Final on Fin 279 by Wanli Feng

Table of Contents

Description for the entire folder.

1. The files required to submit are the follows. *FIN279_FINAL_WFENG.m*, *FIN279_FINAL_WFENG.pdf, Results.mat, fin279Fall2014DataForFinal.xlsx*

2. The user-defined functions files are as follows. *myimport_v2.m*, *logReturn.m*, *fetchfd.m*, *ajGBM.m*.

3. Additional files for references. For convnenience, I also integrate the function in the *AllFunctionFiles.m* and *AllFunctionFiles.pdf*.

4. For the Results matrixs,

```
* In 'Results' matrix, the order of column are [j, N, dt,
YearsUntil2012(k), T, ActualSt, ul, ll, counts, mymse, ld_mean_simp];
* In 'Allmu' matrix, the dimensions are assets, historical years and
time steps. ('Allstd' matrix is alike).
```

Description for this script.

1. This script consider the compatibility as much as possible, such as user-input data.

2. About the raw data I used in this script. I have some problem on the dates of raw data in apple, so I corrected the dates a bit in the user-defined function, *myimport_v2.m* .

Import data

```
% Input the Asset you need to deal with
Assets = {'Asset17','Asset18'};
mydata = myimport_v2('fin279Fall2014DataForFinal','Data',Assets{:});
prices = mydata(:,2:end);
Dates = mydata(:,1);

% Conditions in the problem set.
Paths = [1000 5000 10000]; % number of path to use
TtoM = [30 60 180 250]; % time to maturity
Timesteps= [1 5]; % time steps
```

```
ci = 1.96; % confidence interval
YearsUntil2012 = [1 2 5 10]; % historical bootstrap data.
boottimes = 10000; % bootstrap times

clear fdate mydata;
```

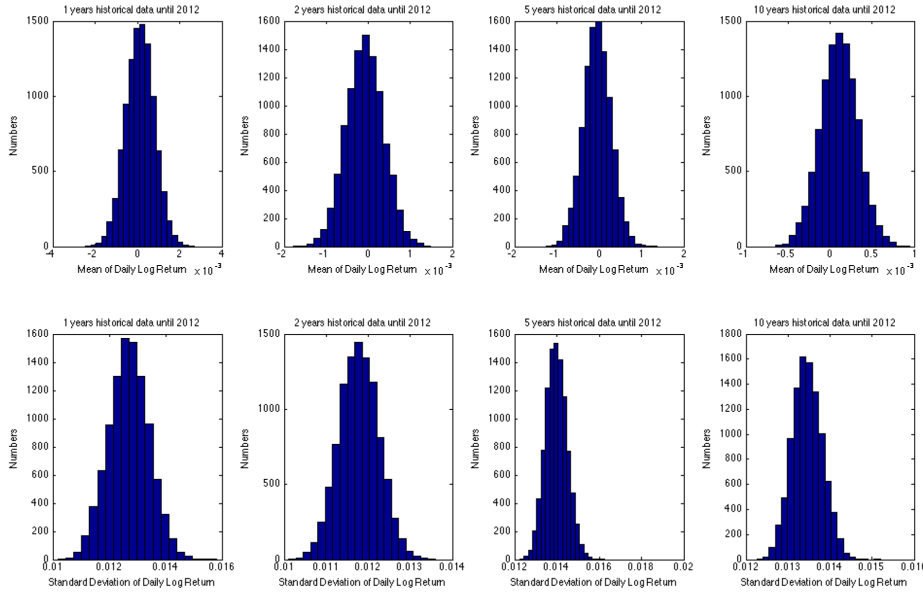
Return and Risk Estimation using Bootstrap

