FINANCE 441
- Investments -

Matthew Ringgenberg
Fall 2011

Using Matlab to Calculate Markowitz Optimized Portfolios
Overview: Optimization Steps

1. Import Data
2. Calculate Markowitz Inputs
3. Calculate the Efficient Frontier
4. Calculate the Optimal Portfolios
   A. Optimal Risky Portfolio
   B. Optimal Complete Portfolio
5. Incorporate Constraints

Note: To use the code shown in this presentation, you must have the Financial Toolbox add-on for Matlab
Matlab Basics and Caveats

- We’ll use the Matlab Financial Toolbox to calculate the Mean-Variance-Efficient portfolio
- Matlab is a matrix based language
  - All data is viewed as a matrix or vector
  - Matlab uses matrix algebra to make computations
- We’re going to do a simple version of the Markowitz calculation
  - We’ll assume that historical returns and covariances are the best estimate of future values
  - We won’t allow short sales (we’ll relax this later)
Starting Matlab

• Start Matlab and you’ll see the following screen:

  You’ll enter commands at the >> prompt

Make sure this points to the directory that contains your data
Calculating Historical Mean & Covariance

• Note: throughout this presentation, the “>>” prompt signifies the beginning of Matlab code
• Before we can proceed, we need to import our asset return data
  • The data should have one column for each asset, and each row should contain that asset’s return on a particular date
  • Remember, when calculating returns from price data, you must account for splits, buybacks, dividends, etc.
Calculating Historical Mean & Covariance

• To import our asset return data:
  • In Matlab, navigate to the location of your data, right click on the Excel file that contains your asset returns and select *Import Data*
  • Follow the prompts
  • Matlab will import the file and create a matrix called “data”

• If the dataset has missing values, you’ll need to drop them before proceeding by entering:

```
>> data(any(isnan(data),2),:) = [];  
```
Calculating Historical Mean & Covariance

• Next, we calculate the historical mean, covariance, and standard deviation of our asset returns

```matlab
>> m = mean(data)
>> c = cov(data)
>> stdev = std(data)
```

• Note: we’re going to make the strong assumption that historical values of the mean, covariance, and standard deviation are the best estimate of future values. Of course, you do not have to use the historical values.
Calculating the Efficient Frontier

• Then, we can use the built-in Financial Toolbox features to calculate efficient frontier portfolios:

```matlab
>> [PortRisk, PortReturn, PortWts] = portopt(m, c)
```

• The `portopt` function runs the Markowitz algorithm using the mean \(m\) and covariance \(c\) of our assets

• The function outputs three datasets: `PortRisk`, `PortReturn`, and `PortWts`
Efficient Frontier Results

- \textit{PortRisk} - an \text{NPORTS} x 1 vector of the standard deviation of returns for each portfolio

- \textit{PortReturn} - an \text{NPORTS} x 1 vector of the expected return of each portfolio

- \textit{PortWts} – \text{NPORTS} x \text{NASSETS} matrix of weights allocated to each asset
  - Each row represents a different frontier portfolio
Calculating the Efficient Frontier: Constraints

- By default, the *portopt* function imposes a non-negativity constraint on the asset weights
  - Weights must sum to 1
  - Non-negativity → short sales are not allowed
- If you want to add other restrictions or allow short sales you can specify your own constraints
- To do this, create a matrix of constraints and use it as an input in the optimization function
  - We’ll talk more about this later
  - Type *help portcons* for more information
Plotting the Efficient Frontier

Now, we can plot the efficient frontier for our assets:

```matlab
>> scatter(stdev, m)
hold on
plot(PortRisk, PortReturn, 'DisplayName', 'PortReturn vs. PortRisk', 'XDataSource', 'PortRisk', 'YDataSource', 'PortReturn'); figure(gcf)
xlabel('Risk (Standard Deviation)')
ylabel('Expected Return')
title('Mean-Variance-Efficient Frontier')
grid on
```
Plotting the Efficient Frontier

Mean-Variance-Efficient Frontier

Expected Return vs. Risk (Standard Deviation)

Efficient Frontier

Individual Assets
Calculating the Optimal Efficient Portfolio

- Finally, we need to calculate the **optimal** portfolio
- To do this, we’ll use the `portalloc` function:

```matlab
>> [RiskyRisk, RiskyReturn, RiskyWts, RiskyFraction, 
   OverallRisk, OverallReturn] = portalloc(PortRisk, 
   PortReturn, PortWts, .01, .01, 5)
```

  - where `.01,.01, 5` are the risk free rate, borrowing rate, and your risk aversion coefficient, respectively
  - Note: if you omit the output arguments to the `portopt` function it will automatically graph the results
The Optimal Efficient Portfolio

Optimal Complete Portfolio for $A = 5$
The Optimal Efficient Portfolio

Optimal Complete Portfolio for $A = 2$

Expected Return vs. Risk (Standard Deviation)

- Optimal Overall Portfolio
- Optimal Risky Portfolio
Examining the Results

- Notice that different risk aversion parameters lead to different allocations on the CAL
  - For low risk aversion, it might suggest margin trading

- Matlab will save the optimal risky portfolio weights in the matrix \textit{RiskyWts}
  - To output these to an Excel file, type:
    \texttt{>>xlswrite('OptimalWeights.xlsx',RiskyWts)}
Changing the Default Constraints

• What if we want to allow short sale constraints and limit our weights to be no bigger than 30% on any asset?
  • We’ll create a matrix that contains upper and lower boundaries for our weights, called AssetBounds
    • The first row contains the lower bound, the second row contains the upper bound
  • Then, we’ll repeat the allocation exercise using the frontcon function in place of the portopt function
    • The frontcon function calculates the efficient frontier with constraints
Create a Matrix that Specifies your Constraints

- The first few columns of the *AssetBounds* matrix are shown below as an example:

<table>
<thead>
<tr>
<th>-0.30</th>
<th>-0.30</th>
<th>-0.30</th>
<th>-0.30</th>
<th>-0.30</th>
<th>-0.30</th>
<th>-0.30</th>
<th>-0.30</th>
<th>-0.30</th>
<th>-0.30</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
</tr>
</tbody>
</table>

- There should be one column for every asset
- Of course, you can choose other boundary values
Incorporating your Revised Constraints

- To incorporate the constraints, we first calculate the efficient frontier using `frontcon`:

  ```
  >> [PortRisk, PortReturn, PortWts] = frontcon(m, c, [], [], AssetBounds)
  ```

- Then, we calculate the optimal portfolios using the same command we used last time:

  ```
  >> [RiskyRisk, RiskyReturn, RiskyWts, RiskyFraction, OverallRisk, OverallReturn] = portalloc(PortRisk, PortReturn, PortWts, .01, .01, 5)
  ```
Allowing Short Sales Leads to “Better” Portfolios

No Short Sales

Short Sales Allowed
- Better Sharpe Ratio
- Efficient Frontier is Higher
Concluding Remarks

• The Financial Toolbox is quite flexible and is capable of more complicated optimizations
  • Weight constraints, short selling, margin trading, transaction costs, etc.
• The toolbox also includes functions that are designed to help back-test and evaluate strategies
• See http://www.mathworks.com/help/toolbox/finance/ for more information