Table of Contents

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 time-base 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