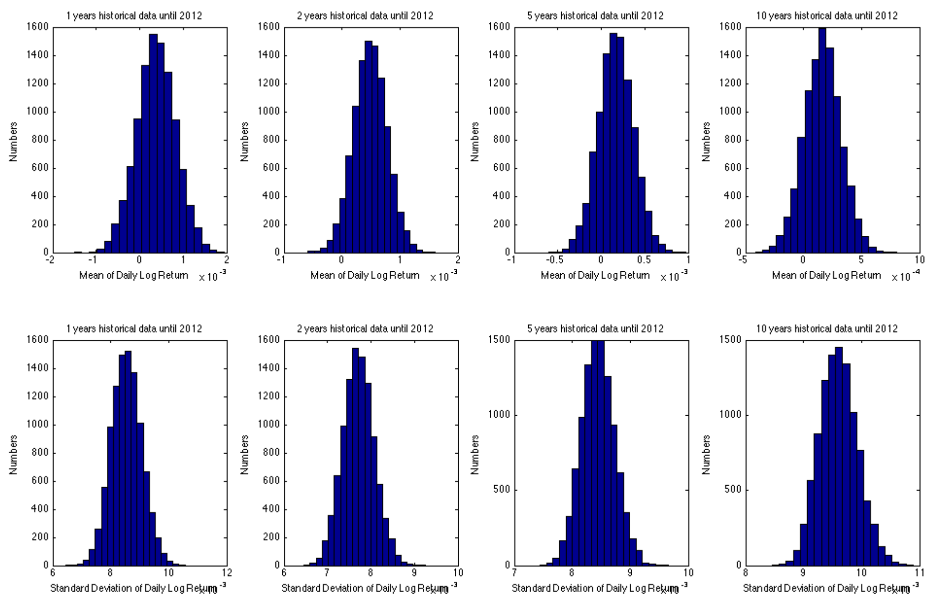
```
% Generation for improving the running speed.
AssetNum = length(Assets);
YearNum = length(YearsUntil2012);
MeanMean = NaN(AssetNum,YearNum);
StdMean = NaN(AssetNum,YearNum);
Allmu = NaN(AssetNum,YearNum,length(Timesteps));
Allstd = NaN(AssetNum,YearNum,length(Timesteps));
```

fetch() is used to fetch the first trading day indices. Here, ind2012fd is the indices of the fisrt trading day in yr year within Dates.

```
ind2012fd = fetchfd(Dates,2012);

% Bootstraping
lp1a = 0; % Factors assists looping.
lp1b = 0;
for dt = Timesteps
    lp1b = lp1b+1;
    for j = 1:AssetNum
        h1=figure(j);
        set(h1,'Units','Normalized','Position',[0.05,0.02,0.9,0.85]);
        for yr = 2012 - YearsUntil2012
            % Here indyrfd are the indices of the fisrt trading day
            % in yr year within Dates.
            indyrfd = fetchfd(Dates,yr);
            % Compute the LogRet between indyrfd and ind2012fd with
            % dt time step.
            LogRet = logReturn(prices(indyrfd:dt:ind2012fd-1,:));
            lp1a = lp1a + 1;
            [MeanMean(j,lp1a), StdMean(j,lp1a), BootMean, BootStd] = ...
                mybootstrap(LogRet(:,j),boottimes);
            if dt == 1
                subplot(2,4,lp1a)
                hist(BootMean,20)
                xlabel('Mean of Daily Log Return')
                ylabel('Numbers')
                title([num2str(2012-yr),...
                    ' years historical data until 2012'])
                subplot(2,4,lp1a+4)
                hist(BootStd,20)
                xlabel('Standard Deviation of Daily Log Return')
                ylabel('Numbers')
                title([num2str(2012-yr),...
                    ' years historical data until 2012'])
            end
        end
```

```
            lp1a = 0;
        end
        % X,Y,Z stand for Assets, HBW and time steps respectively.
        Allmu(:,:,lp1b) = MeanMean;
        Allstd(:,:,lp1b) = StdMean;
    end
```





Simulation and generate the Results matrix.

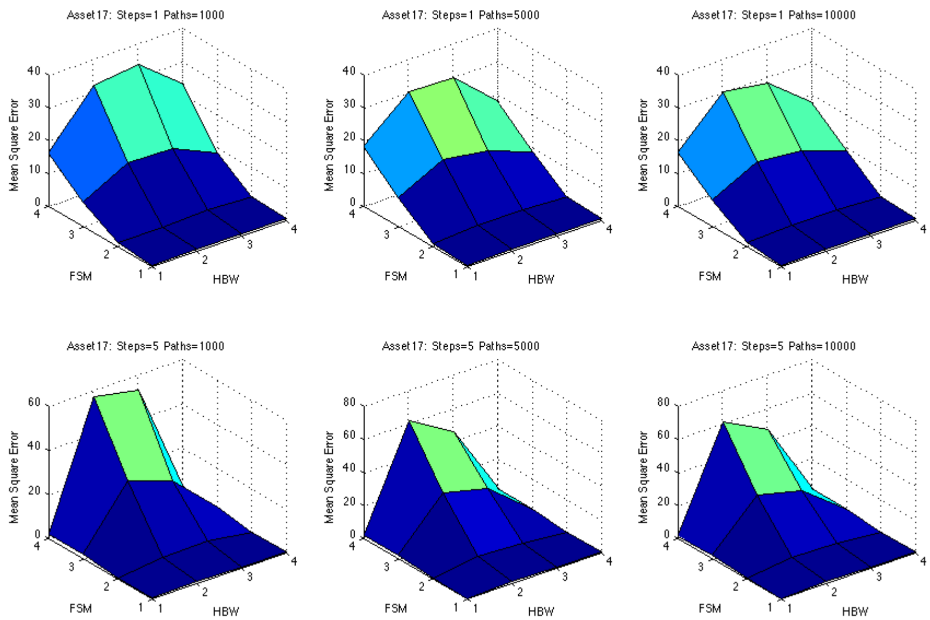
```
    % This is the number of all combinations.
```

