HW#4

Name: **Jingyi Wu**

Due Date: April 14, 2019

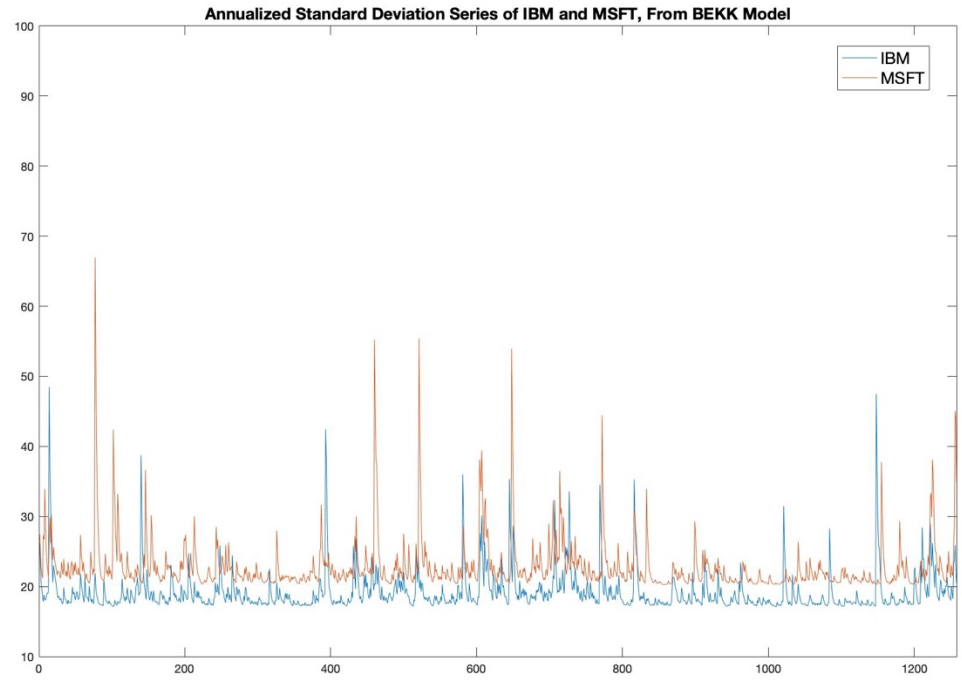
Course Code: ECON 623 Forecasting Financial Markets

