Markowitz Portfolio Example

Mac Radigan

0.1 Introduction

Markowitz portfolio theory is used in investment management as a tool for diversifying away risk through portfolio balancing. Given a set of investment assets, it finds the most efficient allocation (portfolio weights). Here, efficiency is defined as minimizing the expected variance. Markowitz portfolios may be subject to specified constraints, such as a specific return on investment (expected price), non-negativity constraints (restricting short selling), and may include risk-free assets or market indices [1].

This example will select a minimum variance portfolio, constrained to a specified expected rate of return. All results will be annualized as commonly reported to investors.

0.2 Implementation

This optmization problem may be expressed as minimizing the portfolio variance, subject to the constraint that the individual allocation weights of the stocks add to unity, and the expected return is equal to the specified target.

$$
\begin{aligned}
\underset{w}{\text{minimize}} \quad & \sigma_{p,w}^2 = \mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w} \\
\text{subject to} \quad & \mu_{opt} = \mathbf{w}^T \mu \\
& \mathbf{w}^T \underline{\mathbf{1}} = 1
\end{aligned}
$$

Where

$$
\begin{array}{ll}
\sigma_{p,w}^2 & \text{portfolio variance of weighted assets} \\
\mathbf{w} & \text{individual asset weights} \\
\boldsymbol{\Sigma} & \text{covariance matrixw} \\
\mu & \text{individual asset returns} \\
\mu_{opt} & \text{target portfolio return}
\end{array}
\tag{1a}
$$

Forming the Lagrangian function for the constrained minimization, we have

$$
L(w, \lambda_1, \lambda_2) = \mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w} + \lambda_1(\mathbf{w}^T \mu - \mu_{opt})
\tag{2}
$$

So the first order conditions are

$$
2\partial L(w, \lambda_1, \lambda_2) = 2\boldsymbol{\Sigma}\mathbf{w} + \lambda_1 \mu + \lambda_2 \underline{\mathbf{1}} = \underline{\mathbf{0}}
\tag{3}
$$

$$
\frac{2\partial L(w, \lambda_1, \lambda_2)}{\partial \lambda_1} = \mathbf{w}^T \mu - \mu_{opt} = 0
\tag{4}
$$

$$
\frac{2\partial L(w, \lambda_1, \lambda_2)}{\partial \lambda_2} = \mathbf{w}^T \underline{\mathbf{1}} - 1 = 0
\tag{5}
$$

Expressed in matrix form

$$
\mathbf{Ax} =
\begin{bmatrix}
2\boldsymbol{\Sigma} & \mu & \underline{\mathbf{1}} \\
\mu^T & 0 & 0 \\
\underline{\mathbf{1}}^T & 0 & 0
\end{bmatrix}
=
\begin{bmatrix}
\mathbf{w} \\
\lambda_1 \\
\lambda_2
\end{bmatrix}
\begin{bmatrix}
\underline{\mathbf{0}}^T \\
\mu_{opt} \\
1
\end{bmatrix}
= \mathbf{b}
\tag{6}
$$

This may be solved for $\mathbf{b}$, where the first $n - 2$ elements of $\mathbf{b}$ are the portfolio weights ($\mathbf{w}_n = \mathbf{b}_n$).

$$
\mathbf{b} = \mathbf{A}^{-1}\mathbf{x}
\tag{7}
$$

References

[1] Wikipedia, "Modern portfolio theory — Wikipedia, the free encyclopedia," 2011, [Online; accessed 13-December-2013]. [Online]. Available: http://en.wikipedia.org/Modern_portfolio_theory

1 Appendix A: The Example Portfolio

1.1 Portfolio #1: AAPL, JPM, LMT, XOM

Portfolio #1 (AAPL, JPM, LMT, XOM)			
Symbol	**Weight**	**ROI**	**Volatilty**
AAPL	0.08	11.4%	28.8%
JPM	0.12	31.6%	19.1%
LMT	0.39	55.8%	16.3%
XOM	0.40	08.2%	13.0%
portfolio		25.0%	11.76%