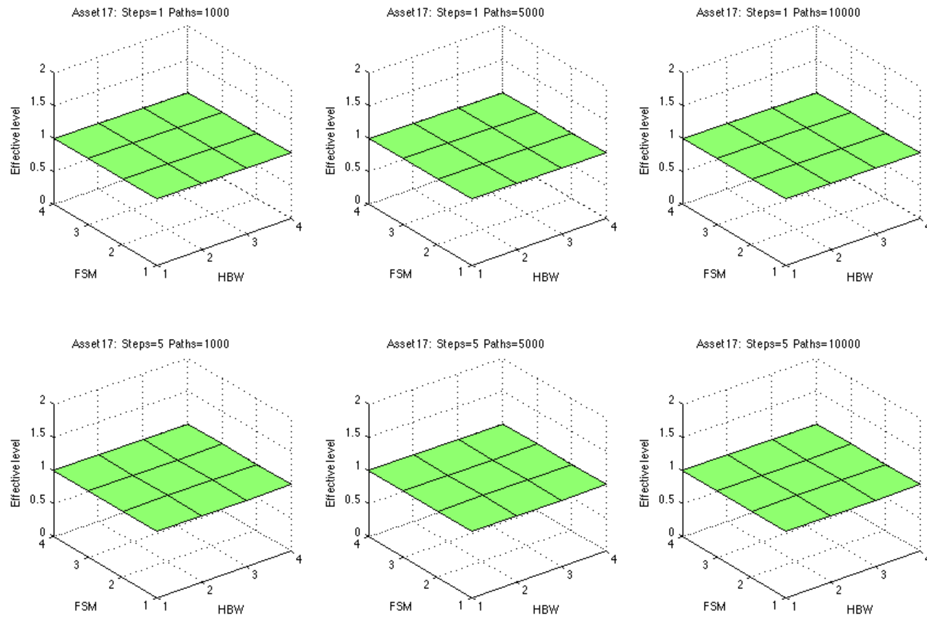
```
combnum = AssetNum*YearNum*length(TtoM)*length(Paths)*length(Timesteps);
% It is easy to know there will be 11 kinds of data we need to fill in.
Results = NaN(combnum,11);

% Simulation for each combinations and calculation of the results.
lp2a =0;
lp2b = 0;
for j = 1:AssetNum %Assets [Asset17 Asset18]
    % Initial stock price is the price of first trading day in 2012
    S0 = prices(ind2012fd,j);

    for dt = Timesteps; %time step [1 5]
        lp2b = lp2b + 1;
        for N = Paths %paths [1000 5000 10000]

            for k = 1:YearNum %HBW [1 2 5 10]
            mu = Allmu(j,k,lp2b); % growth rate
            sigma = Allstd(j,k,lp2b); % volatility

                for T = TtoM; % time to maturity [30 60 180 250]
                    S = ajGBM(mu,sigma,S0,N,T,dt);
                    Actualp = prices(ind2012fd:ind2012fd+T,j);

                    % Compute MSE between simulated prices and autual
                    % prices
                    Simp = mean(S,2);
                    indt1 = 1:dt:T+1;
                    Actualptmp = Actualp(indt1,1);
                    mymse =  mean((Simp-Actualptmp).^2);

                    % Compute the confident interval
                    SimSt = S(end,:);
                    ul = mean(SimSt)+ci*std(SimSt);
                    ll = mean(SimSt)-ci*std(SimSt);
                    lp2a = lp2a + 1;

                    % Compare the simulated prices on last day and
                    % the corresponding actual prices.
                    ActualSt = Actualp(indt1(end));
                    tf = double(ActualSt < ul & ActualSt > ll);

                    % Compute the mean of simulated prices of last
                    % day for each paths
                    ld_mean_simp = mean(SimSt);

                    % Store all results into one matrix.
                    Results(lp2a,:) = [j,N,dt,YearsUntil2012(k),T,...
                        ActualSt, ul, ll, tf, mymse,ld_mean_simp];
                end
            end
        end
    end
    lp2b = 0;
end
```