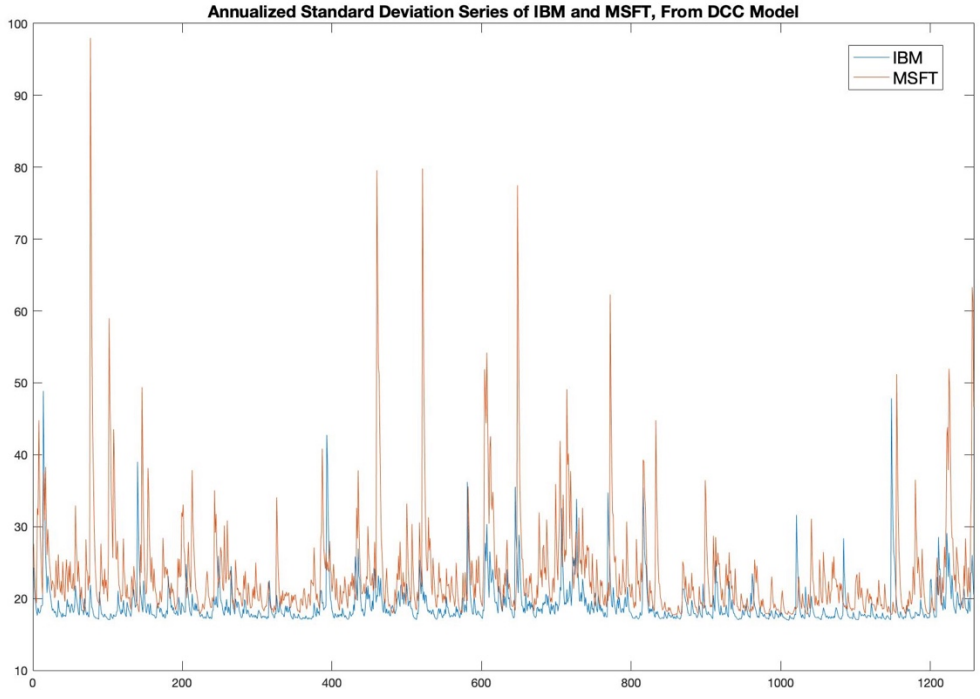
Instructor: Professor Andrew Patton

Exercise 1

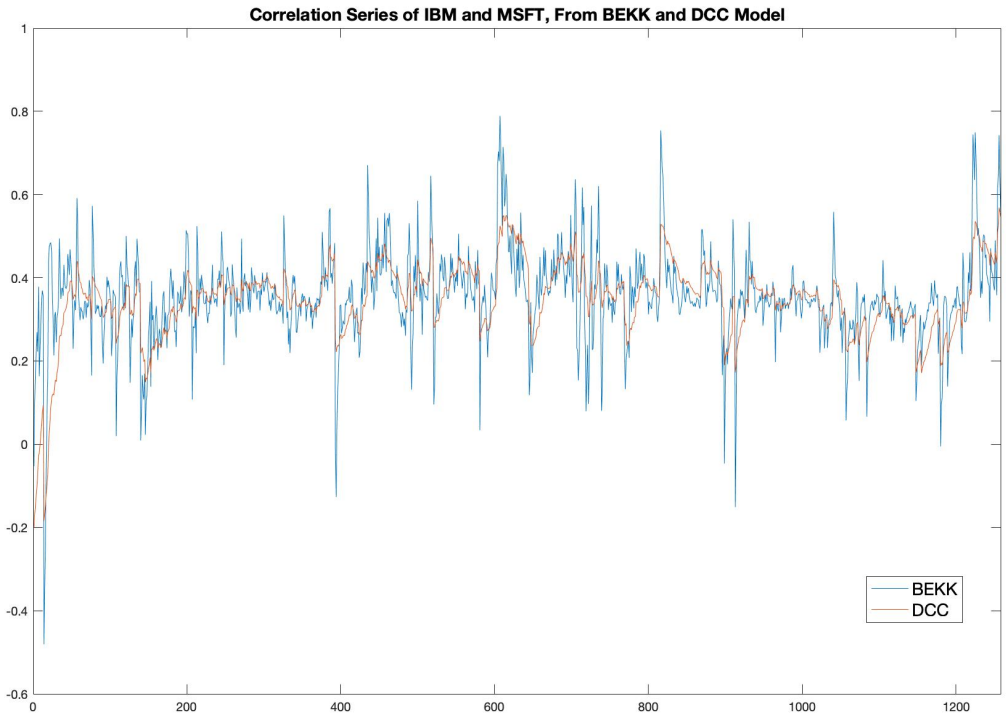
Data: IBM and MSFT on 4/2/13-3/29/18
Finance.yahoo.com

(c)





(d)



(d)

MZ test: H0: Beta0 = 0 and beta1 = 1 H1: Beta0 != 0 and beta1 != 1

1. test for IBM in bekk

```
TestValue1 =

   12.1192


CriticalValue1 =

    5.9915


ans =

  logical

   0
```

MZ test statistics is larger than the critical value. So, we can reject the null. Therefore, there is evidence to show that BEKK model for IBM is optimal.

2. test for MSFT in BEKK

```
TestValue2 =

    2.2549


CriticalValue2 =

    5.9915


ans =

  logical

   1
```

MZ test statistics is smaller than the critical value. So, we cannot reject the null. Therefore, there is no evidence to show that BEKK model for MSFT is optimal.

3. test for IBM in DCC

```
TestValue3 =

   12.9797


CriticalValue3 =

    5.9915


ans =

  logical

   0
```

MZ test statistics is larger than the critical value. So, we can reject the null. Therefore, there is evidence to show that DCC model for IBM is optimal.

4. test for MSFT in DCC

```
TestValue4 =

   46.8541


CriticalValue4 =

    5.9915


ans =

  logical

   0
```

MZ test statistics is larger than the critical value. So, we can reject the null. Therefore, there is evidence to show that DCC model for MSFT is optimal.

5. test for Covariance in BEKK

```
TestValue5 =

    4.2994


CriticalValue5 =

    5.9915


ans =

  logical

   1
```

MZ test statistics is smaller than the critical value. So, we cannot reject the null. Therefore, there is no evidence to show that BEKK model for Covariance is optimal.