Table 1: Portfolio #1

Figure 1: Portfolio Performance

2 Appendix B: Historical Data

2.1 AAPL Stock Chart

Figure 2: AAPL

2.2 JPM Stock Chart

Figure 3: JPM

2.3 LMT Stock Chart

Figure 4: LMT

2.4 XOM Stock Chart

Figure 5: XOM

3 Appendix C: Octave Source Code

3.1 Appendix C.1: script example (main entry point)

```octave
1   #!/usr/bin/env octave
2   %#!/usr/bin/env octave -qf
3   % MARKOWITZDEMO  runs the Markowitz demo
4   %
5   %  Description:  This is a command line entry point for
6   %                demonstrating portfolio balancing using
7   %                a basic applicaton Markowitz portfolio theory.
8   %
9   %                In the current implementation, a target return
10  %                on investments is specified, short selling is
11  %                permitted, and there is no inclusion of a risk-free
12  %                asset.
13  %
14  %  See Also:     http://wikipedia.org/wiki/Modern_portfolio_theory
15  %                for more information on the Markowitz portfolio
16  %
17  %                http://finance.yahoo.com
18  %                for more information on Yahoo! Finance data sources
19  %
20  %  Author:
21  %    Mac Radigan
22
23  return_opt    = 25.0;      % desired annualized rate of return (percent)
24  window_length = 250;       % window size (in days) of stock price samples
25                             % 250 = 1 year of trading days
26  symbols = sort({    ...    % stocks in portfolio
27            'AAPL', ...
28            'JNJ',  ...
29            'LMT',  ...
30            'XOM',  ...
31          })';
32  SAVE_FIGURES = true;       % enable to save figures {T:enable, F:disable}
33
34  [weights, volatility, volatility_opt, portfolio_rreturn, rate_of_return] ...
35    = markowitzPortfolio(symbols, window_length, return_opt);
```

```matlab
36
37   % print the efficient portfolio results as a table
38   fprintf(stdout, 'efficient portfolio: return=%2.2f%%, ...
         volatility=%2.2f%%\n',portfolio_rreturn,volatility_opt);
39   % load all tabular data into a cell array
40   D=vertcat( ...
41     symbols', num2cell(weights)', ...
42     num2cell(rate_of_return), num2cell(volatility));
43   fprintf(stdout, '\tSYM\tWEIGHT\tRETURN\tVOLATILITY\n',D{:});   % print tabular data
44   fprintf(stdout, '\t%s\t%2.2f\t%2.2f%%\t%2.2f%%\n',D{:});   % print tabular data
45
46   % plot results
47   if(SAVE_FIGURES)
48     FS=15;
49     FW='bold';
50     mxy = [max([volatility volatility_opt]), max([rate_of_return return_opt])];
51     dxy = 1.5*[1 1];
52     %hfig = figure('visible','off');
53     hfig = figure('visible','on');
54     scatter(volatility,rate_of_return,40,'red','^');
55     hold on;
56     text(volatility+dxy(1),rate_of_return+dxy(2),symbols);
57     scatter(volatility_opt,return_opt,40,'green','^');
58     text(volatility_opt+dxy(1),return_opt+dxy(2),'efficient');
59     title('Portfolio Performance','FontSize',FS,'fontweight',FW);
60     ylabel('Annualized Return (\mu)','FontSize',FS,'fontweight',FW);
61     xlabel('Annualized Risk (\sigma)','FontSize',FS,'fontweight',FW);
62     legend('Efficient Portfolio');
63     axis([0 mxy(1)],[0 mxy(2)]);
64     outdir='results';
65     fmts={'eps','jpg','png','tiff'};
66     name=sprintf('portfolio_performance_%s',strjoin('_',symbols));
67     mkdir(outdir);
68     for fmt=fmts
69       filename=sprintf('%s/%s.%s',outdir,name,fmt{:});
70       fprintf(stdout,'\tsave> %s\n',filename);
71       saveas(hfig,filename);
72     end
73   end
74
75   %% *EOF*
```

