HW#2

Name: **Jingyi Wu**

Due Date: February 25, 2019

Course Code: ECON 623 Forecasting Financial Markets

Instructor: Professor Andrew Patton

Partner: Xiaomeng Han

Exercise 1

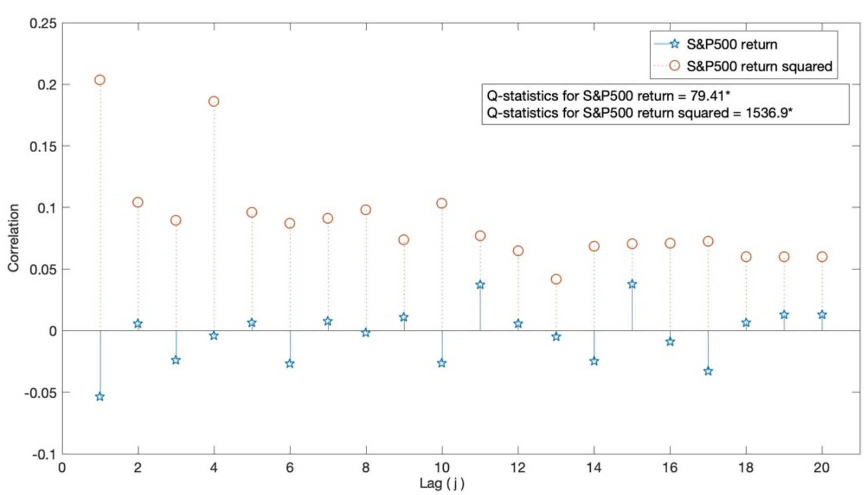
(a)

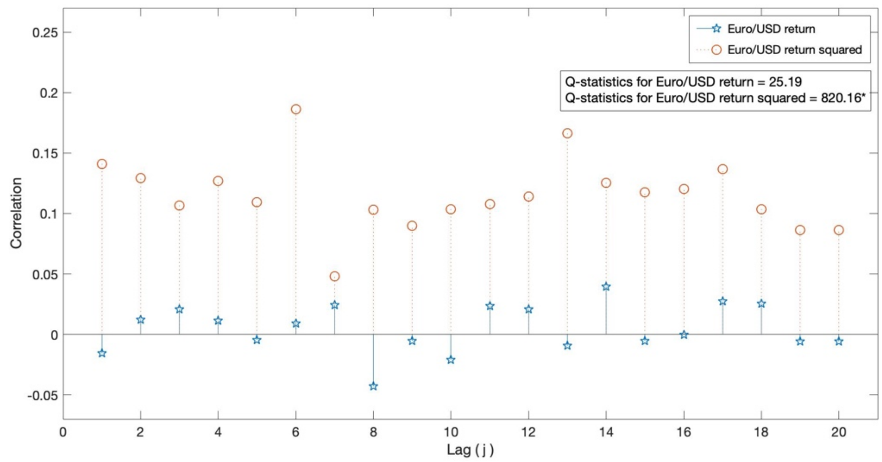
```
function lb = LBtest(X,L)

T = length(X); %return number of observation
Xbar = mean(X); %return mean of X
Xvar = var(X); %return variance of X
sum1(1) = 0; %set default value for loop1
sum2(1) = 0; %set default value for loop2
p = 0;

for j = 1:L
        for t = j+1:T   %calculate jth autocovariance coefficient
            sum1 = sum1 + (X(t,1)-Xbar)*(X(t-j,1)-Xbar);
        end
        sum1 = (1/(T-j))*sum1;
        p(j) = sum1/Xvar; %calculate sample autocorrelation coefficient
        sum2 = sum2 + (1/(T-j))*(p(j))^2;
        lb.test = T*(T+2)*sum2; %return LB test statistics
        p = [p p(j)];
end
lb.pvalue = chi2cdf(lb.test,L,'upper');  %return p value
lb.p = p
```