6. test for Covariance in DCC

```
TestValue6 =

    4.2630


CriticalValue6 =

    5.9915


ans =

  logical

   1
```

MZ test statistics is smaller than the critical value. So, we cannot reject the null. Therefore, there is no evidence to show that DCC model for Covariance is optimal.

(e)

DM test: H0: E(dbar) = 0. H1: E(dbar) !=0

1.w1 = [1,0]

DM1	CriticalValue
-1.7642	1.96

DM test statistics is smaller than the critical value. So, we cannot reject the null. The two estimation perform the same.

2.w2 = [0.5,0.5]

DM2	CriticalValue
-1.4141	1.96

DM test statistics is smaller than the critical value. So, we cannot reject the null. The two estimation perform the same.

3.w2 = [0,1]

DM3	CriticalValue
-2.6146	1.96

DM test statistics is larger than the critical value. So, we can reject the null. The DCC model perform better.

Code:

```
%% Exericise 1
cd('/Users/killshadows/Desktop/DUKE/COURSES/SPRING2019/ECON623/TA Session/5')
addpath('/Users/killshadows/Desktop/DUKE/COURSES/SPRING2019/ECON623/TA
Session/5/mfe-toolbox-master/multivariate')
addpath('/Users/killshadows/Desktop/DUKE/COURSES/SPRING2019/ECON623/TA
Session/5/mfe-toolbox-master/utility')
addpath('/Users/killshadows/Desktop/DUKE/COURSES/SPRING2019/ECON623/TA
Session/5/mfe-toolbox-master/distributions')
addpath('/Users/killshadows/Desktop/DUKE/COURSES/SPRING2019/ECON623/TA
Session/5/mfe-toolbox-master/univariate')

IBM = csvread('IBM.csv', 1, 1);
MSFT = csvread('MSFT.csv', 1, 1);

IBM = IBM(:, 5);
MSFT = MSFT(:, 5);

IBM = 100*price2ret(IBM);
MSFT = 100*price2ret(MSFT);


% (a) BEKK Model
data = [IBM MSFT];
data = [data(:, 1)-mean(data(:, 1)) data(:, 2)-mean(data(:, 2))]; %obtain
residual

[~, ~, Ht_a, ~, ~] = bekk(data,[], 1,0, 1); %estimate multivariate volatility
by BEKK

vol_bekk_IBM = squeeze(Ht_a(1,1,:)); %obtain volatility of IBM from BEKK
model
sd_bekk_IBM = sqrt(252) * sqrt(vol_bekk_IBM); %annualized sd of IBM from BEKK
model

vol_bekk_MSFT = squeeze(Ht_a(2,2,:)); %obtain volatility of MSFT from BEKK
model
sd_bekk_MSFT = sqrt(252) * sqrt(vol_bekk_MSFT); %annualized sd of IBM from
BEKK model

plot(sd_bekk_IBM)
hold on
plot(sd_bekk_MSFT)
hold off
title('Annualized Standard Deviation Series of IBM and MSFT, From BEKK
Model')
```

```
% (b) DCC Model
[~, ~, Ht_b, ~, ~] = dcc(data,[], 1,0, 1); %estimate multivariate volatility
by DCC

vol_dcc_IBM = squeeze(Ht_b(1,1,:)); %obtain volatility of IBM from DCC model
sd_dcc_IBM = sqrt(252) * sqrt(vol_dcc_IBM); %annualized sd of IBM from DCC
model

vol_dcc_MSFT = squeeze(Ht_b(2,2,:)); %obtain volatility of MSFT from DCC
model
sd_dcc_MSFT = sqrt(252) * sqrt(vol_dcc_MSFT); %annualized sd of IBM from DCC
model

plot(sd_dcc_IBM)
hold on
plot(sd_dcc_MSFT)
hold off
title('Annualized Standard Deviation Series of IBM and MSFT, From DCC Model')


% (d) correlation of two model
cov_bekk = squeeze(Ht_a(1,2,:));
corr_bekk = cov_bekk./sqrt(vol_bekk_IBM.*vol_bekk_MSFT);

cov_dcc = squeeze(Ht_b(1,2,:));
corr_dcc = cov_dcc./sqrt(vol_dcc_IBM.*vol_dcc_MSFT);

plot(corr_bekk)
hold on
plot(corr_dcc)
hold off
title('Correlation Series of IBM and MSFT, From BEKK and DCC Model')


% (e) Mincer-Zarnowitz test
proxy_var_IBM = data(:,1).^2;
proxy_var_MSFT = data(:,2).^2;
proxy_cov = data(:,1).*data(:,2);

% test for IBM in bekk
Intercept = ones(size(vol_bekk_IBM, 1), 1);
X = [Intercept vol_bekk_IBM];
Y = proxy_var_IBM;
results = nwest(Y,X);
beta1 = results.beta;
se1 = results.se;
vcv1 = results.vcv;
R1 = [1 0
      0 1];
%Calculate test statistics
TestValue1 = (R1*beta1 - [0; 1])'*((R1'*vcv1*R1)\(R1*beta1 - [0; 1]))
%Calculate Critical Value for 5% significance level
CriticalValue1 = chi2inv(1-0.05,2)
%Test if test statistics is less than the Critical Value
TestValue1 < CriticalValue1 %reject