3.2   Appendix C.2: Markowitz Portfolio optimization function

```
1   % MARKOWITZPORTFOLIO minimum—variance Markowitz portfolio solver
2   %
3   %   [weights, volatility, volatility_opt, portfolio_rreturn, rate_of_return] ...
4   %      = markowitzPortfolio(symbols, window_length, return_opt);
5   %
6   %      where
7   %              symbol          is a cell array of ticker symbols,
8   %                              i.e. {symbol1, symbol2, ...}
9   %
10  %              window_length   is the number of samples used in
11  %                              risk/return estimation
12  %
13  %              return_opt      is the desired annual portfolio percent return
14  %
15  %              weights         is a Nx1 real matrix of the Markowitz portfolio
16  %                              allocation weights, one for each ticker symbol
17  %
18  %              weights         is a Nx1 real matrix of the Markowitz portfolio
19  %                              allocation weights, one for each ticker symbol
20  %
21  %              volatilities    is a Nx1 real matrix of the volatility of
22  %                              asset in the portfolio, one for each ticker symbol
23  %
24  %              volatility_opt  is the volatility of the Markowitz—weighted
25  %                              portfolio
26  %
27  %              rate_of_return  is a Nx1 real matrix of annualized rates of return
28  %                              of each asset in the portfolio, one for each
29  %                              ticker symbol
30  %
31  %   Description:  This class identifies the minimum—variance
32  %                 Markowitz portfolio, for a specified set of
33  %                 portfolio stocks and target portfolio return
34  %                 on investment.
35  %
36  %   Examples:
37  %
```

```matlab
38  %     return_opt    = 25.0;     % desired annualized rate of return (percent)
39  %     window_length = 250;      % window size (in days) of stock price samples
40  %                               % 250 = 1 year of trading days
41  %     symbols = sort({    ...   % stocks in portfolio
42  %                 'AAPL', ...
43  %                 'JNJ',  ...
44  %                 'LMT',  ...
45  %                 'XOM',  ...
46  %                 })';
47  %
48  %     [weights, volatility, volatility_opt, rate_of_return] ...
49  %         = markowitzPortfolio(symbols, window_length, return_opt);
50  %
51  %
52  %  See Also:      http://wikipedia.org/wiki/Modern_portfolio_theory
53  %                 for more information on the Markowitz portfolio
54  %
55  %  Author:
56  %     Mac Radigan
57
58  function [weights, volatility, volatility_opt, portfolio_rreturn, rate_of_return, S, R] = ...
         markowitzPortfolio(symbols, window_length, return_opt=NaN)
59    volatility = [];        % volatility
60    rate_of_return = [];    % rates of return
61    weights = [];           % portfolio weights
62    volatility_opt = NaN;   % portfolio volatility
63    %
64    T = 250;   % number of trading days in a year (time horizon)
65             % (either 250 or 252, depending specific reporting requirements)
66    P = 1/T;   % time period
67    Np = numel(symbols);        % number of stocks in portfolio
68    Ns = window_length-1;       % number of samples
69    % X is an M x N matrix of daily returns,
70    %    where M is the number daily returns
71    %    and  N is the number number of stocks in the portfolio
72    X = zeros(Ns,Np);
73    for idx=1:numel(symbols)
74      % retreive stock data
75      ds = getStockData(symbols{idx});
76      ds2.(symbols{idx}) = ds;
77      x_close  = ds.Close(1:window_length);       % limit samples to window_length
```

```matlab
78       X(:,idx)  = -1*diff(x_close)./x_close(2:end); % rates of return (daily)
79    end
80    R = corrcoef(X);      % correlation matrix of daily returns
81    S = cov(X,1);         % covariance matrix of daily returns
82    mu = mean(X);         % average daily rate of return
83    sig = sqrt(diag(S))'; % variance of daily return
84    %
85    % If a target rate of return is specified, perform a perfolio optimization with
86    % this constraint.  Otherwise, use the global minimum variance solution.
87    %
88    if(isnan(return_opt)) % global minimum-variance Markowitz portfolio
89       [w, mu_opt, sig_opt] = MPglobalMinimumVariance(mu, Np, S);
