Python Lab for Financial Engineering

Final Project

Po-Hsuan (Michael) Lin

Introduction

This project implements the Black-Litterman Model for global portfolio optimization, alongside the classic Markowitz minimum variance and maximum return portfolio. The model is introduced because the traditional Markowitz optimization often leaves investors with portfolios with large short position or portfolios that simply don’t follow the investors’ views. Black-Litterman Model avoids these problems by adjusting the underlying assumption. Instead of assuming that every assets’ expected return is known by investors, the model assumes that the market is in an equilibrium state. Using the CAPM model, investors can calculate the expected return with equilibrium market weights and incorporate their views into the calculation. Furthermore, its application is diverse. It can be utilized as an economics model or a portfolio optimization model. Therefore, by doing this project, I will be able to better understand the mechanisms behind this model and apply it to other fields.

Mathematical Details

For this section, we will first dive into the formulas used for the classic Markowitz optimization, then we can take a closer look at the Black Litterman Model.

One of the easiest way to optimize a portfolio is to first create a tangency portfolio. A tangency portfolio yields the most optimized weightings of all the risky assets in the portfolio and assuming there is no cash position. After calculating the tangency portfolio, minimum variance and maximum return portfolios can be easily derived from the tangency portfolio.

However, this is not the only way to construct minimum variance or maximum return portfolios. One can also apply the Lagrange Multiplier optimization directly to get their respective formulas. These direct formulas are not derived and used in this project. For more information, please refer to Numerical Linear Algebra for Financial Engineering by Dan Stefanica.

Markowitz Portfolio Optimization

Variables used in the formulation[1]:

μ: Expected return of all the assets in the portfolio

r: Risk-free rate or the short rate used in the money market account

$\bar{\mu}$: Expected excess return of all the assets in the portfolio

W: Weight of a given portfolio

μ_p: Expected rate of return of the portfolio; used as a major limiting factor in minimum variance portfolio

σ_p: Expected standard deviation, or volatility of the portfolio; used as a major limited factor in maximum return portfolio

$\sum R$: The covariance matrix of return of all the assets in the portfolio; this can be calculated with the standard deviation of each individual assets and the correlation between them

$\mathbb{1}$: one dimensional column vector with its elements being all one

Expected Return of the Portfolio given weights:

$$\mu_p = r + \bar{\mu}^t W$$

Expected standard deviation, or volatility of the portfolio given weights:

$$\sigma_p = \sqrt{W^t \sum R\, W}$$

Tangency Portfolio:

$$\bar{\mu} = \mu - r * \mathbb{1}$$

$$W_T = \frac{1}{\mathbb{1}^t \sum R^{-1} \bar{\mu}} \sum R^{-1}\, \bar{\mu}$$

Minimum Variance Portfolio with Cash Position:

$$W_{min,cash} = 1 - \frac{\mu_p - r}{\bar{\mu}^t W_T}$$

$$W_{min} = \left(1 - W_{min,cash}\right) * W_T$$

Minimum Variance Portfolio without Cash Position:

$$W_{min} = \frac{1}{\mathbb{1}^t \sum R^{-1} \mathbb{1}} \sum R^{-1}\, \mathbb{1}$$

[1] Note: Transpose of vector and matrix is denoted as the superscript t. Inverse of matrix is denoted with the superscript –1

Maximum Return Portfolio with Cash Position:

$$W_{max,cash} = 1 - \frac{\sigma_p}{\sqrt{W_T{}^t \sum R W_T}} * sign(\mathbb{1}^t \sum R^{-1} \bar{\mu})$$

$$W_{max} = \left(1 - W_{max,cash}\right) * W_T$$

Black-Litterman Portfolio Optimization

Variables used in the formulation:

W_{eq}: Weights of assets in equilibrium, or market weights

W_{BL}: Weights of assets in Black-Litterman model

Π_{eq}: Expected return of assets in equilibrium

Π_{BL}: The modified expected return of assets after incorporating investors' view

$\sum R$: The original covariance matrix of excess return of all the assets in the portfolio; this can be calculated with the standard deviation of each individual assets and the correlation between them

$\sum R_{BL}$: The modified covariance matrix of excess return of all the assets in the portfolio; this is the covariance matrix with views incorporated

δ: Risk aversion coefficients, denoted as λ in Morningstar report on Black-Litterman model

τ: A scalar measure that measures the uncertainty of CAPM

P: Matrix that shows the probability of views happening

Q: Matrix that represents investors' views

Ω: Investors' view uncertainty matrix; can also be seen as the diagonal covariance matrix of error terms from the expressed views, representing the uncertainty in each view

Equilibrium expected return:

$$\Pi_{eq} = \delta \sum R \, W_{eq}$$

Modified expected return after incorporating investors' view:

$$\Pi_{BL} = [(\tau \sum R)^{-1} + P^t \Omega^{-1} P]^{-1} [(\tau \sum R)^{-1} \Pi_{eq} + P^t \Omega^{-1} Q]$$

Can also be written in the following form, which is the one I used in the python implementation:

$$\Pi_{BL} = \Pi_{eq} + \tau \sum R \, P^t (P \tau \sum R \, P^t + \Omega)^{-1} (Q - P \Pi_{eq})$$ [2]