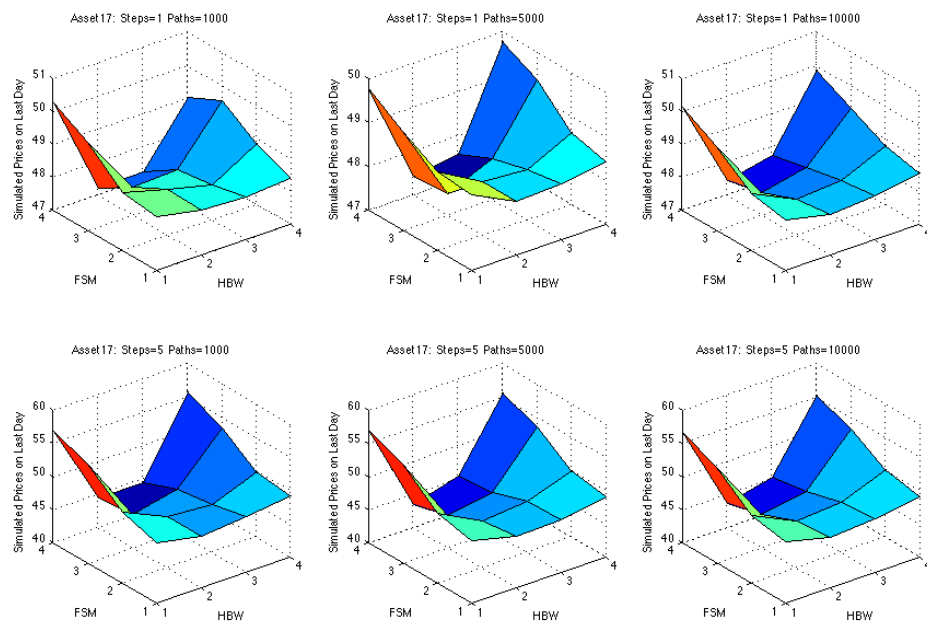
Plot 3D figures.

