

---

## Table of Contents

HW3 - Portfolio Optimization .....	1
Q1 .....	1
Q2 .....	3
Q3 .....	3

## HW3 - Portfolio Optimization

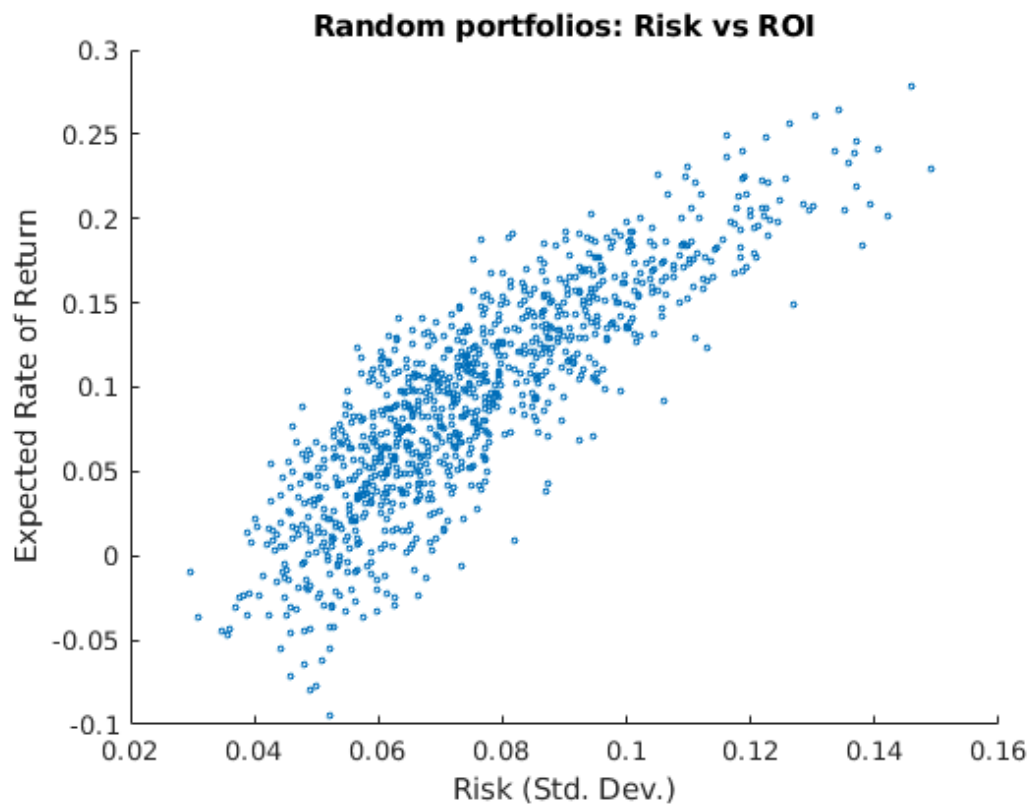
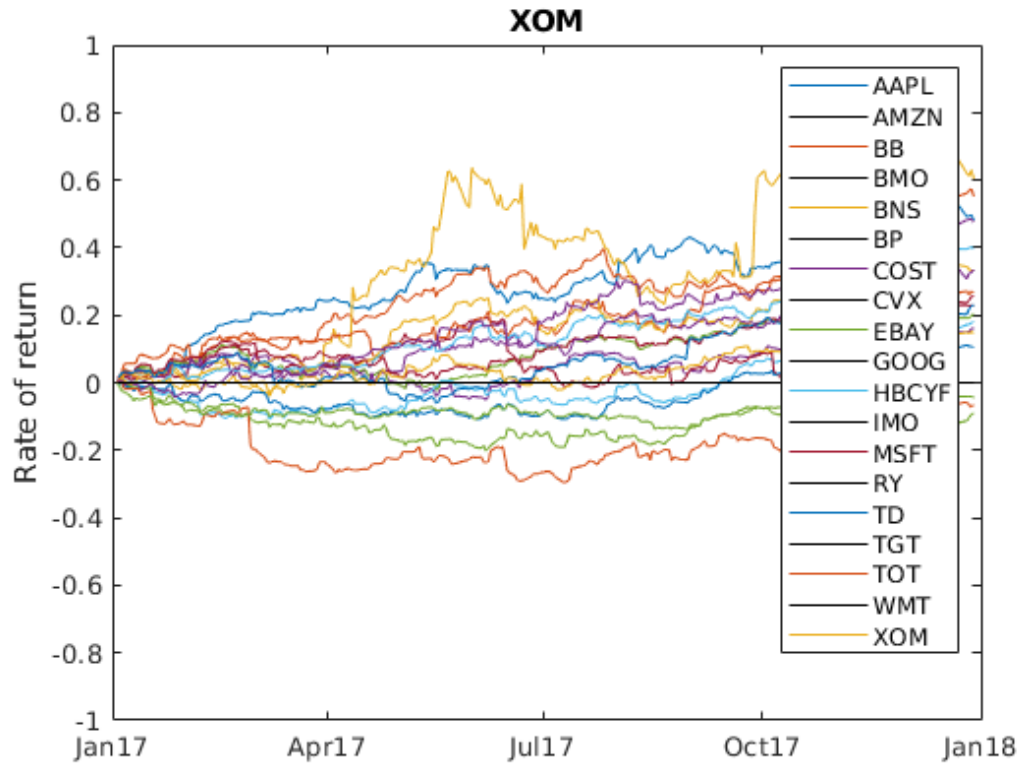
Guowei Huang - 25911158 Arash Outadi - 35898139

