Table of Contents

myimport

```
function [dates,price,volume] = myimport(stock,filename,sheetname,columnsum)
    % myimport(stock,filename,sheetname,columnsum) imports the stock data in the s
    % stock refers to the complete name of stock in excel.
    % filename refers to the name of excel file.
    % sheetname refers to the name of sheet in the excel file you want to use.
    % columnsum refers to the total columns in the sheet.

% import the rawdata.
[rawdata,text] = xlsread(filename,sheetname);

% process the rawdata.
for i = 1:3:columnsum;
    tf = strcmp(text(1,i), stock); % Compare text(1,i) and stock.
    if tf == true;      % loop until we find the stock.
        mydata = rawdata(:,i:i+2);
    end
end

dates = mydata(:,1);
price = mydata(:,2);
volume = mydata(:,3);

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
```

volumeChange

```
function volumeChange = volumeChange(volume)
    % volumeChange(volume) computes the change in volume.
    % volume refer to the stock volume you want to analyze.
    volumeChange = volume(2:end,:)-volume(1:end-1,:);

    %concatenate NaNs to the result to make it have same row and column as price.
    row = size(volume,1)-size(volumeChange,1);
    col = size(volume,2);
    volumeChange = [NaN(row,col);volumeChange];
end
```

MAmean

```
function mu = MAmean(data,n,lambda)
    % UWMAmean(data,n) computes the normal moving average mean of data keeping n n
    % data can be the one like original prices, original volume, log returns, etc.
    row = size(data,1);
    col = size(data,2);
    % creat a matirx mu with same rows an columns as data.
    mu = NaN(row,col);

    if nargin == 2
        % fill out mu with the following means.
        for t = n+2:row
            mu(t,:) = mean(data(t-n:t-1,:));
        end
    end

    % EWMAmean(data,n) computes the exponentially weighted moving average mean of
    % data can be the one like original prices, original volume, log returns, etc.
    if nargin == 3
        lambdaN = lambda.^(0:n-1)';
        lambdaN = repmat(lambdaN,1,col);
        % fill out mu with the following means.
        for t = n+1:row
            mu(t,:) = (1-lambda) .* sum(lambdaN .* data(t-1:-1:t-n,:));
        end
    end

end
```

EWMAvolatility

```
function sigma = EWMAvolatility(data,mu,n,lambda)
    % EWMAvolatility(data,mu,n,lambda) computes the exponentially weighted moving
    % here the decay factor is lambda.
    % data can be the one like original prices, original volume, log returns, etc.
```

```
        % mu is supposed to the mean used here, like normal mean and weighted average

        row = size(data,1);
        col = size(data,2);

        % to transform lambda in vector to be in matrix form in case that we use it in
        lambdaN = lambda.^[0:n-1]';
        lambdaN = repmat(lambdaN,1,col);

        % creat a matirx mu with same rows an columns as data.
        var = NaN(row,col);

        % fill out mu with the following means.
        for t = n+2:row
            var(t,:) = (1-lambda) .* sum(lambdaN.*(data(t-1:-1:t-n,:)-mu(t,:)).^2);
        end
        sigma = sqrt(var);
    end
```

myBootstrap

```
    function bounds = myBootstrap(data,rep,ci,flag)
        % [BootData,cibounds] = myBootstrapB(data,rep,ci) do the bootstrap data here.
        % it returns the bootstrap dataset for rep times for the input data as BootDat
        % it also returns the bounds of ci confidence interval ci as cibounds.
        % data decides what value you need to bootstrap.
        % rep is the bootstrap repetition.
        % ci is the confidence interval.

        % delete the NaN elements in data.
        data(isnan(data)) = [];

        % bootstraping data
        N = size(data,1);
        %INDICES = fix(1+N*rand(rep*N,1)); % Random integers between 1 and N
        INDICES = randi([1,N],rep*N,1);
        BootData = data(INDICES,:);
        BootData = reshape(BootData, N, rep);

        % computing the critical points.
        alpha = 1-ci;
        bounds = norminv([alpha/2 1-alpha/2],0,1)';

        switch flag
            case 1
            % computing the bounds with ci for mean.
                BootDataMu = mean(BootData);
                MuMu = mean(BootDataMu);
                MuStd = std(BootDataMu);
                bounds = MuMu + bounds .* MuStd;
            case 2
                BootDataStd = std(BootData);
                StdMu = mean(BootDataStd);
```

```
            StdStd = std(BootDataStd);
            bounds = StdMu+ bounds .* StdStd;
    end


% QUESTION:
%1) what is the difference for the following ways to create indices for bootstrap.
% a. INDICES = fix(1+N*rand(rep*N,1)); % Random integers between 1 and N
% b. INDICES = randi([1,N],rep*N,1);
% 2) I know there arer some other way which might computing the bounds for specifi
% a. normfit()
% b. norminv()
% c. paramci()
%  d. prctile()
end
```

