%=========================================================================

%

% Program to compute the statistical properties of daily equity returns

%

%=========================================================================

function garch\_statistic( )

clear all

clc

% Load data

load equity

% Choose equity index and compute percentage return

equity = ftse; % ftse, dow, nikkei

% this is just daily percentage reuturn of y...

y = 100\*(trimr( log( equity ),1,0 ) - trimr( log( equity ),0,1 ));

% Compute the autocorrelation of returns and returns squared

% First lets make the arrays with proper sdimension with zeros....than

% we are to replace the array values with proper values...

mlag = 20;

acfy = zeros( mlag+1,1 );

acfy2 = zeros( mlag+1,1 );

acfy(1) = acfunction( y,0 );

acfy2(1) = acfunction( y.^2,0 );

% this following part is created for the purpose of comparison.

for j = 2:mlag

acfy(j) = acfunction( y,j );

acfy2(j) = acfunction( y.^2,j );

end

% Compute the moments of the unconditional (empirical) distribution

y\_z = (y - mean(y))/std(y);

disp( ' Skewness ');

disp( mean(y\_z.^3) );

disp( ' Kurtosis ');

disp( mean(y\_z.^4) );

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%\*\* Generate graphs

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Switch off TeX interpreter and clear figure

set(0,'defaulttextinterpreter','none');

figure(1);

clf;

tt = 1:1:length(y);

lags = 0:1:mlag;

subplot(2,2,1)

plot(tt,y,'-k','LineWidth',0.75);

title('(a) Returns');

ylabel('$y\_t$');

xlabel('$t$');

box off;

axis tight;

subplot(2,2,2)

plot(tt,y.^2,'-k','LineWidth',0.75);

title('(b) Squared Returns');

ylabel('$y\_t^2$');

xlabel('$t$');

box off;

axis tight;

subplot(2,2,3)

%plot( lags,acfy,'-k','LineWidth',0.75);

bar( lags,acfy,0.65,'k' )

title('(c) ACF Returns');

ylabel('$\text{acf}(y\_t)$');

xlabel('$p$');

box off;

axis tight;

subplot(2,2,4)

%plot( lags,acfy2,'-k','LineWidth',0.75);

bar( lags,acfy2,0.65,'k' )

title('(d) ACF Squared Returns');

ylabel('$\text{acf}(y\_t^2)$');

xlabel('$p$');

box off;

axis tight;

%laprint(1,'fig-volatility1','options','factory');

% Kernel density graph

figure(2);

clf;

xi = -6:0.1:6;

f = ksdensity( y,xi);

ft = normpdf( xi,0,1 );

plot( xi,f','-k',xi,ft','--k','LineWidth',0.75 );

ylabel('$f(y)$');

xlabel('$y$');

%laprint(2,'fig-volatility2','options','factory');

end

%

%--------------------------- Functions ----------------------------------

%

%-------------------------------------------------------------------------

% Compute the ACF at kag p

%-------------------------------------------------------------------------

function r = acfunction( x,p )

% d is the deviation from mean.

d = x - mean( x );

% This following part is also calculating the growth..and division

% gives us linear adjustment or skewness measure...

r = sum( trimr(d,p,0).\*trimr(d,0,p) ) / sum(d.^2);

end

%=========================================================================

%

% Program to test for and estimate an ARCH(p) model

AN important note: in ARCH(q) has the property that memory in the variance stops at lag(q).In page 762 , the order of the lagged squared returns , q, defines the maximum number of lags(unlike GARCH). So, squaring the returns is also crucial

%

%=========================================================================

function garch\_test( )

clear all;

clc;

% Load data

load equity;

% Choose index

y = ftse; % ftse, dow, nikkei

%Adjusting the index for linear transformation...

y = 100\*(log(trimr(y,1,0)) - log(trimr(y,0,1)));

y=y-mean(y);

% Set ARCH order

p = 2;

% Test for ARCH(p)

[lm,pv]=testarch(y,p);

% Estimate ARCH(p)

ops = optimset('LargeScale','off','Display','off');

start = rand(p+1,1);

% the function fminunc is not different here...we r just putting start

% command..

[theta,lf] = fminunc( @(b) neglog(b,y,p),start,ops);

theta = abs(theta);

disp('ARCH Estimation and Testing');

disp('---------------------------')

disp(['ARCH order = ',num2str(p) ]);

disp(['alpha\_0 = ',num2str(theta(1)) ]);

disp(['alpha\_1 = ',num2str(theta(2)) ]);

disp(['Likelihood function = ',num2str(-lf) ]);

disp(' ');

disp(['LM test = ',num2str(lm) ]);

disp(['p-value = ',num2str(pv) ]);

end

%

%--------------------------- Functions -----------------------------------

%

%-------------------------------------------------------------------------

% Test for ARCH(maxlag)

%-------------------------------------------------------------------------

function [ lm,pv ] = testarch(data,maxlag)

% Create lags of y

%the lagmatrix is creating a matrix of upto maximum lags which is 2

%here and starting from 1 (not 0). So, only the yt-1 and yt-1 is

%mapped on the data. simple.

tmp = lagmatrix(data,1:maxlag);

% 'any' looks for non-zero elements in column or rows. any(A,1)

% means it will look for numbers of non-zero elements in each column

% and will give the result in a row. any(A,2) means it will look for

% numbers of non-zero elements in each row and will return that number

% in a column. 'isnan' simply looks for if any NaN value is there ,

% which certainly is there in the tmp file. so here, it lokks for if

% there is any NaN value in tmp primarily, generated due to lagmatrix,

% and than any looks for non-zero numbers in all the rows and create a

% matrix.[ ] Brackets are used in forming vectors and matrices or to

% concatenate them. A =[] stores an empty matrix in A

tmp(any(isnan(tmp),2),:) = [];

% Now we are moving towards estimating maximum likelihood estimator of

% e, book page 768, equation 20.21. For the need of e, we arfe squaring

% y

y = trimr(data,maxlag,0).^2;

% as mmentioned in page 769, in the steps, first we need to estimate

% the regression model by least swuares and than we need to compute

% TR^2. here R^2 is simple regression.