### Supplementary files:

```
sup_code = [  
    "load_stocks"  
    "load_stock"  
    "portfolio_scatter"  
    "meancov"  
    "return_range"  
    "efficient_frontier"  
    "market_portfolio"  
];  
for i = 1:length(sup_code)  
    publish(sup_code(i), 'format', 'pdf', 'evalCode', false);  
end  
  
clear all  
close all  
  
warning('off', 'MATLAB:table:ModifiedAndSavedVarnames');  
% rng(1); % Random seed for testing\debugging
```

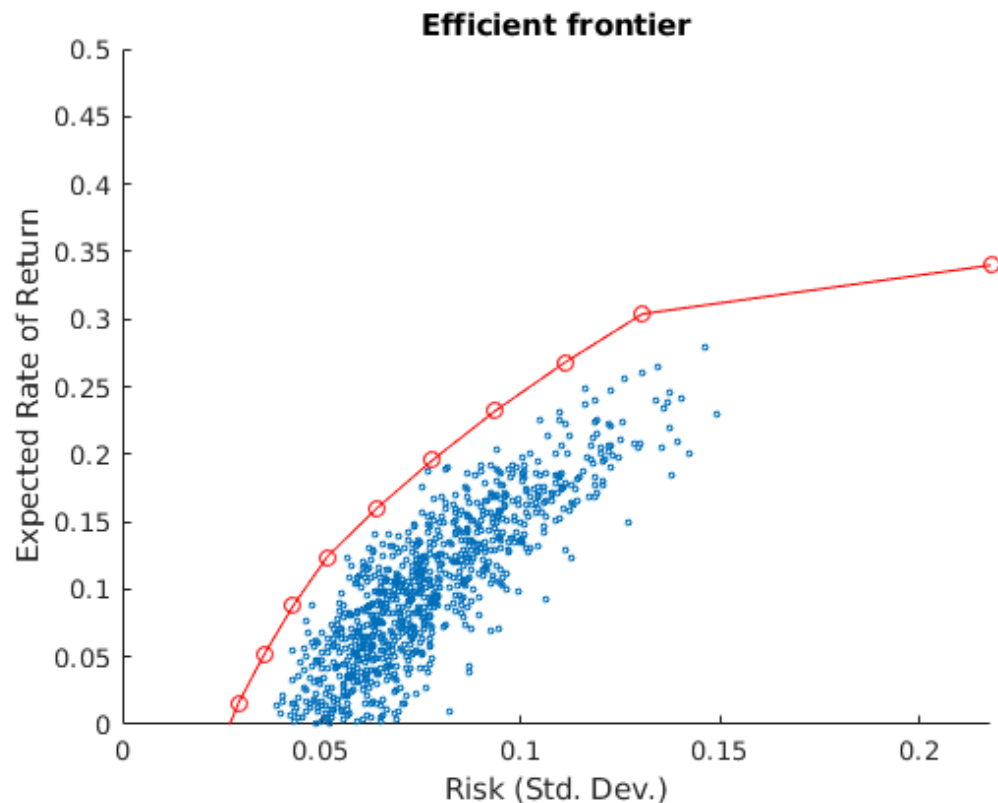
## Q1

```
[X, dates, names] = load_stocks("data", "2017-01-03", "2017-12-29");  
disp_stocks(X, dates, names);  
[r, Sig] = meancov(X);  
h = portfolio_scatter(r, Sig, 1000);
```