90    else                  % Markowitz portfolio with specified rate of return
91       [w, mu_opt, sig_opt] = MPconstrainedReturn(mu, S, Np, T, return_opt);
92    end
93    % return values: portfolio weights, portfolio volatility, and
94    %                 individual rates of return and volatilities
95    % retport all return/volatility as annualized percentages
96    weights = w;                          % rename rates
97    volatility = sig/sqrt(1/T)*100;          % annualized volatility (%)
98    volatility_opt = sig_opt/sqrt(1/T)*100;   % annualized portfolio variance (%)
99    rate_of_return = ((1+mu).^T-1)*100;       % annualized portfolio
100                                             % rate of return (%)
101   portfolio_rreturn = ((1+mu_opt).^T-1)*100; % annualized target return (%)
102 end
103
104 function [w, mu_opt, sig_opt] = MPglobalMinimumVariance(mu, Np, S)
105   % Markowitz global minimum-variance portfolio optimization
106   %
107   %     minimize:    sigma_p,w^2 = w'*S*w
108   %
109   %     subject to:       w'*1v = 1
110   %
111   %     where
112   %        sigma_p,w^2    portfolio variances
113   %        w              portfolio weights
114   %        S              covariance matrix
115   %        1v             one vector
116   %
117   %
118   %        A       *       x       =       b
```

```matlab
119    %      | 2S 1v | * | w       | = | 0v  |
120    %      | 1' 0  |   | lambda2 |   | 1   |
121    %
122    %         where
123    %           S       daily rate of return covariance matrix
124    %           S       daily rate of return covariance matrix
125    %           mu      daily mean ROI
126    %           w       portfolio weights
127    %           1v      one vector (size of portfolio)
128    %           0v      zero vector (size of portfolio)
129    %           lambda2 Lagrange multiplier for normalized weight constraint
130    %
131    % Markowitz linear equations
132    A = [ 2*S ones(1,Np)' ; ones(1,Np) 0 ];
133    b = [ zeros(1,Np) 1 ]';
134    z = A\b;                  % solve Ax=b
135    w = z(1:Np);              % assign portfolio weights
136    mu_opt   = mu*w;          % mean portfolio return
137    sig2_opt = w'*S*w;        % portfolio variance
138    sig_opt  = sqrt(sig2_opt); % portfolio standard deviation
139 end

140

141 function [w, mu_opt, sig_opt] = MPconstrainedReturn(mu, S, Np, T, return_opt)
142    % Markowitz portfolio optimization with constraint on ROI
143    %
144    %      minimize:  sigma_p,w^2 = w'*S*w
145    %
146    %      subject to:    mu_opt = w'*mu
147    %
148    %                     w'*1v  = 1
149    %
150    %      where
151    %         sigma_p,w^2      portfolio variances
152    %         mu_opt  target   portfolio rate of return
153    %         w                portfolio weights
154    %         S                covariance matrix
155    %         1v               one vector
156    %
157    %
158    %          A       *    x      =     b
159    %      | 2S  mu 1v | * | w       | = | 0v     |
```

```
160    %      | mu' 0  0 |   | lambda1 |   | mu_opt |
161    %      | 1'  0  0 |   | lambda2 |   | 1      |
162    %
163    %        where
164    %         S      daily rate of return covariance matrix
165    %         mu     daily mean ROIs
166    %         mu_opt target portfolio rate of return
167    %         w      portfolio weights
168    %         1v     one vector (size of portfolio)
169    %         0v     zero vector (size of portfolio)
170    %         lambda1 Lagrange multiplier for target return
171    %         lambda2 Lagrange multiplier for normalized weight constraint
172    %
173    mu_opt = (1+return_opt/100)^(1/T)-1; % convert target return to fractional daily
174    % Markowitz linear equations
175    A = [ 2*S mu' ones(1,Np)' ; mu 0 0 ; ones(1,Np) 0 0 ];
176    b = [ zeros(1,Np) mu_opt 1 ]';
177    z = A\b;                        % solve Ax=b
178    w = z(1:Np);                    % assign portfolio weights
179    mu_opt_assert  = mu*w;          % mean portfolio return (sanity check)
180    sig2_opt       = w'*S*w;        % portfolio variance
181    sig_opt        = sqrt(sig2_opt); % portfolio standard deviation
182    % check result
183    tol = 10*eps;                        % numerical tolerance
184    assert(abs(mu_opt_assert-mu_opt)<tol);  % assert
185    A
186    b
187    z
188  end
189
190  %% *EOF*
```