Modified covariance matrix of excess return:

$$\sum R_{BL} = \sum R + \tau \sum R - \tau \sum R \, P^t (P \tau \sum R \, P^t + \Omega)^{-1} P \tau \sum R$$

Modified weights with views incorporated:

$$W_{BL} = (\delta \sum R_{BL})^{-1} \Pi_{BL}$$

These formulas are developed to minimize the following objective function[3], as listed on page 9 of the paper (page 35 of the journal):

$$\min_{\Pi} (\Pi_{BL} - \Pi_{eq}) \tau \sum R^{-1} (\Pi_{BL} - \Pi_{eq})^t$$

[2] In the actual implementation, this formula is separated into two terms since it is too complicated to write it in a single line. I have to separately test the output to ensure that the dot product, inverse and transpose are all working as expected

[3] In the paper, the formula is written as $P * \Pi_{BL}{}^t = Q$, which is under the assumption that investors are 100% confidence in their views. This assumption is unrealistic and is used for simplicity sake. Thus, the formula I computed above incorporated the error matrix, Ω

Subject to this constraint with the error term:
$P * \Pi_{BL}{}^{t} = Q + \varepsilon$

Python Implementation

Part I Markowitz Portfolio Optimization

Data Selection and Processing

The data selected for this project is stock prices from the following companies (shown with ticker symbols): BAC, BNS, C, GS, JPM, MS, TD, USB and WFC. This data set is between January 4th, 2016 and July 29th, 2016. It is chosen at random.

The first step is to input and process the data. Pandas data frame is used to store the data and log return is found for the return as it is more commonly used than percentage change. Once the log return is found, covariance matrix of the return can be calculated, using the built-in Pandas covariance attributes. A covariance heat map can be generated using Seaborn package.