## Q2

```
num = 12;  
[Y,rates,sigs]= efficient_frontier(r,Sig,num);  
figure(h);  
hold on;  
title("Efficient frontier");  
plot(sigs, rates, 'ro-');  
ylim([0 0.5]);  
xlim([0 max(sigs)]);
```



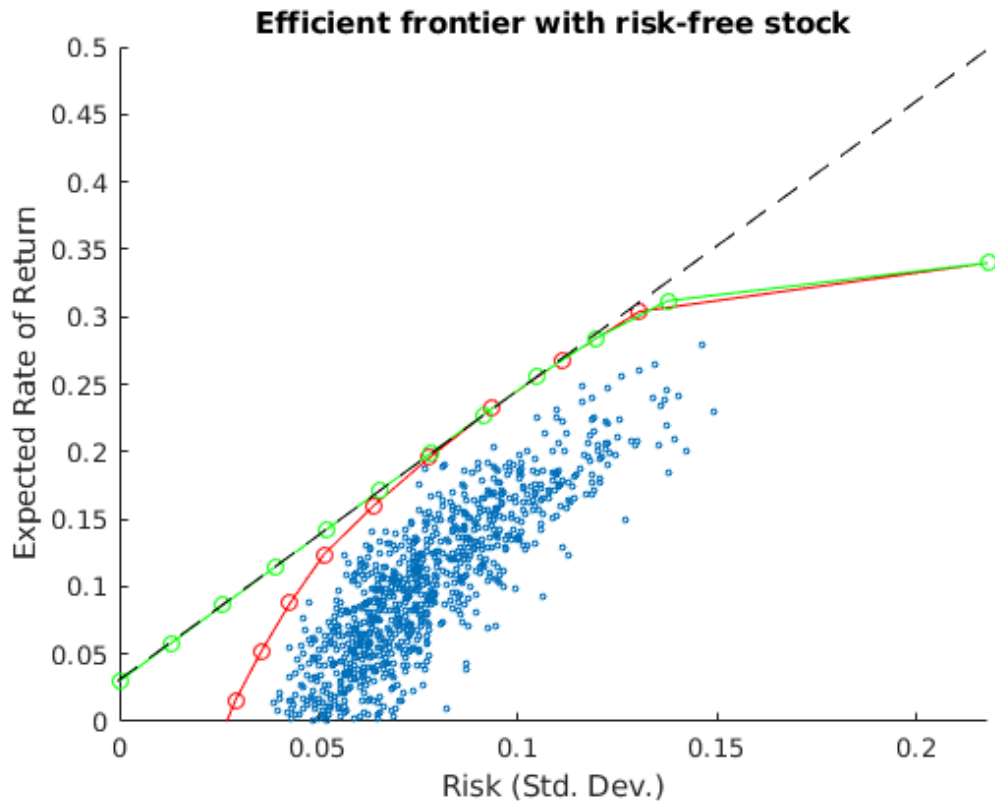
## Q3

```
f = 0.03;  
r_ = [r, f]; % add risk free return  
Sig_ = [Sig, zeros(19,1)];  
Sig_ = [Sig_; zeros(1,20)]; % build risk free Sig  
[Y_,rates_,sigs_]=efficient_frontier(r_,Sig_,num); %calculate  
corresponding efficient frontier  
plot(sigs_,rates_,'go-');  
title("Efficient frontier with risk-free stock")  
market_x = market_portfolio(f,r',Sig);  
annotation(h,'line',[0.128070175438597 0.910526315789474],...  
[0.15852380952381 0.928571428571429],'LineStyle','--');
```