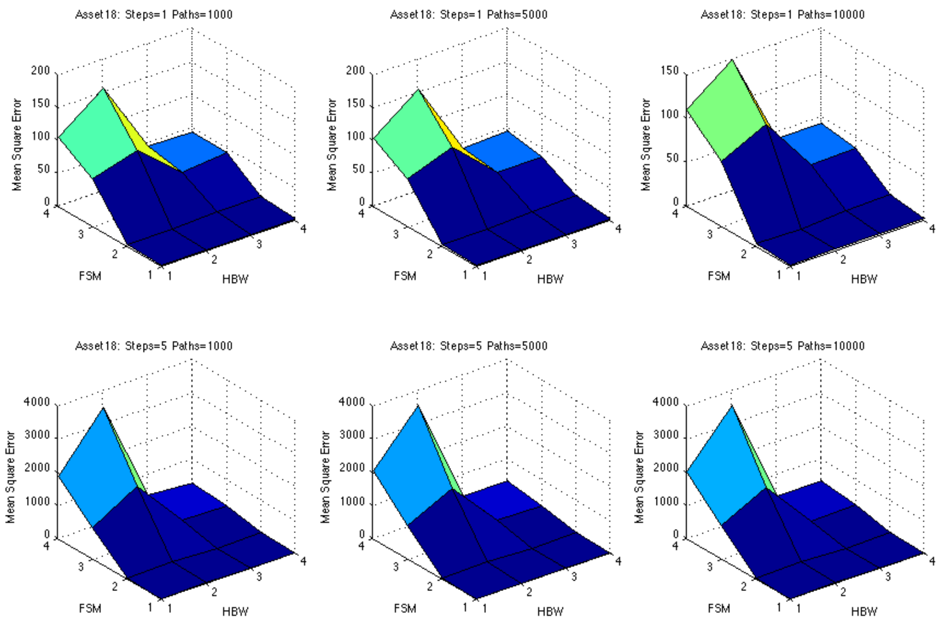
```
lp3a = 0;
lp3b = 0;

for kk1 = 1:AssetNum
    for kk2 = 1:3
        lp3b = lp3b+1;
        h2=figure(j+lp3b);
        for kk3 = Timesteps
            for kk4 = Paths

                % Prepare for the data on third dimension.

                % 96 combinations for one assets
                iii = combnum/AssetNum*(kk1-1);
                tmp = 16*lp3a+iii;
                mseplot=reshape(Results(1+tmp:16+tmp,10),...
                    YearNum,length(TtoM));
                tfplot=reshape(Results(1+tmp:16+tmp,9),...
                    YearNum,length(TtoM));
                simpplot=reshape(Results(1+tmp:16+tmp,11),...
                    YearNum,length(TtoM));
                lp3a = lp3a + 1;
                subplot(2,3,lp3a)

                % Present,respectively, the relation oof MSE, sum
                % of 'Value(1 or 0 )', and the simulated price on
                % HBW and FSM.
                switch kk2
                    % Plot the mse.
                    case 1
                    set(h2,'Units','Normalized','Position',...
                        [0.05,0.3,0.8,0.8], 'NumberTitle','off',...
                        'Name',['Asset=',num2str(kk1),':',...
                        ' Mean Squre Error'])
                    surf(1:YearNum,1:length(TtoM),mseplot)
                    xlabel('HBW');ylabel('FSM')
                    zlabel('Mean Square Error')

                    % Plot the sum of value.
                    case 2
                    set(h2,'Units','Normalized','Position',...
                        [0.05,0.3,0.8,0.8], 'NumberTitle','off',...
                        'Name',['Asset=',num2str(kk1),':',...
                        ' Effective level'])
                    surf(1:YearNum,1:length(TtoM),tfplot)
                    xlabel('HBW');ylabel('FSM')
                    zlabel('Effective level')

                    % Plot the simulated prices
                    case 3
                    set(h2,'Units','Normalized','Position',...
```

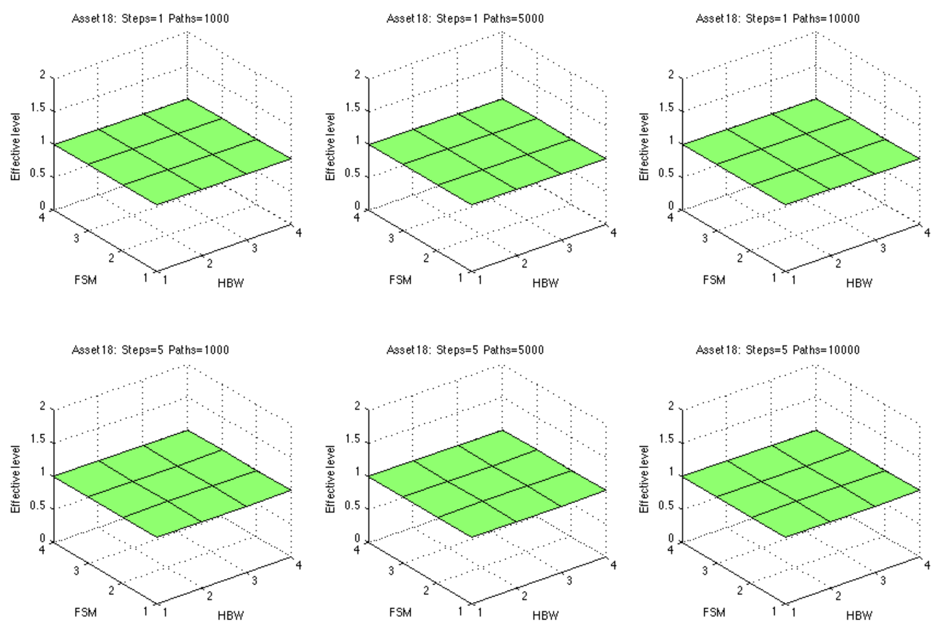
```
                        [0.05,0.3,0.8,0.8],'NumberTitle','off',...
                        'Name',['Asset=',num2str(kk1),':',...
                        ' Last Day Prices by Simulation'])
                    surf(1:YearNum,1:length(TtoM),simpplot)
                    xlabel('HBW');ylabel('FSM')
                    zlabel('Simulated Prices on Last Day')
                end
                title(eval(['''{\bf ',char(Assets(kk1)),':  Steps=',...
                num2str(kk3),'  Paths=',num2str(kk4),'}''']));
            end
        end
        lp3a = 0;
    end
end
```