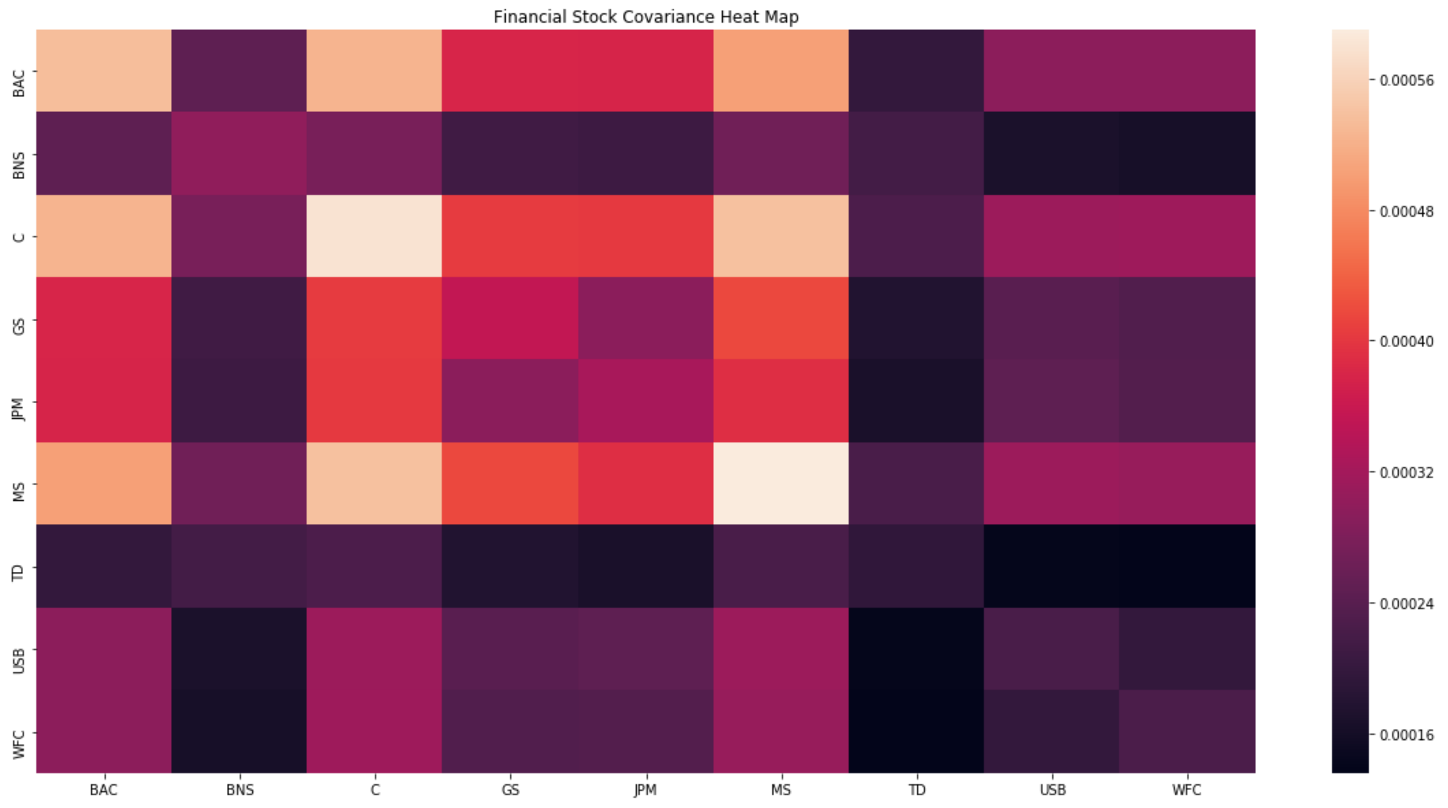

Figure 1 Financial Stock Covariance Matrix Heat Map

Variables Selection

As shown in the mathematical detail section, the required variables for Markowitz portfolio optimizer are the expected rate of return, risk-free rate and covariance matrix. Since we already have the data inputted and processed, we can estimate the expected rate of return using historical rate of return. This historical means approach, as pointed out in Black-Litterman paper, provides "very poor forecasts of future returns". Nonetheless, without the proper fundamental research on these companies, an estimate of historical means is reasonable for Markowitz optimizer. Thus, the table below shows the average of log return of each stock.

Expected Return	BAC	BNS	C	GS	JPM	MS	TD	USB	WFC
μ	-0.08%	0.18%	-0.10%	-0.07%	0.014%	-0.05%	0.11%	0.02%	-0.06%

Risk-free rate is assumed to be 1%. Covariance matrix is found in the previous section. The table below shows the data:

$\sum R$	BAC	BNS	C	GS	JPM	MS	TD	USB	WFC
BAC	0.00054	0.00025	0.00053	0.00038	0.00038	0.00050	0.00020	0.00030	0.00030
BNS	0.00025	0.00030	0.00027	0.00021	0.00021	0.00027	0.00022	0.00017	0.00016
C	0.00053	0.00027	0.00058	0.00040	0.00040	0.00054	0.00023	0.00031	0.00031
GS	0.00038	0.00021	0.00040	0.00035	0.00030	0.00042	0.00018	0.00024	0.00023
JPM	0.00038	0.00021	0.00040	0.00030	0.00032	0.00039	0.00017	0.00025	0.00023
MS	0.00050	0.00027	0.00054	0.00042	0.00039	0.00059	0.00022	0.00031	0.00031
TD	0.00020	0.00022	0.00023	0.00018	0.00017	0.00022	0.00019	0.00014	0.00014
USB	0.00030	0.00017	0.00031	0.00024	0.00025	0.00031	0.00014	0.00022	0.00020
WFC	0.00030	0.00016	0.00031	0.00023	0.00023	0.00031	0.00014	0.00020	0.00023

Portfolio Summary
All the Markowitz portfolio optimizers are written in a function file called PortfolioFunctions.py. In the python file, there are six functions including all the possible portfolios mentioned above.

Tangency Portfolio:

Tangency Weights	BAC	BNS	C	GS	JPM	MS	TD	USB	WFC
W_T	4%	-44%	-40%	38%	16%	-42%	79%	25%	64%

One can double check that the weights add up to 100%. One of the problems mentioned in Black-Litterman paper are the large short positions, which can be seen from BNS, C and MS weights. Though not very extreme, shorting half of the portfolio values is a risky move and significant cash reserves in preparation for potential losses. Furthermore, these large positions most of the time do not reflect investors' views of the market.

Minimum Variance Portfolio with Cash Position:

W_{min}	BAC	BNS	C	GS	JPM	MS	TD	USB	WFC	Cash
μ_p = 0.5%	2%	-22%	-20%	19%	16%	-42%	79%	25%	64%	49.8%
μ_p = 2%	-4%	45%	40%	-38%	-16%	43%	-79%	-26%	-65%	200.5%
μ_p = 5%	-16%	178%	159%	-152%	-63%	170%	-318%	-102%	-259%	501.9%
μ_p = 10%	-36%	401%	358%	-342%	-141%	383%	-715%	-231%	-582%	1004.3%

The extremely large shorting position is partially due to the low starting expected return from these companies. If investors are aiming for 10% as the portfolio expected return, then they have to short 582% of the total portfolio value of Wells Fargo stock and hold an approximate 1000% position on cash. Such distorted portfolio is impossible to construct.

Minimum Variance Portfolio without Cash Position:

No Cash Weights	BAC	BNS	C	GS	JPM	MS	TD	USB	WFC
W_{min}	4%	-29%	-52%	25%	28%	-34%	75%	33%	49%

Without the possibility of holding cash position, the minimum variance portfolio is considerably moderate but its variance is also higher.

Maximum Return Portfolio with Cash Position:

W_{min}	BAC	BNS	C	GS	JPM	MS	TD	USB	WFC	Cash
σ_p = 0.5%	-2%	22%	19%	-18%	-8%	21%	-38%	-12%	-31%	148.5%
σ_p = 2%	-8%	86%	77%	-73%	-30%	82%	-153%	-49%	-125%	294.1%
σ_p = 5%	-19%	215%	192%	-184%	-76%	206%	-384%	-124%	-313%	585.2%
σ_p = 10%	-39%	431%	384%	-367%	-151%	411%	-767%	-247%	-625%	1070.5%

Similar to the situation for minimum variance portfolio with cash position, large short position and cash position can be observed.