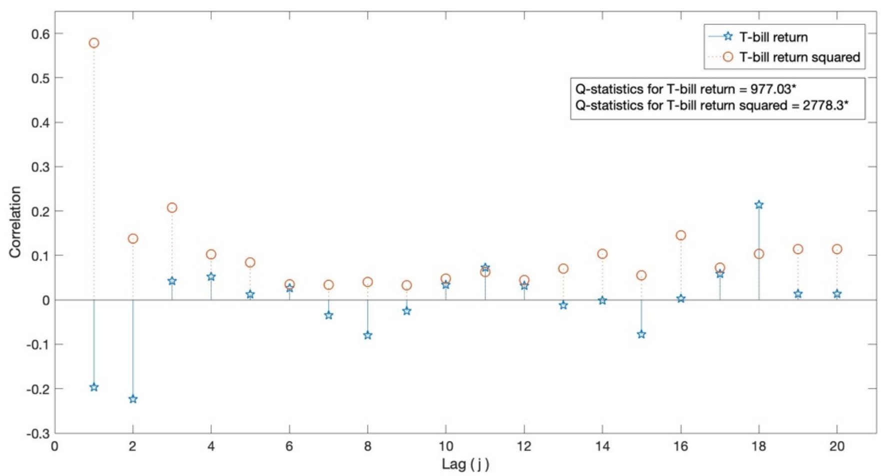
(b)

```
function  rlb = robustLBtest(X,L)

Y = X(L+1:end);
Indep = ones(length(Y), 1); %set default independent variable(intercept)

for i =1:L %return independent variable
    Indep = [Indep X(L+1-i:end-i)];
end

results = nwest(Y,Indep,L); %return Newey-West regression using nwest()

rlb.beta = results.beta; %return robust betas
Betas = rlb.beta(2:end); %drop the constant coefficient
CovarianceMatrixBetas = results.vcv(2:end, 2:end); % drop the constant
R = eye(L); %define the restriction matrix
%return test statistics based on Newey-West regression
rlb.test = (R*Betas - zeros(L,1))'*((R'*CovarianceMatrixBetas*R)\(R*Betas - zeros(L,1)));
%return p Value
rlb.pvalue = chi2cdf(rlb.test,L,'upper');

end
```

(c)







Code:

```
%% Exercise 1
%(a) see function for detail

%(b) see function for detail

%(c)
L = 20; %set Lag = 20

SP = csvread('^GSPC.csv', 1, 0);
P_SP = SP(:, 6);
ret_SP = log(P_SP(2:end)./P_SP(1:end-1)); %calculate log return
ret_SP_sqr = ret_SP.^2; %calculate log return squared

lb_SP = LBtest(ret_SP,L); %return lbtest for SP return
p_SP = lb_SP.p(2:end)'; %return sample correlation for SP return
lb_SP_sqr = LBtest(ret_SP_sqr,L); %return lbtest for SP return squared
p_SP_sqr = lb_SP_sqr.p(2:end)'; %return sample correlationfor SP return
squared

plot(p_SP,'p:','MarkerSize',10) %plotting
hold on
```

```
plot(p_SP_sqr,'o:','MarkerSize',10)
hold off

%same for Exchange rate
Ex = csvread('Exc.csv', 1, 0);
P_Ex = Ex(:, 5);
ret_Ex = log(P_Ex(2:end)./P_Ex(1:end-1));
ret_Ex_sqr = ret_Ex.^2;

lb_Ex = LBtest(ret_Ex,L);
p_Ex = lb_Ex.p(2:end)';
lb_Ex_sqr = LBtest(ret_Ex_sqr,L);
p_Ex_sqr = lb_Ex_sqr.p(2:end)';

plot(p_Ex,'p:','MarkerSize',10)
hold on
plot(p_Ex_sqr,'o:','MarkerSize',10)
hold off

%same for 3-month T-bill
Tbill = csvread('Tbill.csv', 1, 0);
P_Tbill = Tbill(:, 2);
P_Tbill = 100./(1+P_Tbill).^0.25;
ret_Tbill = log(P_Tbill(2:end)./P_Tbill(1:end-1));
ret_Tbill_sqr = ret_Tbill.^2;

lb_Tbill = LBtest(ret_Tbill,L);
p_Tbill = lb_Tbill.p(2:end)';
lb_Tbill_sqr = LBtest(ret_Tbill_sqr,L);
p_Tbill_sqr = lb_Tbill_sqr.p(2:end)';

plot(p_Tbill,'p:','MarkerSize',10)
hold on
plot(p_Tbill_sqr,'o:','MarkerSize',10)
hold off
```

Exercise 2

(a)

```
function result = x3(x)

    result = abs(x)^3;

end
```

(b)

```
>> [X,FVAL,EXITFLAG] = fminunc(@x3,2)

Local minimum found.

Optimization completed because the size of the gradient is less than
the default value of the optimality tolerance.

<stopping criteria details>


X =

    0.0020


FVAL =

   8.3202e-09


EXITFLAG =

     1
```

(c)

```
>> A = -1;
b = -1;
[X,FVAL,EXITFLAG] = fmincon('x3', 3, A, b)

Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the default value of the optimality tolerance,
and constraints are satisfied to within the default value of the constraint tolerance.

<stopping criteria details>


X =

    1.0000


FVAL =

    1.0000


EXITFLAG =

     1
```