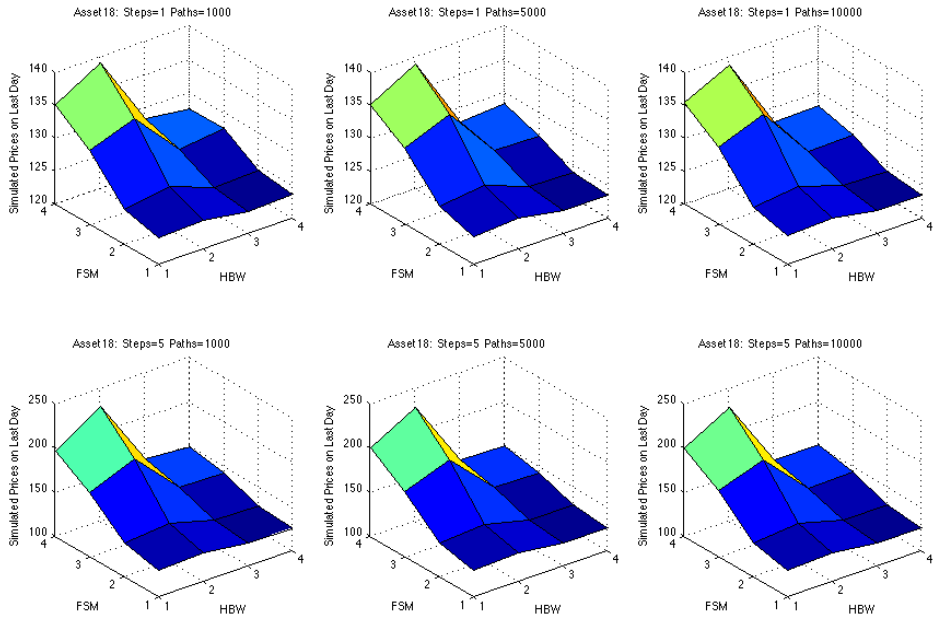



Asset17: Steps=1 Paths=1000
Asset17: Steps=1 Paths=5000
Asset17: Steps=1 Paths=10000
Asset17: Steps=5 Paths=1000
Asset17: Steps=5 Paths=5000
Asset17: Steps=5 Paths=10000
Effective level
FSM
HBW


Asset17: Steps=1 Paths=1000
Asset17: Steps=1 Paths=5000
Asset17: Steps=1 Paths=10000
Asset17: Steps=5 Paths=1000
Asset17: Steps=5 Paths=5000
Asset17: Steps=5 Paths=10000
Simulated Prices on Last Day
FSM
HBW


Asset18: Steps=1 Paths=1000
Asset18: Steps=1 Paths=5000
Asset18: Steps=1 Paths=10000
Asset18: Steps=5 Paths=1000
Asset18: Steps=5 Paths=5000
Asset18: Steps=5 Paths=10000
Mean Square Error
FSM
HBW


Asset18: Steps=1 Paths=1000
Asset18: Steps=1 Paths=5000
Asset18: Steps=1 Paths=10000
Asset18: Steps=5 Paths=1000
Asset18: Steps=5 Paths=5000
Asset18: Steps=5 Paths=10000
Effective level
FSM
HBW



Clean the workspace.

```
clear Actualp Actualptmp AssetNum boottimes dt tf combnum...
    h1 h2 indt1 indyrfd j k kk1 kk2 kk3 kk4 ld_mean_simp lp1a lp1b...
    lp2a lp2b T tmp BootMean BootStd ActualSt iii...
    lp3a lp3b lp3c lp3d mu mymse N prob S sigma Simp...
    YearNum yr mseplot tfplot simpplot ul ll SimSt LogRet ind2012fd...
    fdate counts ci S0
```