The portfolio return and standard deviation under various conditions can be calculated and is shown below:

Portfolios	Tangency Portfolio	Minimum Variance Portfolio w/ Cash				Minimum Variance Portfolio w/o Cash	Maximum Return Portfolio			
		0.5%	2%	5%	10%		0.5%	2%	5%	10%
μ_p	-0.985	0.5	2	5	10	-0.932	0.493	1.942	4.839	9.669
σ_p	1.030	0.518	1.035	4.141	9.318	1.003	0.5	2	5	10

Analysis
From the portfolio calculations, one can easily see the problem with this simple Markowitz optimizer. At a lot of the cases, it is simply impossible to construct the portfolio. Even if the portfolio is constructed, the risks and the cash position involved are too large. This is all due to the fact that this Markowitz model assumes that investors know all the expected return of each asset. To estimate an expected return for one asset is already a taunting task but to estimate all of the risky assets, it is simply unrealistic. Approximation methodologies such as historical mean yield less than satisfactory results. At the same time, this model might lead to shorting and longing asset that investors have contradictory views on. For example, if the investor believes that TD might outperform JPM in the upcoming quarter, then the maximum return portfolio with cash position is not the right portfolio to hold as it requires a significantly large position on TD and a relatively small short position on JPM. To avoid such contradiction and better incorporate investors' views, we will now move on to the Black-Litterman model.

Python Implementation
Part II Black-Litterman Portfolio Optimization

Data and Variable Selection
For Black-Litterman model, the same data set is used. However, instead of using the historical mean to find expected rate of return, the equilibrium market weight is estimated. I decide to estimate the market weight by using their market capitalization weight. **Market cap numbers are taken from May 18th, 2018**. Even though the data we are using is from 2016, we can make the assumption that the market capitalization ratio is approximately the same from 2016 to 2018. Using the market cap weight results in no short position.

Market Cap	BAC	BNS	C	GS	JPM	MS	TD	USB	WFC
M in B	306.82	75.38	178.39	89.83	378.37	95.45	107.75	83.03	261.67
W_{Mk}	19%	5%	11%	6%	24%	6%	7%	5%	17%

Once the weights are determined, the expected return at equilibrium can be found. The next step is to determine the scenarios we want to run. These scenarios will provide different P, Q and Ω matrices.

Scenario Selection
The following scenarios are arbitrarily chosen.

Scenario I: Morgan Stanley will have absolute excess return of 5%
Confidence of View: 25%

Scenario II: Goldman Sachs will outperform Citigroup by 2 basis point, 0.02%
Confidence of View: 50%

Scenario III: Bank of America and J.P. Morgan will outperform Wells Fargo and Bank of Nova Scotia by 1.5 basis point, 0.015%
Confidence of View: 75%

Based on Satchell and Scowcroft (2000) paper, matrix P reflects the scenarios above:

$$P_{equal-weighting} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & -0.5 & 0 & 0 & 0.5 & 0 & 0 & 0 & -0.5 \end{bmatrix}$$

The first row of P matrix corresponds to the absolute view. Since it only involves MS, the element representing MS is replaced by 1. This is the absolute view case. As for the relative view cases (scenario II and III), the row will have to sum to 0. Therefore, the second row of P matrix corresponds to the relative view of scenario II. For relative weighting scheme, it varies. Black-Litterman model though not specified in the paper most likely assigns the values using some benchmark criterion. As for Satchell and Scowcroft, they use an equal weighting scheme.

One can also use a market capitalization scheme to determine the multi-assets relative view case. For example, the last scenario III can be changed to the following due to the fact that Wells Fargo and J.P. Morgan have significant larger market capitalization:

$$P_{mkt-cap} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0.35 & -0.35 & 0 & 0 & 0.65 & 0 & 0 & 0 & -0.65 \end{bmatrix}$$

The numbers are again arbitrarily chosen. Technically, they can be calculated using the market capitalization data mentioned above.

Q matrix on the other hand doesn't have any ambiguity. It represents investors' views on the scenarios.

$$Q = \begin{bmatrix} 0.05 \\ 0.02 \\ 0.015 \end{bmatrix}$$

After getting both Q and P matrices, one can move on to Ω matrix, which is the most abstract and hard to calculate one. Ω matrix shows investors uncertainty in their views, or in other words, it shows how confident they are about the view. It is the variance of the error term shown earlier, ε. According to Morningstar report on Black-Litterman model written by Thomas Idzorek, Ω matrix can be computed using both P and τ by the following formula:

$$\Omega = \begin{bmatrix} p_1 \sum R \, {p_1}^t & 0 & 0 \\ 0 & p_2 \sum R \, {p_2}^t & 0 \\ 0 & 0 & p_3 \sum R \, {p_3}^t \end{bmatrix} = \begin{bmatrix} 0.000589872 & 0 & 0 \\ 0 & 0.000124397 & 0 \\ 0 & 0 & 9.26371e-05 \end{bmatrix}$$

In Black-Litterman paper, it suggests setting the constant τ to a number between 0.01 and 0.05, having been working with the model for a long time. There have been various papers that suggest different values for τ and δ. For more information, please read Satchell & Scowcroft (2000) and Blamont & Firoozye (2003).