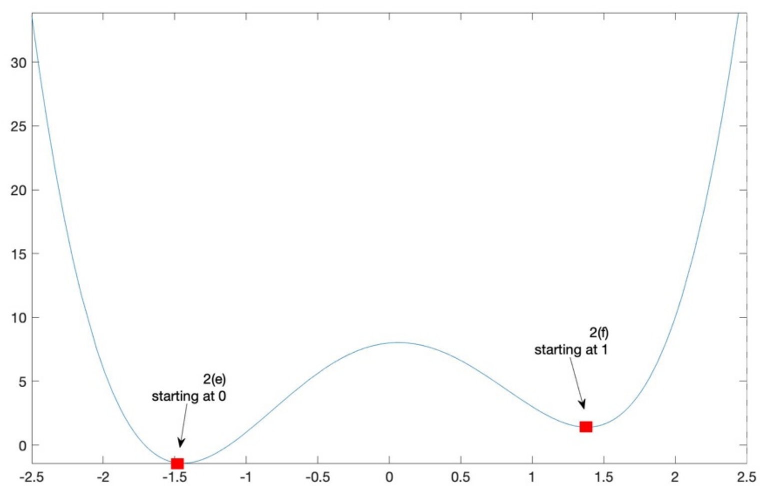
(e)

```
>> [X,FVAL,EXITFLAG] = fminunc(@x4,0)

Local minimum found.

Optimization completed because the size of the gradient is less than
the default value of the optimality tolerance.

<stopping criteria details>


X =

   -1.4445


FVAL =

   -1.4295


EXITFLAG =

     1
```

(f)

```
>> [X,FVAL,EXITFLAG] = fminunc(@x4,1)

Local minimum found.

Optimization completed because the size of the gradient is less than
the default value of the optimality tolerance.

<stopping criteria details>


X =

    1.3819


FVAL =

    1.3982


EXITFLAG =

     1
```

(g)



Code:

```
%% Exercise 2
%(a) see funtion for detail

%(b)
[X,FVAL,EXITFLAG] = fminunc(@x3,2)

%(c)
A = -1;
b = -1;
[X,FVAL,EXITFLAG] = fmincon('x3', 3, A, b)

%(d) see function for detail

%(e)
[X,FVAL,EXITFLAG] = fminunc(@x4,0)
```

```
%(f)
[X,FVAL,EXITFLAG] = fminunc(@x4,1)

%(g)
fplot(@x4,[-2.5,2.5]);
```

Exercise 3

(a)

```
function [LL] = LL_normal(theta, X)

mu = theta(1);
sig2 = theta(2);
T = length(X);
sum = 0;

    for t = 1:T

        sum = sum + log(1/(sqrt(sig2*2*pi))*exp(-(X(t)-mu)^2/(2*sig2)));

    end

LL = -sum;

end
```

(b)

```
>> rng(1);
R = 1 + 2. *randn(1000,1);
mean_R = mean(R)
var_R = var(R)

X = R;
x0 = [0,1];
A = [0,-1];
b = 0;
Aeq = [];
beq = [];
theta = fmincon(@(theta) LL_normal(theta,X) , x0, A, b, Aeq, beq)

mean_R =

    0.9735


var_R =

    3.9879


Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the default value of the step size tolerance and constraints are
satisfied to within the default value of the constraint tolerance.

<stopping criteria details>


theta =

    0.9735    3.9839
```

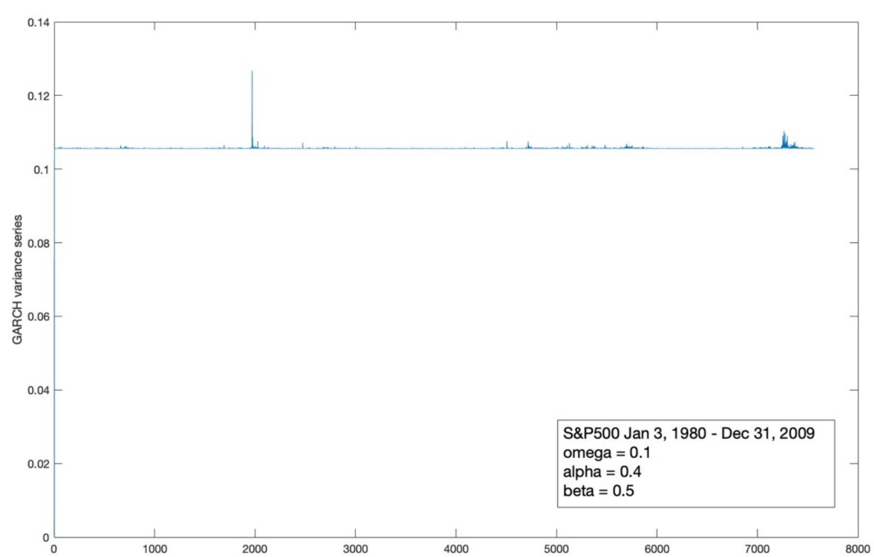
Code:

```
%% Exercise 3
%(a) see function for detail

%(b)
%Generate data set(sample) from a normal distribution with mean 1 and
standard deviation 2.
rng(1);
R = 1 + 2. *randn(1000,1);
mean_R = mean(R) %calculate sample mean
var_R = var(R) %calculate sample variance

X = R; %insert sample
x0 = [0,1]; %set starting value
A = [0,-1]; %set constraint
b = 0;
Aeq = []; %set non-equal constraint
beq = [];
theta = fmincon(@(theta) LL_normal(theta,X) , x0, A, b, Aeq, beq) %maxmize
Likelihood
```