myplot

```
function [OutBoundsDates,OutData] = myplot(dates,data2,ci,plottype)

    switch plottype
        case 1
            hold off
            plot(dates,data2);
            hold on
%             plot(dates,repmat(ci(1),1,size(dates,1)),'r-')
            plot(dates,ci(1),'r--')
%             plot(dates,repmat(ci(2),1,size(dates,1)),'r-')
            plot(dates,ci(2),'r--')
            datetick('x','yy')
            xlabel('Date')

        case 2
            indices = find(data2>ci(2)|data2<ci(1));
            OutBoundsDates = dates(indices);
            OutData = data2(indices);
            hold on
            plot(OutBoundsDates,OutData,'g+')
    end

end
```

Addition: myBootstrapPlot

```
function [Muci,Stdci] = myBootstrapPlot(data1,rep,ci,dates,data2,type,PlotOutOfBou

    %%%%%%%%Instruction%%%%%%%%
    %[Muci,Stdci] = myBootstrapPlot(data1,rep,ci,dates,data2,type,PlotOutOfBoundsD
    % 1) this function do the bootstrap here
    % 2) it returns the mean(Muci) and standard deviation(Stdci) confidence bounds
    % 3) it can also plot the rawdata with bounds of assigned confident interval.
    % 4) it can also point out the data which are out of the bounds of
    %  assigned confident interval.
```

```
%
%%%%%%%%%%Usage%%%%%%%%%%%%
%%%% Basic parameters %%%%
% 1) data1: the raw data you want to use to bootstrap. it only can be
a vector.
% 2) rep: repetition times for bootstraping.
% 3) ci: percentage of confidence interval.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Other parameters %%%%
% You can skip the following if you do not need to do the plot. If
you do, please read the following.
% 1) dates: the dates correspond to the raw data.
% 2) data2: the data you want to compare with the bounds of assigned
confidence interval.
% 3) type: the type of your raw data. i.e. 'Volatility' or 'Mean of Daily Trad
% 4) PlotOutOfBoundsData: just input 'y' to plot the points out of
bounds of assigned confidence interval.


% delete the NaN elements in data.
data1(isnan(data1)) = [];

% bootstraping data
N = size(data1,1);
%INDICES = fix(1+N*rand(rep*N,1)); % Random integers between 1 and N
INDICES = randi([1,N],rep*N,1);
BootData = data1(INDICES,:);
BootData = reshape(BootData, N, rep);

% computing the critical points.
alpha = 1-ci;
bounds = norminv([alpha/2 1-alpha/2],0,1)';

% computing the bounds with ci for mean.
BootDataMu = mean(BootData);
MuMu = mean(BootDataMu);
MuStd = std(BootDataMu);
Muci = MuMu + bounds .* MuStd;

% computing the bounds with ci for standard deviation.
BootDataStd = std(BootData);
StdMu = mean(BootDataStd);
StdStd = std(BootDataStd);
Stdci = StdMu+ bounds .* StdStd;

if nargin >= 6
    switch type
        case 'Volatility'
            tempci = Stdci;
            ylname = 'Volatility';
        case 'mVolume'
            tempci = Muci;
            ylname = 'Mean of Daily Trading Volume';
    end
```

```
    end

    if nargin == 6

        hold off
        plot(dates,data2);
        hold on
        plot(dates,tempci(1),'r-')
        plot(dates,tempci(2),'r-')
        datetick('x','yy')
        xlabel('Date')
        ylabel(ylname)
        title(['Repetition = ',num2str(rep)]);
    end

    if nargin == 7
        if PlotOutOfBoundsData == 'y'
            indices = find(data2>tempci(2)|data2<tempci(1));
            OutBoundsDates = dates(indices);
            OutData = data2(indices);
            hold on
            plot(OutBoundsDates,OutData,'g+')
        end
    end

end
```

Published with MATLAB® R2014a