Portfolio Summary

Black-Litterman model with all the variables above is implemented in another independent function file. BlackLittermanPortfolioOptimizer.py contains one function that implements the transformation written in the paper. The function returns four important matrices: expected return, modified expected return, modified covariance matrix and modified weight. The word modified here basically means investors' views are incorporated into the model, using P, Q and Ω matrix.

Expected return using the market capitalization weighted return:

Expected Return	BAC	BNS	C	GS	JPM	MS	TD	USB	WFC
$\mu_{mkt-cap}$	0.1%	0.056%	0.10%	0.077%	0.078%	0.1%	0.045%	0.062%	0.062%

Modified expected return:

Expected Return	BAC	BNS	C	GS	JPM	MS	TD	USB	WFC
μ_{BL}	0.29%	0.12%	0.29%	0.27%	0.23%	0.34%	0.10%	0.18%	0.16%

Modified covariance matrix:

$\sum R$	BAC	BNS	C	GS	JPM	MS	TD	USB	WFC
BAC	0.00056	0.00026	0.00055	0.00040	0.00039	0.00052	0.00021	0.00031	0.00031
BNS	0.00026	0.00031	0.00029	0.00022	0.00022	0.00028	0.00023	0.00018	0.00017
C	0.00055	0.00029	0.00061	0.00042	0.00042	0.00056	0.00024	0.00033	0.00033
GS	0.00040	0.00022	0.00042	0.00037	0.00031	0.00044	0.00018	0.00025	0.00024
JPM	0.00039	0.00022	0.00042	0.00031	0.00034	0.00041	0.00017	0.00026	0.00025
MS	0.00052	0.00028	0.00056	0.00044	0.00041	0.00062	0.00023	0.00033	0.00032
TD	0.00021	0.00023	0.00024	0.00018	0.00017	0.00023	0.00020	0.00015	0.00014
USB	0.00031	0.00018	0.00033	0.00025	0.00026	0.00033	0.00015	0.00023	0.00021
WFC	0.00031	0.00017	0.00033	0.00024	0.00025	0.00032	0.00014	0.00021	0.00024

Modified Weight:

Modified Weight	BAC	BNS	C	GS	JPM	MS	TD	USB	WFC
W_{BL}	118%	-96%	-297%	313%	209%	157%	7%	5%	-170%

A comparison graph can be constructed by comparing the original weight and the new weight.