% test for MSFT in bekk
Intercept = ones(size(vol_bekk_MSFT, 1), 1);
```

```
X = [Intercept vol_bekk_MSFT];
Y = proxy_var_MSFT;
results = nwest(Y,X);
beta2 = results.beta;
se2 = results.se;
vcv2 = results.vcv;
R2 = [1 0
      0 1];
%Calculate test statistics
TestValue2 = (R2*beta2 - [0; 1])'*((R2'*vcv2*R2)\(R2*beta2 - [0; 1]))
%Calculate Critical Value for 5% significance level
CriticalValue2 = chi2inv(1-0.05,2)
%Test if test statistics is less than the Critical Value
TestValue2 < CriticalValue2 %fail to reject

% test for IBM in dcc
Intercept = ones(size(vol_dcc_IBM, 1), 1);
X = [Intercept vol_dcc_IBM];
Y = proxy_var_IBM;
results = nwest(Y,X);
beta3 = results.beta;
se3 = results.se;
vcv3 = results.vcv;
R3 = [1 0
      0 1];
%Calculate test statistics
TestValue3 = (R3*beta3 - [0; 1])'*((R3'*vcv3*R3)\(R3*beta3 - [0; 1]))
%Calculate Critical Value for 5% significance level
CriticalValue3 = chi2inv(1-0.05,2)
%Test if test statistics is less than the Critical Value
TestValue3 < CriticalValue3 %reject

% test for MSFT in dcc
Intercept = ones(size(vol_dcc_MSFT, 1), 1);
X = [Intercept vol_dcc_MSFT];
Y = proxy_var_MSFT;
results = nwest(Y,X);
beta4 = results.beta;
se4 = results.se;
vcv4 = results.vcv;
R4 = [1 0
      0 1];
%Calculate test statistics
TestValue4 = (R4*beta4 - [0; 1])'*((R4'*vcv4*R4)\(R4*beta4 - [0; 1]))
%Calculate Critical Value for 5% significance level
CriticalValue4 = chi2inv(1-0.05,2)
%Test if test statistics is less than the Critical Value
TestValue4 < CriticalValue4 %reject

% test for covariance in bekk
Intercept = ones(size(cov_bekk, 1), 1);
X = [Intercept cov_bekk];
Y = proxy_cov;
results = nwest(Y,X);
beta5 = results.beta;
se5 = results.se;
vcv5 = results.vcv;
R5 = [1 0
      0 1];
```

```
%Calculate test statistics
TestValue5 = (R5*beta5 - [0; 1])'*((R5'*vcv5*R5)\(R5*beta5 - [0; 1]))
%Calculate Critical Value for 5% significance level
CriticalValue5 = chi2inv(1-0.05,2)
%Test if test statistics is less than the Critical Value
TestValue5 < CriticalValue5 %fail to reject

% test for covariance in dcc
Intercept = ones(size(cov_dcc, 1), 1);
X = [Intercept cov_dcc];
Y = proxy_cov;
results = nwest(Y,X);
beta6 = results.beta;
se6 = results.se;
vcv6 = results.vcv;
R6 = [1 0
      0 1];
%Calculate test statistics
TestValue6 = (R6*beta6 - [0; 1])'*((R6'*vcv6*R6)\(R6*beta6 - [0; 1]))
%Calculate Critical Value for 5% significance level
CriticalValue6 = chi2inv(1-0.05,2)
%Test if test statistics is less than the Critical Value
TestValue6 < CriticalValue6 %fail to reject


% (f) Diebold-Mariano test
%w1 = [1,0]
e_bekk_w1_squared = (proxy_var_IBM - vol_bekk_IBM).^2;
e_dcc_w1_squared = (proxy_var_IBM - vol_dcc_IBM).^2;
d1bar = mean(e_bekk_w1_squared - e_dcc_w1_squared);
d1var = var(e_bekk_w1_squared - e_dcc_w1_squared);
DM1 = d1bar/sqrt(d1var/1258);
CriticalValue = 1.96;
Summary1 = table(DM1, CriticalValue) %fail to reject, perform the same

%w2 = [0.5,0.5]
w2 = [0.5, 0.5];
proxy_w2 = (w2*data').^2';
vol_bekk_w2 = 0.25*vol_bekk_IBM+0.25*vol_bekk_MSFT+0.5*cov_bekk;
vol_dcc_w2 = 0.25*vol_dcc_IBM+0.25*vol_dcc_MSFT+0.5*cov_dcc;
e_bekk_w2_squared = (proxy_w2 - vol_bekk_w2).^2;
e_dcc_w2_squared = (proxy_w2 - vol_dcc_w2).^2;
d2bar = mean(e_bekk_w2_squared - e_dcc_w2_squared);
d2var = var(e_bekk_w2_squared - e_dcc_w2_squared);
DM2 = d2bar/sqrt(d2var/1258);
CriticalValue = 1.96;
Summary2 = table(DM2, CriticalValue) %fail to reject, perform the same

%w3 = [0,1]
e_bekk_w3_squared = (proxy_var_MSFT - vol_bekk_MSFT).^2;
e_dcc_w3_squared = (proxy_var_MSFT - vol_dcc_MSFT).^2;
d3bar = mean(e_bekk_w3_squared - e_dcc_w3_squared);
d3var = var(e_bekk_w3_squared - e_dcc_w3_squared);
DM3 = d3bar/sqrt(d3var/1258);
CriticalValue = 1.96;
Summary3 = table(DM3, CriticalValue) %reject, DCC perform better
```

