 0.0023, 0.0031, 0.0032, 0.0030,

0.0047, 0.0062, 0.0030, 0.0033, 0.0016, 0.0024, 0.0026, 0.0020,

0.0024, 0.0030, 0.0020, 0.0015, 0.0006, 0.0009, 0.0012, 0.0008,

0.0036, 0.0033, 0.0015, 0.0468, 0.0354, 0.0371, 0.0379, 0.0414,

0.0023, 0.0016, 0.0006, 0.0354, 0.0354, 0.0323, 0.0317, 0.0371,

0.0031, 0.0024, 0.0009, 0.0371, 0.0323, 0.0349, 0.0342, 0.0364,

0.0032, 0.0026, 0.0012, 0.0379, 0.0317, 0.0342, 0.0432, 0.0384,

0.0030, 0.0020, 0.0008, 0.0414, 0.0371, 0.0364, 0.0384, 0.0498)

variance\_covariance\_12

variance\_covariance\_matrix\_12<-matrix(variance\_covariance\_12,nrow=8,byrow=TRUE)

variance\_covariance\_matrix\_12

market\_weight\_vector<-c(0.166666667, 0.083333333, 0.083333333, 0.220567036 ,0.118350604, 0.106472289 ,0.059099996, 0.162176741)

market\_weight\_matrix<-matrix(market\_weight\_vector,nrow=1,byrow=TRUE)

market\_weight\_matrix

implied\_excess\_return<-t(variance\_covariance\_matrix\_12%\*%t(market\_weight\_matrix))

implied\_return<- implied\_excess\_return+0.03

variance\_covariance<-c(0.000412966199505151, 0.000395768587657186, 0.000202396498777172, 0.000304084572915505, 0.000195620148983520 , 0.000255722526398289, 0.000263479868666769, 0.000248382123204202,

0.000395768587657186, 0.000518246145254603, 0.000247609148564934, 0.000273612202288466, 0.000134781935224511 , 0.000197967785313568, 0.000216888577332329, 0.000169976013622030,

0.000202396498777172, 0.000247609148564934, 0.000168797909742308, 0.000121553891868696, 0.000049993111871523 , 0.000078013651337264, 0.000097698862073383, 0.000069446144065229,

0.000304084572915505, 0.000273612202288466, 0.000121553891868696, 0.003904090566468030, 0.002952462758800290 , 0.003093416356615380, 0.003160776502683750, 0.003452164986787250,

0.000195620148983520, 0.000134781935224511, 0.000049993111871523, 0.002952462758800290, 0.002949739577310730 , 0.002692719730901670, 0.002642327104647200, 0.003091159289760520,

0.000255722526398289, 0.000197967785313568, 0.000078013651337264, 0.003093416356615380, 0.002692719730901670 , 0.002912186032634290, 0.002847878616236560, 0.003032844569962850,

0.000263479868666769, 0.000216888577332329, 0.000097698862073383, 0.003160776502683750, 0.002642327104647200 , 0.002847878616236560, 0.003598733470548300, 0.003197740182175170,

0.000248382123204202, 0.000169976013622030, 0.000069446144065229, 0.003452164986787250, 0.003091159289760520 , 0.003032844569962850, 0.003197740182175170, 0.004151772642640040 )

variance\_covariance\_matrix<-matrix(variance\_covariance,nrow=8,byrow=TRUE)

variance\_covariance\_matrix

excess\_return\_vector<-c(0.0295 ,0.0253, 0.0245 ,0.1002 ,0.0814 ,0.0816,0.0917 ,0.0920)

excess\_return\_matrix <-matrix(excess\_return\_vector,nrow=1,byrow=TRUE)

y<-t(solve(variance\_covariance\_matrix)%\*%t(excess\_return\_matrix))

rowSums(y)

maximum\_sharpe\_ratio<-c(y/rowSums(y))

maximum\_sharpe\_ratio

p\_vector<-c(0.0000, 0.0000, 0.0000, 0.3309, 0.1775, 0.1597, 0.0886, 0.2433,

-0.5000, -0.2500, -0.2500, 0.3309, 0.1775, 0.1597, 0.0886, 0.2433,

0.0000, 0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000,

0.0000, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 0.0000,

0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000,

0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000)

p\_matrix<- matrix(p\_vector,nrow=6,byrow=TRUE)

q\_vector<- c(0.0634, 0.0400, 0.0834, 0.0784, 0.0484, 0.0434 )

q\_matrix<- matrix(q\_vector,nrow=6,byrow=TRUE)

omega\_vector<-c(0.0012, 0.0000, 0.00000, 0.00000, 0.00000, 0.00000,

0.0000, 0.0009, 0.00000, 0.00000, 0.00000, 0.00000,

0.0000, 0.0000, 0.00090, 0.00000, 0.00000, 0.00000,

0.0000, 0.0000, 0.00000, 0.00360, 0.00000, 0.00000,

0.0000, 0.0000, 0.00000, 0.00000, 0.00360, 0.00000,

0.0000, 0.0000, 0.00000, 0.00000, 0.00000, 0.00090)

omega\_matrix<-matrix(omega\_vector,nrow=6,byrow=TRUE)

blm\_return<-implied\_return+t((variance\_covariance\_matrix\_12\*0.0025)%\*%t(p\_matrix)%\*%(solve(((p\_matrix%\*%(variance\_covariance\_matrix\_12\*0.0025))%\*%t(p\_matrix))+omega\_matrix))%\*%(q\_matrix-(p\_matrix%\*%t(implied\_return))))

blm\_excess\_return<-blm\_return-0.03

ys<-t(solve(variance\_covariance\_matrix\_12)%\*%t(blm\_excess\_return))

optimal\_weight<-c(ys/rowSums(ys))

optimal\_weight