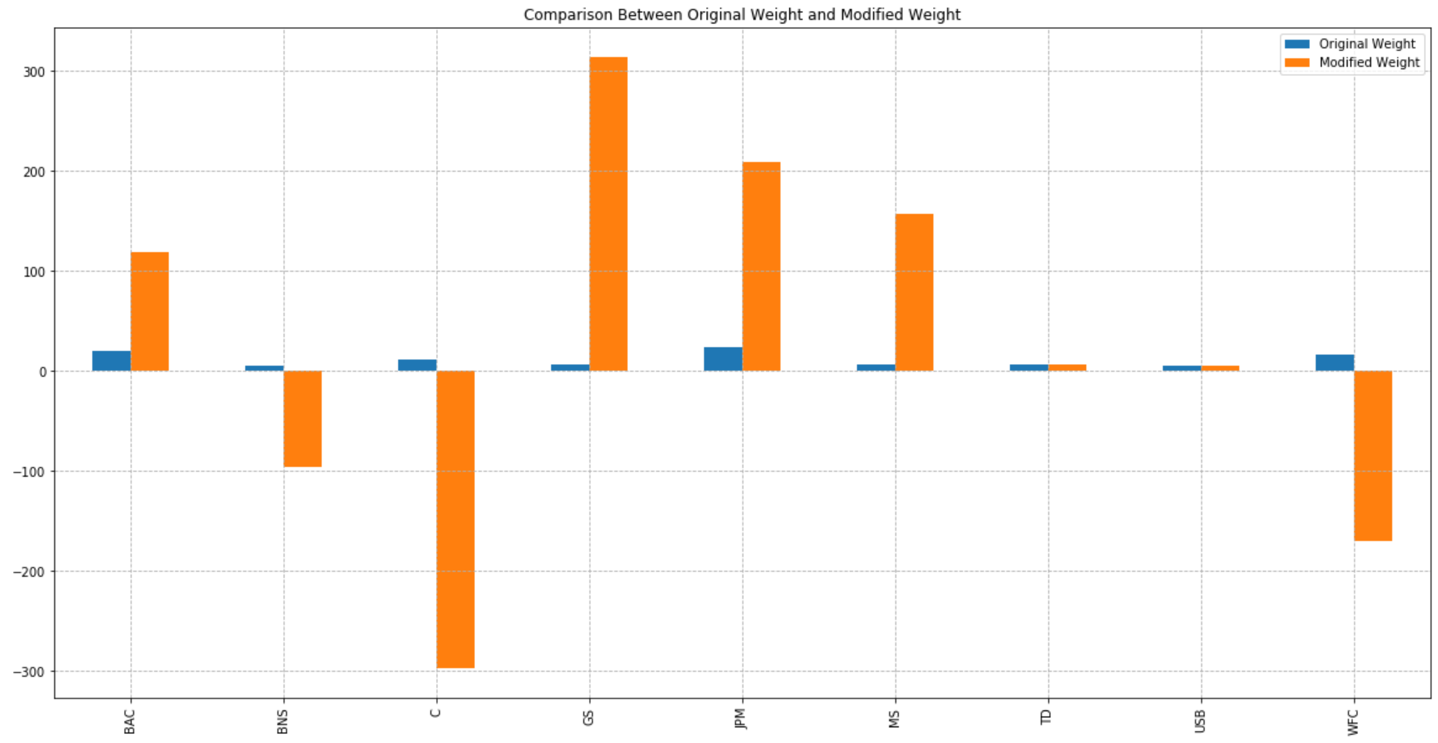

Figure 2 Comparison Between Original Weight and Modified Weight

Analysis:
1. The modified expected return is a lot higher than the original expected return. I believe this is due to the fact that I limited the original weights to strictly positive and reverse engineer out the expected return. Hence the original expected return is a bit low. Another possible reason is that having large shorting position on stocks that have relatively poor outlook tends to rein in more alpha.
2. Similar to the Markowitz model, there are large short positions. This is possibly due to the data set, having barely any log return during the period. However, if one looks at the weights closely, one can see that this is indeed what we desire. All the weights are closely matched with the arbitrary scenarios.
3. In response to scenario 1, one would expect to invest more in Morgan Stanley due to the view that the company will have positive excess return. In response to scenario 2, one should bank on Goldman Sachs performing better than Citigroup. A long-short strategy can be created and this is indeed the case shown in the graph with a large long position on GS and large short position on C. As for the last scenario, investing in JPM and BAC while shorting WFC and BNS is the desired strategy. The portion is also included with WFC having a larger short position than BNS. Having the views incorporated into the portfolio gives a much more desirable result despite the large short position.

Conclusion

As one can see from the analysis above, having views incorporated into the model allows the investors to express their views instead of blindly following the original Markowitz setting. Even though there are large short positions in both situations, investors can easily have justified the position using their scenarios.

Advantages and Disadvantages

Portfolio managers can easily incorporate their views into the Black-Litterman model. Since it is a relatively flexible model, investors can input any amount of views their desire. They don’t even have to input any view. The model can still provide a result.

However, this model is yet to be perfected. P and Ω matrix are all hard to estimate, especially Ω matrix. There are various ways suggested in different papers. There doesn’t seem to be a consensus as in how to incorporate a subjective feeling, such as confidence in views.

Improvement

As suggested in Morningstar paper, instead of calculating Ω matrix, one can replace the confidence in views with a confidence level and tilt system. This way investors can find the desired expected return and weights according to the confidence level they want to achieve. This method follows an intuitive approach and perhaps yields better result than arbitrarily selecting an Ω matrix.

Appendix

```
# PortfolioFunctions.py
# FRE6381 Final Project
# Black Litterman Model
# Written by: Po-Hsuan (Michael) Lin
# Date: May, 17, 2018
# Function files: containing functions that calculate minimum variance portfoliio,
# maximum return portfolio, tangency portfolio

## VARIABLES LIST ##
# Expected return: expected (equilibrium) return of assets; should be a matrix
# r: risk-free rate of assets
# weights: weights from the portfolios
# sigmaR: covariance matrix of the given assets' return in the portfolio; should be a matrix
# portfolio_return: the required return for a minimum variance portfolio; should be a number
# portfolio_sigma: the required std for a maximum return portfolio; should be a number

import numpy as np

def Portfolio_Return(expected_return, r, weights):
    mean_return = expected_return - r
    mean_return = np.transpose(mean_return)
    return (r + np.dot(mean_return,weights))

def Portfolio_Std(sigmaR, weights):
    return np.sqrt(np.transpose(weights) @ sigmaR @ weights)

def Tangency_Portfolio(expected_return, r, sigmaR):
    # In order for the formula to work, please ensure that expected_return and sigmaR are both matrices
    mean_return = expected_return - r
    mean_return = np.transpose(mean_return)
    H_matrix = np.linalg.solve(sigmaR, mean_return)
    one_matrix = np.ones(np.size(sigmaR,1))
    tangency_weight = (1 / (np.dot(np.transpose(one_matrix),H_matrix))) * H_matrix
    # tangency_weight should be a vector with the tangency weights for each assets
    return tangency_weight

def Minimum_Variance_Portfolio(expected_return, r, sigmaR, portfolio_return):
    # Call the tangency portfolio
    tangency_weight = Tangency_Portfolio(expected_return, r, sigmaR)
    # Calculate the cash portion of minimum variance portfolio
    mean_return = expected_return - r
    mean_return = np.transpose(mean_return)
    mv_cash = 1 - ((portfolio_return - r) / np.dot(mean_return,tangency_weight))
    mv_weight = (1 - mv_cash) * tangency_weight
    # mv_weight should be a vector while mv_cash is just a percentage
    return mv_cash, mv_weight

def Minimum_Variance_No_Cash_Portfolio(expected_return, r, sigmaR):
    # This portfolio has no cash investment
    one_matrix = np.ones(np.size(sigmaR,1))
    one_matrix = np.transpose(one_matrix)
    H_matrix = np.linalg.solve(sigmaR,one_matrix)
    mv_weight = (1 / (np.dot(np.transpose(one_matrix),H_matrix))) * H_matrix
    # mv_weight should be a vector
    return mv_weight

def Maximum_Return_Portfolio(expected_return, r, sigmaR, portfolio_sigma):
    # Call the tangency portfolio
    tangency_weight = Tangency_Portfolio(expected_return, r, sigmaR)
    # Calculate the cash portion of maximum return portfolio
    mean_return = expected_return - r
    mean_return = np.transpose(mean_return)
    H_matrix = np.linalg.solve(sigmaR, mean_return)
    one_matrix = np.ones(np.size(sigmaR,1))
    sign = np.sign(np.dot(np.transpose(one_matrix),H_matrix))
    G_matrix = np.transpose(tangency_weight) @ sigmaR @ tangency_weight
    mr_cash = 1 - ((portfolio_sigma * sign) / np.sqrt(G_matrix))
    mr_weight = (1 - mr_cash) * tangency_weight
    # mr_weight should be a vector while mr_cash is just a percentage
    return mr_cash, mr_weight
```