Exercise 2

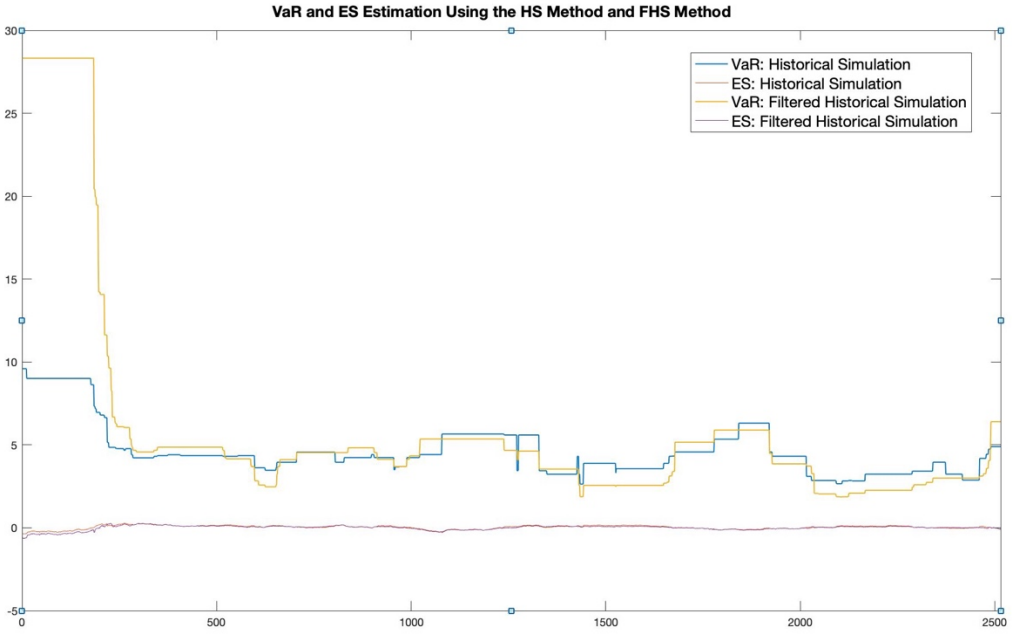
data input: AAPL from 1/1/2008 - 12/31/2018

(a)

VaR_un_1	VaR_un_5	VaR_un_10
-5.6014	-2.8754	-2.1142

ES_un_1	ES_un_5	ES_un_10
-7.6757	-4.6601	-3.5614

(d)