Binary search:

Iter	x	$ f(x) $
1	1.20e-01	3.28e-02
2	7.09e-02	3.14e-02
3	9.54e-02	5.54e-04
4	1.08e-01	1.99e-02
5	1.01e-01	1.10e-02
6	9.84e-02	5.67e-03
7	9.69e-02	2.68e-03
8	9.61e-02	1.09e-03
9	9.57e-02	2.78e-04
10	9.56e-02	1.36e-04
11	9.57e-02	7.13e-05
12	9.56e-02	3.23e-05
13	9.56e-02	1.96e-05
14	9.56e-02	6.35e-06
15	9.56e-02	6.61e-06
16	9.56e-02	1.29e-07

Done .



### Q3 - Meaning of linear part

The linear half-line (Before the tangent point) represents the new extended region of the efficient frontier thanks to the addition of the risk free stock. This stock has no variance by definition thus including in the portfolio results in a linear gain in rate of return

Published with MATLAB® R2018a

---

```
function [r, dates] = load_stock(filename, startdate, enddate)

A = readtable(filename);

dates = table2array(A(:,1));
p = table2array(A(:,6));
% Filter out the dates not between start and end date
index_array = find(dates >= startdate & dates <= enddate);
dates = dates(index_array);
p = p(index_array);

% Compute rate of return
r = (p - p(1))/p(1);

end
```

*Published with MATLAB® R2018a*

---

```
function [X, dates, names] = load_stocks(dirname, startdate, enddate)
% Convert strings to datetime objects for comparisons in load_stock.m
startdate = datetime(startdate, 'InputFormat', 'yyyy-MM-dd');
enddate = datetime(enddate, 'InputFormat', 'yyyy-MM-dd');

files = dir(fullfile(pwd, dirname));
datafiles = [];
for i = 1:length(files)
    file = files(i);
    if endsWith(file.name, ".csv")
        datafiles = [datafiles; file];
    end
end

n = length(datafiles); % n stocks
X = []; % Annoying that we do not know the number of days in
advance...

names = cell(n, 1);
for i = 1:n
    file = datafiles(i);
    names(i) = {file.name(1:end-4)}; % Need curly brackets to convert
to cell... Stupid..
    [roi, dates] = load_stock(fullfile(file.folder, file.name),
startdate, enddate);
    X = [X, roi]; % Concanate new vector to matrix
end

end
```

*Published with MATLAB® R2018a*

---

```
function h = portfolio_scatter(r, Sig, num)
h = figure;
randmu = zeros(num,1);
randSig = zeros(num,1);

% For each portfolio generated a random allocation
numStocks = 5;
n = length(r);

for ii=1:num
    randomStocks = randperm(19,5); %randperm is used to avoid
    overwriting same stock
    rn = rand(numStocks, 1);
    randomAllocation = rn/sum(rn);
    weight = zeros(size(r));
    weight(randomStocks') = randomAllocation; % the selected
    stocks have some allocation, unselected stocks will be 0
    expectedReturn = r* weight';
    expectedRisk = sqrt(weight * Sig * weight'); % From modern
    portfolio theory page
    randSig(ii) = expectedRisk;
    randmu(ii) = expectedReturn;
end
scatter(randSig, randmu, 5);
title("Random portfolios: Risk vs ROI");
xlabel('Risk (Std. Dev.)');
ylabel('Expected Rate of Return');
end
```