# BlackLittermanPortfolioOptimizer.py

```
    # FRE6381 Final Project
    # Black Litterman Model
    # Written by: Po-Hsuan (Michael) Lin
    # Date: May, 17, 2018
    # Function files: containing function that calculates Black-Litterman Models

    # Black Litterman Model assumes market is in equilibrium and analysts have opinions
    # on specific assets relative to the accepted equilibrium, rather than expected returns of all assets

    ## VARIABLES LIST ##
    # eq_weights: equilibrium weights
    # sigmaR: covariance matrix of the given assets' return in the portfolio; should be a matrix
    # delta: risk aversion parameter; see paper footnote
    # tau: a scalar that measures the uncertainty of CAPM; see paper footnote
    # P_matrix: probability of views happening; see paper
    # Q_matrix: analyst's view matrix
    # Omega_matrix: analyst's view uncertainty matrix; see paper

    import numpy as np

    def Black_Litterman_Model(eq_weights,sigmaR,delta,tau,P_matrix,Q_matrix,Omega_matrix):
        # First find the equilibrium expected return, assuming it is not given
        # According to the footnote, expected return = equilibrium risk premium and can be found using CAPM
        # Expected_return should be a vector
        expected_return = delta * np.dot(sigmaR, eq_weights)

        # From page nine of the paper (or page 35 from the journal)
        # We have the formula to calculate the new expected return with analyst's views incorporated
        # First calculate the view added term
        analyst_view_r_1 = np.dot(tau * np.dot(sigmaR,np.transpose(P_matrix)), np.linalg.inv(tau *
    np.dot(np.dot(P_matrix,sigmaR),np.transpose(P_matrix)) + Omega_matrix))
        analyst_view_r_2 = Q_matrix - np.dot(P_matrix,expected_return)
        view_term = np.dot(analyst_view_r_1,analyst_view_r_2)

        # Incorporate the added term into expected return
        modified_expected_return = expected_return + view_term

        # SigmaR can also be updated with the view
        analyst_view_s_1 = tau * sigmaR
        analyst_view_s_2 = np.dot(analyst_view_r_1, tau * np.dot(P_matrix, sigmaR))
        modified_sigmaR = sigmaR + analyst_view_s_1 - analyst_view_s_2

        # Find the optimal weight using modified_expected_return and modified_sigmaR
        modified_weight = np.dot(np.linalg.inv(delta * modified_sigmaR),modified_expected_return)
        return expected_return, modified_expected_return, modified_sigmaR, modified_weight
```