(e)

1. test HS method

```
beta7 =

    0.9855
   -0.0143
    0.0038


TestValue7 =

   9.1222e+06


CriticalValue7 =

    5.9915


ans =

  logical

   0
```

Reject the null

2. test FHS method

```
beta8 =

    0.9758
   -0.0035
    0.0013


TestValue8 =

   4.1544e+05


CriticalValue8 =

    5.9915


ans =

  logical

   0
```

Reject the null

(f)

1.test HS method

```
beta9 =

   16.8097
   18.9243
    1.3289


TestValue9 =

   2.5595e+03


CriticalValue9 =

    5.9915


ans =

  logical

   0
```

Reject the null

2. test FHS method

```
beta10 =

   16.4936
    9.2802
    2.2405


TestValue10 =

   2.6902e+03


CriticalValue10 =

    5.9915


ans =

  logical

   0
```

Reject the null

(g)

DM4	CriticalValue
-13.394	1.96

DM test statistics is larger than the critical value. So, we can reject the null. The HS model perform better.

(h)

1×2 table

DM5	CriticalValue
-0.49177+1.5234e-05i	1.96

Weird output because there is some ES > 0 .

Code:

```
%% Exercise 2
% data input: AAPL from 1/1/2008 - 12/31/2018
AAPL = csvread('AAPL.csv', 1, 1);
AAPL = AAPL(:, 5);
AAPL = 100*price2ret(AAPL);

% (a) unconditional VaR and ES
VaR_un_1 = quantile(AAPL,.01);
ES_un_1 = mean(AAPL(AAPL < VaR_un_1));

VaR_un_5 = quantile(AAPL,.05);
ES_un_5 = mean(AAPL(AAPL < VaR_un_5));

VaR_un_10 = quantile(AAPL,.10);
ES_un_10 = mean(AAPL(AAPL < VaR_un_10));

Summary_VaR_un = table(VaR_un_1,VaR_un_5,VaR_un_10)
Summary_ES_un = table(ES_un_1,ES_un_5,ES_un_10)

% (b) historical simulation
testwindow = 254:2768; %start at 1/2/2009
windowsize = 250;
pVaR = 0.01;
historical1 = zeros(length(testwindow),1);
for t = testwindow
    i = t - 254 + 1;
    estimationwindow = t-windowsize:t-1;
    X = AAPL(estimationwindow);
    historical1(i) = -quantile(X,pVaR);
    es1(i) = mean(X(X < historical1(i)));
end

% (c) filtered historical simulation
sd_old = sqrt(var(AAPL)); %old volatility

results = nwest(AAPL, ones(length(AAPL),1)); %use GARCH model to get new
volatility
resid = AAPL - results.yhat;
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

sd_new = sqrt(sigmasqr); %obtain new volatility
sd_new = sd_new(1:2768,:);
AAPL_filtered = AAPL.*sd_new/sd_old; %obtain filtered return

historical2 = zeros(length(testwindow),1); %do historical simulation
for t = testwindow
    i = t - 254 + 1;
    estimationwindow = t-windowsize:t-1;
    X = AAPL_filtered(estimationwindow);
    historical2(i) = -quantile(X,pVaR);
    es2(i) = mean(X(X < historical2(i)));
end

% (d) plot
plot(historical1)
hold on
plot(es1)
plot(historical2)
plot(es2)
hold off
title('VaR and ES Estimation Using the HS Method and FHS Method')

% (e)
% test HS method
hit_hs = AAPL(254:2768);
for i = 1:2515
if hit_hs(i) <= historical1(i)
    hit_hs(i) = 1;
else
    hit_hs(i) = 0;
end
end

Intercept = ones(size(hit_hs(1:end-1), 1), 1);
X = [Intercept hit_hs(1:end-1) historical1(2:end)];
Y = hit_hs(2:end);
results = nwest(Y,X);
beta7 = results.beta
se7 = results.se;
vcv7 = results.vcv;
R7 = [1 0 0
      0 1 0
      0 0 1];
%Calculate test statistics
TestValue7 = (R7*beta7 - [0.01; 0; 0])'*((R7'*vcv7*R7)\(R7*beta7 - [0.01; 0;
0]))
```