*Published with MATLAB® R2018a*

---

```
function [r, Sig] = meancov(X)

r = mean(X);
Sig = cov(X);

end
```

*Published with MATLAB® R2018a*



---

```
function [rrange] = return_range(r,Sig,num)

n = length(r);

% Find the minimum bound for rate of return of a portfolio
cvx_begin quiet ;
    variable x1(n);
    maximize ( r * x1 );
    subject to ;
        sum(x1) == 1;
        min(x1) >= 0;
cvx_end;

maxr = x1;

% Find the minimum bound for rate of return of a portfolio
cvx_begin quiet;
    variable x2(n) ;
    minimize (quad_form(x2, Sig));
    subject to ;
        ones(1,n) * x2 == 1;
        min(x2) >= 0;
cvx_end;

minv = x2;

rrange = linspace(r*minv, r*maxr, num);
```

*Published with MATLAB® R2018a*

---

```
function [Y, rates, sigs] = efficient_frontier(r, Sig, num)

n = length(r);
rrange = return_range(r, Sig, num);

Y = zeros(n, num);

for jj = 1:num
    current_rate_of_return = rrange(jj);
    cvx_begin quiet ;
        variable x1(n);
        minimize (quad_form(x1, Sig));
        subject to ;
            sum(x1) == 1;
            min(x1) >= 0;
            sum(r*x1) >= current_rate_of_return; % optimize subject to
    this annual return
    cvx_end;

    Y(:,jj) = x1;
end

rates = (r* Y)' ;
sigs = sqrt(diag(Y' * Sig * Y));
```

*Published with MATLAB® R2018a*

---

```

function x = market_portfolio(f, r, Sig)

% Define function func such that
% func(sig) = 0 when risk_free_rate(sig, r, Sig) = f.
func = @(sig) risk_free_rate(sig,r,Sig) - f;

% Compute the minimum value of sig
cvx_begin quiet
    variable x(size(r))
    minimize (x' * Sig * x)
    subject to
        sum(x) == 1;
        x >= 0;
cvx_end

% Lower and upper bounds for sig
sig1 = sqrt( x' * Sig * x);
sig2 = sqrt(max(diag(Sig)));

% Use BinarySearch to solve func(sig) = 0
sig = BinarySearch(func, sig1, sig2);

% The market portfolio is the portfolio on the efficient frontier with
risk
% equal to the sig satisfying risk_free_rate(sig, r, Sig) = f.
cvx_begin quiet
    variable x(size(r))
    maximize (r'* x)
    subject to
        sum(x) == 1;
        x >= 0;
        norm(sqrtm(Sig)*x) <= sig;
cvx_end ;
end

function rate = risk_free_rate(sig, r, Sig)
n = length(r);
[sqrtSig, resnorm] = sqrtm(Sig);

% Dual multiplier lambda gives slope of efficient frontier at the
point
% (r'*x, sqrt(x'*Sig*x)), where x is the portfolio with maximum
expected
% rate of return with risk at most sig.
cvx_begin quiet
    variable x(n)
    dual variable lambda
    maximize( r'*x )
    subject to
        norm(sqrtSig*x) <= sig : lambda
        sum(x) == 1

```

---

---

```

        x >= 0
    cvx_end

    rmax = r'*x;
    % The risk-free rate is the y-intercept of the line tangent to the
    % efficient frontier at the point (r'*x, sqrt(x'*Sig*x)).
    rate = lambda*(-sig) + rmax;
end

function x = BinarySearch(func, x1, x2)
% This is a generic binary search routine.
% Given x1 and x2, with
%   func(x1) < 0 and func(x2) > 0, or
%   func(x1) > 0 and func(x2) < 0,
% returns x with abs(func(x)) < 1e-6.

disp('Binary search:');
if func(x1) > 0 && func(x2) < 0
    % Swap x1 and x2
    tmp = x1; x1 = x2; x2 = tmp;
end
ii = 0; y = Inf;
fprintf('%4s%10s%10s\n', 'Iter', 'x', '|f(x)|');
while abs(y) > 1e-6
    ii = ii + 1; x = (x1 + x2)/2; y = func(x);
    fprintf('%4d%10.2e%10.2e\n', ii, x, abs(y));
    if y < 0
        x1 = x;
    else
        x2 = x;
    end
end
disp('Done.');
```

---

*Published with MATLAB® R2018a*