# Main.py

```
# FRE6381 Final Project
# Black Litterman Model
# Written by: Po-Hsuan (Michael) Lin
# Date: May, 17, 2018
# This is the main code that calls on different functions from PortfolioFunctions.py
# and BLackLittermanPortfolioOptimizer.py

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
import PortfolioFunctions as pf
import BlackLittermanPortfolioOptimizer as BL

# Instantiation
r = 0.01

# Extract data from data file of some financial stocks in 2016
data = pd.read_csv("Data.csv",header = 0, index_col = 0)
data.index = pd.to_datetime(data.index)

# Calculate the log return
data = np.log(data) - np.log(data.shift(1))
data = data.dropna()

# Calculate the covariance matrix of return of the stocks
sigmaR = data.cov()
fig = plt.figure(figsize = (20,10))
sns.heatmap(sigmaR,xticklabels = sigmaR.columns, yticklabels = sigmaR.columns)
ax = plt.gca()
ax.set_title("Financial Stock Covariance Heat Map")

# Store the data
```

```
    covariance_matrix = sigmaR.as_matrix()
    expected_return = (data.mean()).as_matrix()
    # sigmaR.to_csv("sigmaR.csv")

    ## Markowitz Portfolio Optimizer ##
    # Calculate tangency portfolio
    W_T = pf.Tangency_Portfolio(expected_return,r,covariance_matrix)
    Portfolio = pd.DataFrame(data = W_T * 100)
    Portfolio.columns = ['Tangency Portfolio']
    Portfolio.index = data.columns

    # Calculate minimum variance portfolio with cash position
    portfolio_expected_return = np.array([0.005,0.02,0.05,0.10])
    portfolio_expected_return = np.reshape(portfolio_expected_return, (4,1))
    W_min_cash, W_min =
pf.Minimum_Variance_Portfolio(expected_return,r,covariance_matrix,portfolio_expected_return)
    W_min_cash = W_min_cash * 100
    W_min = np.transpose(W_min * 100)
    Portfolio['MVP with mu = 0.5%'] = W_min[:,0]
    Portfolio['MVP with mu = 2%'] = W_min[:,1]
    Portfolio['MVP with mu = 5%'] = W_min[:,2]
    Portfolio['MVP with mu = 10%'] = W_min[:,3]

    # Calculate minimum variance portfolio without cash position
    W_min_no_cash = pf.Minimum_Variance_No_Cash_Portfolio(expected_return,r,covariance_matrix)
    W_min_no_cash = np.transpose(W_min_no_cash * 100)
    Portfolio['MVP with no cash'] = W_min_no_cash

    # Calculate maximum return portfolio with cash position
    portfolio_expected_vol = np.array([0.005,0.02,0.05,0.10])
    portfolio_expected_vol = np.reshape(portfolio_expected_vol, (4,1))
    W_max_cash, W_max =
pf.Maximum_Return_Portfolio(expected_return,r,covariance_matrix,portfolio_expected_vol)
    W_max_cash = W_max_cash * 100
    W_max = np.transpose(W_max * 100)
    Portfolio['MRP with sigma = 0.5%'] = W_max[:,0]
    Portfolio['MRP with sigma = 2%'] = W_max[:,1]
    Portfolio['MRP with sigma = 5%'] = W_max[:,2]
    Portfolio['MRP with sigma = 10%'] = W_max[:,3]

    # Calculate return and std
    tangency_return = pf.Portfolio_Return(expected_return,r,W_T*100)
    tangency_sigma = pf.Portfolio_Std(covariance_matrix,W_T*100)
    min_no_cash_return = pf.Portfolio_Return(expected_return,r,W_min_no_cash)
    min_no_cash_sigma = pf.Portfolio_Std(covariance_matrix,W_min_no_cash)
    min_return = np.zeros((4,1))
    min_sigma = np.zeros((4,1))
    max_return = np.zeros((4,1))
    max_sigma = np.zeros((4,1))
    for i in range(0,4):
        min_return[i] = pf.Portfolio_Return(expected_return,r,W_min[:,i])
        min_sigma[i] = pf.Portfolio_Std(covariance_matrix,W_min[:,i])
        max_return[i] = pf.Portfolio_Return(expected_return,r,W_max[:,i])
        max_sigma[i] = pf.Portfolio_Std(covariance_matrix,W_max[:,i])

    ## Black-Litterman Portfolio Optimizer ##
    # Instantiation
    delta = 2.5
    tau = 0.05
    # W_eq found using market capitalization
    W_eq =
np.array([[0.194597543,0.047809018,0.113142089,0.056973787,0.239977421,0.060538216,0.068339369,0.05
2660954,0.165961603]]).T
    P = np.array([[0,0,0,0,0,1,0,0,0],[0,0,-1,1,0,0,0,0,0],[0.35,-0.35,0,0,0.65,0,0,0,-0.65]])
    Q = np.array([[0.05],[0.02],[0.015]])
    omega = np.zeros((3,3))
    omega[0,0] = P[0,:] @ covariance_matrix @ np.transpose(P[0,:])
    omega[1,1] = P[1,:] @ covariance_matrix @ np.transpose(P[1,:])
    omega[2,2] = P[2,:] @ covariance_matrix @ np.transpose(P[2,:])

    # Call the formula
    expected_return, modified_expected_return, modified_sigmaR, modified_weight =
BL.Black_Litterman_Model(W_eq,covariance_matrix,delta,tau,P,Q,omega)
    modified_sigmaR = pd.DataFrame(data = modified_sigmaR)
    modified_weight = modified_weight * 100
    # modified_sigmaR.to_csv("sigmaR.csv")

    # Visualization
    W_eq = W_eq * 100
    Weight_comparison_data = pd.DataFrame(data = W_eq)
    Weight_comparison_data.columns = ['Original Weight']
    Weight_comparison_data.index = data.columns
    Weight_comparison_data['Modified Weight'] = modified_weight
    fig = plt.figure()
    Weight_comparison_data.plot(figsize = (20,10),kind = 'bar')
    ax = plt.gca()
    ax.set_title('Comparison Between Original Weight and Modified Weight')
    ax.grid(True,linestyle = '--')
```

References

Black, F. and Litterman, R. (1991). “Global Asset Allocation with Equities, Bonds, and Currencies.” Fixed Income Research, Goldman, Sachs & Company, October.

Black, F. and Litterman, R. (1992). “Global Portfolio Optimization.” Financial Analysts Journal, September/October, 28-43.

Blamont, D. and Firoozy, N. (2003). “Asset Allocation Model.” Global Markets Research: Fixed Income Research, Deutsche Bank, July.

He, G. and Litterman, R. (1999). “The Intuition Behind Black-Litterman Model Portfolios.” Investment Management Research, Goldman, Sachs & Company, December.

Idzerok, T. (2005). “A STEP-BY-STEP GUIDE TO THE BLACK-LITTERMAN MODEL.” Morningstar research, April.