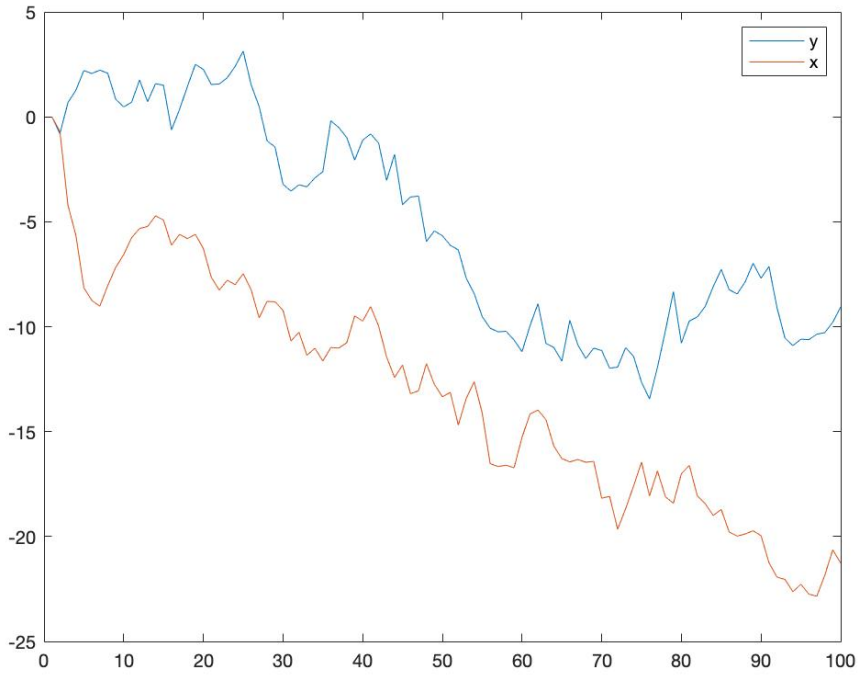
```
%Calculate Critical Value for 5% significance level
CriticalValue7 = chi2inv(1-0.05,2)
%Test if test statistics is less than the Critical Value
TestValue7 < CriticalValue7 %reject

%test FHS method
hit_fhs = AAPL(254:2768);
for i = 1:2515
if hit_fhs(i) <= historical2(i)
    hit_fhs(i) = 1;
else
    hit_fhs(i) = 0;
end
end

Intercept = ones(size(hit_fhs(1:end-1), 1), 1);
X = [Intercept hit_fhs(1:end-1) historical2(2:end)];
Y = hit_fhs(2:end);
results = nwest(Y,X);
beta8 = results.beta
se8 = results.se;
vcv8 = results.vcv;
R8 = [1 0 0
      0 1 0
      0 0 1];
%Calculate test statistics
TestValue8 = (R8*beta8 - [0.01; 0; 0])'*((R8'*vcv8*R8)\(R8*beta8 - [0.01; 0;
0]))
%Calculate Critical Value for 5% significance level
CriticalValue8 = chi2inv(1-0.05,2)
%Test if test statistics is less than the Critical Value
TestValue8 < CriticalValue8 %reject

% (f)
for i = 1:2515
a = AAPL(254:2768).*hit_hs;
es_proxy(i) = 100*mean(AAPL(254:254+i-1));
end
es_proxy = es_proxy';

es1 = es1';
es2 = es2';

%test HS method
Intercept = ones(size(hit_hs(1:end-1), 1), 1);
X = [Intercept es1(2:end) hit_hs(1:end-1)];
Y = es_proxy(2:end);
results = nwest(Y,X);
beta9 = results.beta
se9 = results.se;
vcv9 = results.vcv;
R9 = [1 0 0
      0 1 0
      0 0 1];
%Calculate test statistics
TestValue9 = (R9*beta9 - [0; 1; 0])'*((R9'*vcv9*R9)\(R9*beta9 - [0; 1; 0]))
%Calculate Critical Value for 5% significance level
CriticalValue9 = chi2inv(1-0.05,2)
%Test if test statistics is less than the Critical Value
```

```
TestValue9 < CriticalValue9 %reject

%test FHS method
Intercept = ones(size(hit_fhs(1:end-1), 1), 1);
X = [Intercept es2(2:end) hit_fhs(1:end-1)];
Y = es_proxy(2:end);
results = nwest(Y,X);
beta10 = results.beta
se10 = results.se;
vcv10 = results.vcv;
R10 = [1 0 0
       0 1 0
       0 0 1];
%Calculate test statistics
TestValue10 = (R10*beta10 - [0; 1; 0])'*((R10'*vcv10*R10)\(R10*beta10 - [0;
1; 0]))
%Calculate Critical Value for 5% significance level
CriticalValue10 = chi2inv(1-0.05,2)
%Test if test statistics is less than the Critical Value
TestValue10 < CriticalValue10 %reject

% (g)
loss_hs = (hit_hs-0.01).*(historical1-AAPL(254:2768));
loss_fhs = (hit_fhs-0.01).*(historical2-AAPL(254:2768));
dbar4 = mean(loss_hs-loss_fhs);
dvar4 = var(loss_hs-loss_fhs);
dbarvar4 = dvar4/2515;
DM4 = dbar4/sqrt(dbarvar4);
CriticalValue = 1.96;
Summary4 = table(DM4, CriticalValue)

% (h)
loss_hs2 = ((hit_hs.*(AAPL(254:2768) - historical1))./(0.01*es1)) - ((es1-
historical1)./es1) + log(-es1);
loss_fhs2 = ((hit_fhs.*(AAPL(254:2768) - historical2))./(0.01*es2)) - ((es2-
historical2)./es2) + log(-es2);
dbar5 = mean(loss_hs2-loss_fhs2);
dvar5 = var(loss_hs2-loss_fhs2);
dbarvar5 = dvar5/2515;
DM5 = dbar5/sqrt(dbarvar5);
CriticalValue = 1.96;
Summary5 = table(DM5, CriticalValue)
```