% so, for regression model we compute y = bx +e and so e = y - bx. in X

% we always compute the first column to be one, and than we simply

% square the original file tmp, to be in consistency with y.

x = [ ones(length(tmp),1) tmp.^2 ];

%ordinary least square estimator of b (just search RSS in wiki), is

%xt\*y/xt\*x...and so we remove xt to get b = y/x or right array division

% b = x\y

b = x\y;

e = y - x\*b;

t = length(y);

yc = y-mean(y);

% residual sum sum of squares is e^t\*e (just search RSS in wiki)

rss = e'\*e;

% search for tss in wiki

tss = yc'\*yc;

r2 = 1- rss/tss;

% page 769 of the book, step 2

lm = t\*r2;

%p = chi2cdf(x,v) computes the chi-square cdf at each of the values in x using the corresponding degrees of freedom in v.

pv = 1-chi2cdf(lm,maxlag);

end

%-------------------------------------------------------------------------

% Likelihood function for an ARCH(p) model

%-------------------------------------------------------------------------

function lf = neglog( b,data,maxlag )

b = abs(b);

% Compute u and lags of u

u = data;

tmp = lagmatrix(data,1:maxlag);

tmp(any(isnan(tmp),2),:) = [];

% Compute

% here h= x\*b, this part concatenates the matrix. Remember h depicts conditional variance, book page 773.

h = [ ones(length(tmp),1) tmp.^2 ]\*b;

% in the first rows, we take residulas and multiply with ones to

% maintain dimension upto lags, follows h.

h = [std(u)^2\*ones(maxlag,1) ; h];

% Likelihood function

% lets just replace alphas with hessian.

f = - 0.5\*log( 2\*pi ) - 0.5\*log( h ) - 0.5\*(u./sqrt( h )).^2;

lf = -mean( f );

end

%=========================================================================

%

% Program to simulate a garch model

%

%=========================================================================

function garch\_simulate( )

clear all

clc

% Set random number generator

RandStream.setGlobalStream( RandStream('mt19937ar','seed',12345) );

nobs = 100000;

mlag = 20;

% creating the matrixes with random numbers and ones..

z = randn( nobs+1000,1 );

u = z;

y = z;

h = ones( nobs+1000,1 );

% Choose parameter values

%a0 = 0.1; a1 = 0.7; b1 = 0.2;

%a0 = 0.05; a1 = 0.15; b1 = 0.8;

a0 = 0.05; a1 = 0.05; b1 = 0.9;

%Simple, h is unconditional variance (page 771, eq 20.30) and u is

%mentioned in the question. here starting value v0 will be a draw between

%these h and u.

h(1) = a0/(1-a1-b1);

u(1) = normrnd(0,a0/(1-a1-b1));

% Generate data

for t = 2:nobs+1000

h(t) = a0 + a1\*u(t-1)^2 + b1\*h(t-1); % Conditional variance

%now, z(t) is random number of observations that is multiplies with

%conditional variance square root so that we find disturbance term.

%So the properties of disturbace term is primarily they are coming

%from random numbers and than they are standeerdized with

%conditional variance...and this is the most important use of

%conditional variance as they represent the returns.

u(t) = z(t)\*sqrt( h(t) ); % Disturbance term

y(t) = u(t); % Returns

end

% we are taking the latest values.

y = trimr( y,1000,0 );

% and then y is giving us the distance from central value or average.

% Computing centred equity returns.

y = y - mean(y);

%load test;

% Compute the autocorrelation of returns squared

acfy1 = zeros( mlag+1,1 );

for j = 0:mlag

acfy1(j+1) = acfunction( y.^2,j );

end

% Parameter values second model

a0 = 0.05;

a1 = 0.15;

b1 = 0.80;

% Generate data

for t = 2:nobs+1000

h(t) = a0 + a1\*u(t-1)^2 + b1\*h(t-1); % Conditional variance

u(t) = z(t)\*sqrt( h(t) ); % Disturbance term

y(t) = u(t); % Returns

end

y = trimr( y,1000,0 );

y = y - mean(y);

% Compute the autocorrelation of returns squared

acfy2 = zeros( mlag+1,1 );

for j = 0:mlag

acfy2(j+1) = acfunction( y.^2,j );

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%\*\* Generate graphs

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Switch off TeX interpreter and clear figure

set(0,'defaulttextinterpreter','none');

figure(1);

clf;

lags = 0:1:mlag;

subplot(2,1,1)

plot(lags,[ acfy1 zeros(mlag+1,1) ] ,'-k','LineWidth',0.75);

ylim([ -0.5 1 ] );

xlim( [0 10] );

title('(a)');

ylabel('ACF $y\_{1t}$');

xlabel('$p$');

box off;

%axis tight;

subplot(2,1,2)

plot(lags,[ acfy2 zeros(mlag+1,1) ],'-k','LineWidth',0.75);

ylim( [-0.5 1 ] );

xlim( [0 10] );

title('(b)');

ylabel('ACF $y\_{2t}$');

xlabel('$p$');

box off;

%axis tight;

%laprint(1,'volsim','options','factory');

end

%-------------------------------------------------------------------------

% Subroutine: Compute the ACF at kag p

%-------------------------------------------------------------------------

function r = acfunction( x,p )

d = x - mean( x );

%measure of skewness...measure of variance.

r = sum( trimr(d,p,0).\*trimr(d,0,p) ) / sum(d.^2);

end