All Function file

```
% %% *myimport_v2*
% function [mydata,rawdata,text] = myimport_v2(filename,sheetname,varargin)
%     % myimport() imports the assets' data in the sheet of excel.
%     % varargin refers to each of the complete name of assets in excel.
%     % In our case, they are on first line.
%     % filename refers to the name of excel file.
%     % sheetname refers to the name of sheet in the excel file you want
%     % to use.
%
% % Import the rawdata.
% [rawdata,text] = xlsread(filename,sheetname);
%
% % Fetch the data we need
% if nargin > 2
%     col = size(text,2);
%     %The number of columns of data we need to import.
%     nVarargs = length(varargin);
%     mydata = NaN(size(rawdata,1),nVarargs);
```

```
%     for i = 1:col;
%         for j = 1:nVarargs
%             %Compare text(1,i) and the asset's name.
%             tf = strcmp(text(1,i), varargin{j});
%             if tf == true;
%                 mydata(:,j) = rawdata(:,i);
%             end
%         end
%         % Loop until we find the asset.
%         if sum(double(~isnan(mydata(:,nVarargs)))) > 0
%             break
%         end
%     end
%     % A bit correction for apple Dates. the following two lines will
%     % still work in windows.
%     fdate = rawdata(:,1);
%     Dates = fdate+datenum(2002,1,1)-fdate(1);
%     mydata = [Dates, mydata];
% end
%
%
% %% *logReturn*
% function logReturn = logReturn(price)
%     % logReturn(price) computes the log return of price.
%     % price refer to the stock volume you want to analyze.
%     logReturn = log(price(2:end,:)./price(1:end-1,:));
%
%     %concatenate NaN to the result to make it have same row and column as price.
%     row = size(price,1)-size(logReturn,1);
%     col = size(price,2);
%     logReturn = [NaN(row,col);logReturn];
%
%
% end
%
%
%
%
% %% *fetchfd*
% function fd = fetchfd(originaldates,year,month)
%
% %%
% % *Decription*
% % this function is used when the dates are not consecutive, like when we
% % are dealing with trading days.
%
% %%
% % *Instruction for the fuction*
% % fetchfd() returns the index of first trading day in specific
% % year or month.
% % originaldates is the dates you need fetch the year from. The format
% % should be number.
% % year and month are the first days of those you want.
%
```

```
% %%
% % *Example*
% % >> [fd] = fetchfd([731217:734869],2011,1)
% % >> fd =
% %            3288
%
% %%
% % consider the possibility of that we only need the firsty trading in a year.
% if nargin < 3
%     month = 1;
% end
%
% %%
% % *fetch the index*
% for i = 1:4
%     tmpnum = datenum(year,month,i);
%     tmpind = find(originaldates==tmpnum);
%     if ~isempty(tmpind)
%         fd = tmpind;
%         break
%     end
% end
%
%
% %% *ajGBM*
% function S = ajGBM(mu,sigma,S0,N,T,dt)
%
% %%
% % *this function is based on the GBM_FCN from Dr. K*
%
% %%
% % *geometric brownian motion*
% %
% % [S,t] = GBM_fcn(mu,sigma,S0,N,T,dt)
% % returns
% %       S - matrix of Wiener process paths
% %       t - corresponding timebase
% % where
% %       mu - mean
% %       sigma - volatility
% %       S0 - intial value of Stock
% %       N - number of paths
% %       T - stop time
% %       dt - timestep
% %
% % Given mu, sigma dt and S0 we generate random stock path and
% % create a histogram of the final prices. We create the return
% % series by using the formula:
% %
% %   S(n+1) = S(n)*exp((mu-sigma^2/2)*dt + sigma*dW*sqrt(dt))
% %
% % Example
% %
% %   [StockPaths,Timebase] = GBM_fcn(.5,.4,50,100,2,1/500);
```

```
% %
% % see also
% %       randn
%
% %%
% M = floor(T/dt); % Number of steps to take
%
% %%
% % *Generate the Wiener process*
% % take the random numbers created in dS and insert into the
% % bottom M+1 rows of the S matrix, so that we can use
% % cumulative product on S to generate the return paths and
% % avoid using a for loop over the rows.
%
% % Generate Wiener process
% dW = randn(M,N);
% % Create dS according to equation
% dS = exp((mu-sigma^2/2)*dt + sigma*dW*sqrt(dt));
%
% S = S0*ones(M+1,N); % Initialise S matrix
%
% % take the random numbers created in dS and insert into the
% % bottom M-1 rows of the S matrix, so that we can use
% % cumulative product on S to generate the return paths and
% % avoid using a for loop over the rows.
% S(2:end,:) = dS;
%
% % Create final paths using cumulative product
% S = cumprod(S);
%
%
% end
```

Published with MATLAB® R2014a