Exercise 2

(b)

```
% (c)
x = zeros(100,1);
y = zeros(100,1);
Tstatistics =zeros(1000,1);
for j = 1:1000
    for i = 2:100
        epsx = randn(100,1);
        x(i) = x(i-1) + epsx(i);
        epsy = randn(100,1);
        y(i) = y(i-1) + epsy(i);
    end
    results = ols(y, [ones(length(x),1) x]);
    Tstatistics(j) = results.tstat(2);
end

plot(y)
hold on
plot(x)
hold off
```



(c)

	StandardNormal	sim
Pr[tstat<-2.58]	0.005	0.353
Pr[tstat<-1.96]	0.025	0.385
Pr[tstat<-1.65]	0.05	0.409
Pr[tstat<0]	0.5	0.513
Pr[tstat<1.65]	0.05	0.393
Pr[tstat<1.96]	0.025	0.368
Pr[tstat<2.58]	0.005	0.34

Code:

```
%% Exercise 3
% (a)
x = zeros(100,1);
y = zeros(100,1);
for i = 2:100
    epsx = randn(100,1);
    x(i) = x(i-1) + epsx(i);
    epsy = randn(100,1);
    y(i) = y(i-1) + epsy(i);
end

% (b)
results = ols(y, [ones(length(x),1) x]);
Tstatistics_default = results.tstat(2);

% (c)
x = zeros(100,1);
y = zeros(100,1);
Tstatistics =zeros(1000,1);
for j = 1:1000
    for i = 2:100
        epsx = randn(100,1);
        x(i) = x(i-1) + epsx(i);
        epsy = randn(100,1);
        y(i) = y(i-1) + epsy(i);
    end
    results = ols(y, [ones(length(x),1) x]);
    Tstatistics(j) = results.tstat(2);
end

plot(y)
hold on
plot(x)
hold off

% (d)
sim = zeros(7,1);
t1 = Tstatistics < -2.58;
sim(1) = sum(t1)/1000;
t2 = Tstatistics < -1.96;
sim(2) = sum(t2)/1000;
t3 = Tstatistics < -1.65;
sim(3) = sum(t3)/1000;
t4 = Tstatistics < 0;
sim(4) = sum(t4)/1000;
t5 = Tstatistics > 1.65;
sim(5) = sum(t5)/1000;
t6 = Tstatistics > 1.96;
sim(6) = sum(t6)/1000;
t7 = Tstatistics > 2.58;
sim(7) = sum(t7)/1000;
StandardNormal = [0.005; 0.025; 0.05; 0.5; 0.05; 0.025; 0.005];
table = table(StandardNormal,sim);
table.Properties.RowNames = {'Pr[tstat<-2.58]'; 'Pr[tstat<-1.96]';
'Pr[tstat<-1.65]'; 'Pr[tstat<0]'; 'Pr[tstat<1.65]'; 'Pr[tstat<1.96]';
'Pr[tstat<2.58]'}
```