3.3 Appendix C.3: Stock Data Reader

```
1  function [data] = getStockData(symbol)
2  % GETSTOCKDATA read Yahoo! Finance time series CSV files
3  %
4  %  getStockData(symbol)
5  %
```

```matlab
 6  %       where
 7  %               symbol is a ticker symbol (character array)
 8  %
 9  %               data   is a time—series structure with the
10  %                   following fields:
11  %                       .symbol   — ticker symbol          (character array)
12  %                       .Date     — epoch time in seconds  (Nx1 int32 matrix)
13  %                       .DateStr  — localized date string   (character array)
14  %                       .Open     — opening price          (Nx1 real matrix)
15  %                       .High     — high price             (Nx1 real matrix)
16  %                       .Low      — low price              (Nx1 real matrix)
17  %                       .Close    — closing price          (Nx1 real matrix)
18  %                       .Volume   — volume                 (Nx1 real matrix)
19  %                       .Adj_Close — adjusted closing price (Nx1 real matrix)
20  %
21  %  Examples:
22  %     data = getStockData('LMT')
23  %
24  %  Author:
25  %     Mac Radigan
26  %
27    data = {};                                        % stock data
28    data.symbol = symbol;                             % stock ticker symbol
29    filename = sprintf('../data/%s/%s.csv',symbol,symbol); % path to stock data
30    if(¬exist(filename,'file'))
31      error('file %s not found',filename);            % assert
32    end
33    fid = fopen(filename,'r');                         % open for reading
34    fields = regexprep(strsplit(fgetl(fid),','),' ','_'); % read field names
35    fmt = '%10c,%f,%f,%f,%f,%d,%f\n';                   % CSV field pattern
36    n = size(strsplit(fmt,','),2);                     % number of fields
37    D = cell(n,1);                                     % temporary storage
38    for fIdx=1:n, data.(fields{fIdx}) = []; end        % initialize fields
39    while(!feof(fid))
40     [D{:},¬] = fscanf(fid,fmt,'C');                    % read each line
41     data.(fields{1}){end+1} = ...
42       mktime(strptime(D{1},'%Y—%m—%d'));               % record datetime
43     for fIdx=2:n, data.(fields{fIdx})(end+1) = D{fIdx}; end % assign fields
44    endwhile
45    % all matrix operations are defined in terms of column vectors,
46    % so transpose
```

```matlab
47    for fIdx=2:n, data.(fields{fIdx}) = data.(fields{fIdx})'; end
48    % convert datetime stamp to ASCII string, using system localization
49    for rIdx=2:numel(data.Date)
50      data.DateStr{rIdx} = asctime(localtime(data.Date{rIdx})); % localized date
51    end
52    fclose(fid);
53  end
54
55  %% *EOF*
```