Exercise 4

(a)

```
function [sigma2] = garch_variance(theta,data)

omega = theta(1);
alpha = theta(2);
beta = theta(3);
sigma2(1) = var(data);

for i = 2:length(data)
    sigma2(i) = omega + alpha*(data(i-1))^2 + beta*(sigma2(i-1))^2;
end

end
```

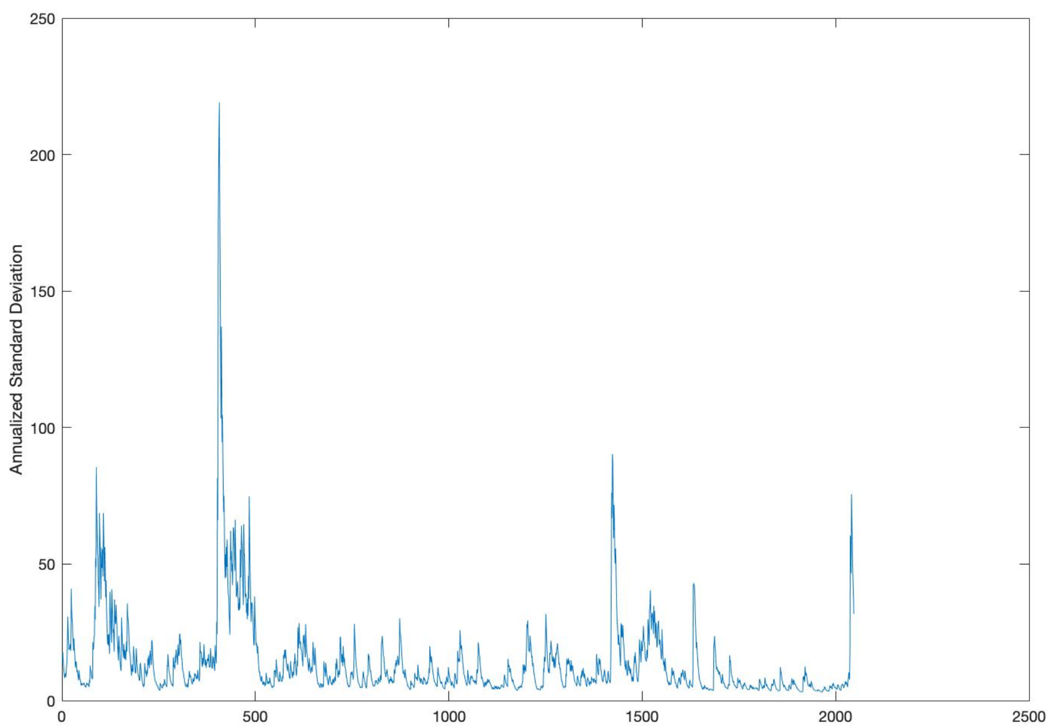


Code:

```
%% Exercise 4
%see function for detail
SP = csvread('^GSPC.csv', 1, 0); %import data
P_SP = SP(:, 6);
ret_SP = log(P_SP(2:end)./P_SP(1:end-1));
demeaned_ret_SP = ret_SP - mean(ret_SP);

data = demeaned_ret_SP;
theta = [0.1; 0.4; 0.5];
sigma2 = garch_variance(theta,data)';
plot(sigma2);
```

Exercise 5

Output:



Code:

```
%% Exercise 5

SP500 = csvread('SP_from10to17.csv', 1, 1);
SP500 = 100*SP500(:, 7);

results = nwest(SP500, ones(length(SP500),1), 5);
resid = SP500 - results.yhat;

addpath('/Users/killshadows/Desktop/DUKE/COURSES/SPRING2019/ECON623/TA
Session/3/mfe-toolbox-master/univariate')
```

```
addpath('/Users/killshadows/Desktop/DUKE/COURSES/SPRING2019/ECON623/TA Session/3/mfe-toolbox-master/distributions')

parameters = tarch(resid,1,0,1);

sigmasqr = NaN(length(resid)+1, 1);
sigmasqr(1) = var(resid);

omega = parameters(1);
alpha = parameters(2);
beta = parameters(3);

for i = 2:length(resid)+1

    sigmasqr(i) = omega + alpha*resid(i-1)^2 + beta*sigmasqr(i-1);

end

annual_sd = sqrt(252)*sigmasqr;
plot(annual_sd);
```