|  |
| --- |
| IGIDR |
| NESTING SYMMETRIC AND ASYMMETRIC MODELS: DEVELOPMENT OF COMPUTER MODEL FOR TESTING AND SELECTION |
|  |
|  |
| Vikas Kumar Jha  M.Sc. 2014-16 |
| **27/11/2015** |

|  |
| --- |
|  |

**Introduction**

In his paper ‘All in a family: Nesting symmetric and asymmetric GARCH models’, Hentschel develops a a family of models of generalized autoregressive heteroskedasticity

(GARCH) that encompasses all the popular existing GARCH models. The nesting shows that the models are connected within themselves, and can be arrived by manipulating appropriate parameters. However, as the theory goes, practical solutions for the models are not simple due to non-linearity involved. Hence, iterative procedure is adopted.

This project attempts to develop a coded computer model for the same on Octave platform which can be run and used for testing and selection of appropriate model. It also highlights the problems faced and possible solution in developing a practical model from the theory.

**Description**

Autoregressive Conditional Heteroskedasticity (ARCH) models is class of models which is used to account for error terms which are believed to be a function of previous time periods’ error terms. Usually, the model specifies variance as a weighted sum of squares of previous periods’ error terms. The ARCH models are specified as ARCH (q), where q is the length of ARCH lags.

Gnenralized Autoregressive Conditional Heteroskedasticity (GARCH) are more general class of models which allow for a much more flexible lag structure. While in ARCH models, the error terms follow an AR process, which limits the effect of the lag to the time period specified, sometimes the error structure warrants a much longer lag structure for the conditional variance equation. GARCH models take care of this by specifying the conditional error variance as an ARMA structure of past sample variances and conditional variances.

ARCH models were introduced by Engle (1982) and were generalized by Bollerslev (1986) as GARCH models. Post GARCH models introduction, the models have been modified multiple times to explain the reality more accurately.

Two major empirical irregularities have resulted in the specification of many new models (from Hentchel):

1. The equity returns are strongly asymmetric. Negative returns are followed by larger increases in volatility than equally large positive returns, which is referred to now as ‘leverage effect’ (Black, 1976). An asymmetric GARCH model is required to model this requirement. EGARCH process introduced by Nelson (1991), the quadratic GARCH process of Sentana (1991) and Engle (1990), and the TGARCH model of Zakoian (1991) are among the popular asymmetric GARCH models.
2. Stock market returns are fat tailed i.e. leptokurtosis is high. It is reduced when returns are normalized using time varying variances, although not eliminated to satisfactory level. A standard GARCH model computes the next period’s variance by squaring the current period’s shock. For very large shocks, this produces dramatic increases in variance. Accroding to Friendman and Laibson (1989), these are extraordinary events. Taylor and Schwert propose to capture it using a moving average if lagged absolute residuals. Nelson and Foster make similar argument for GARCH models. However, these models do not capture asymmetry, although they can be modified to capture it.

To capture these two issues, a plethora of models have been proposed. However, these GARCH models do not seem to be linked to each other. Hentshel (1995), in his paper provides a unifying framework in which the models can be viewed and tested.

**Specification of Equations**

A first order Autoregression model can be specified as:

where *t* is vector of values at time ‘*t*’, and *zt-1* is vector of values at time ‘*t-1*’. where et is white noise with var(*et*) = *σ2*. The conditional mean of is , while the unconditional mean is 0. Similarly, the conditional variance of the model based on information available at time ‘t = *t-1*’ is var(*et*). However, the unconditional variance is not the same and is equal to *σ2*/(1-2).

The best forecasts are obtained by obtained by using all the available information. So, it would be beneficial to capture the variation in the variance in the model. The model can be specified by modelling the error in the mean equation as:   
  
 & ,

represents the information set available at time ‘*t*’. , which is a function of past error and shock terms. i.e. and , represents the conditional variance of *et.*

For instance in case of b-garch(Bollerslev):

**Asymmetry Modelling**

The b-GARCH model does not take into account the asymmetry of the modelling in variance equation. However, it can be modelled using absolute value function:

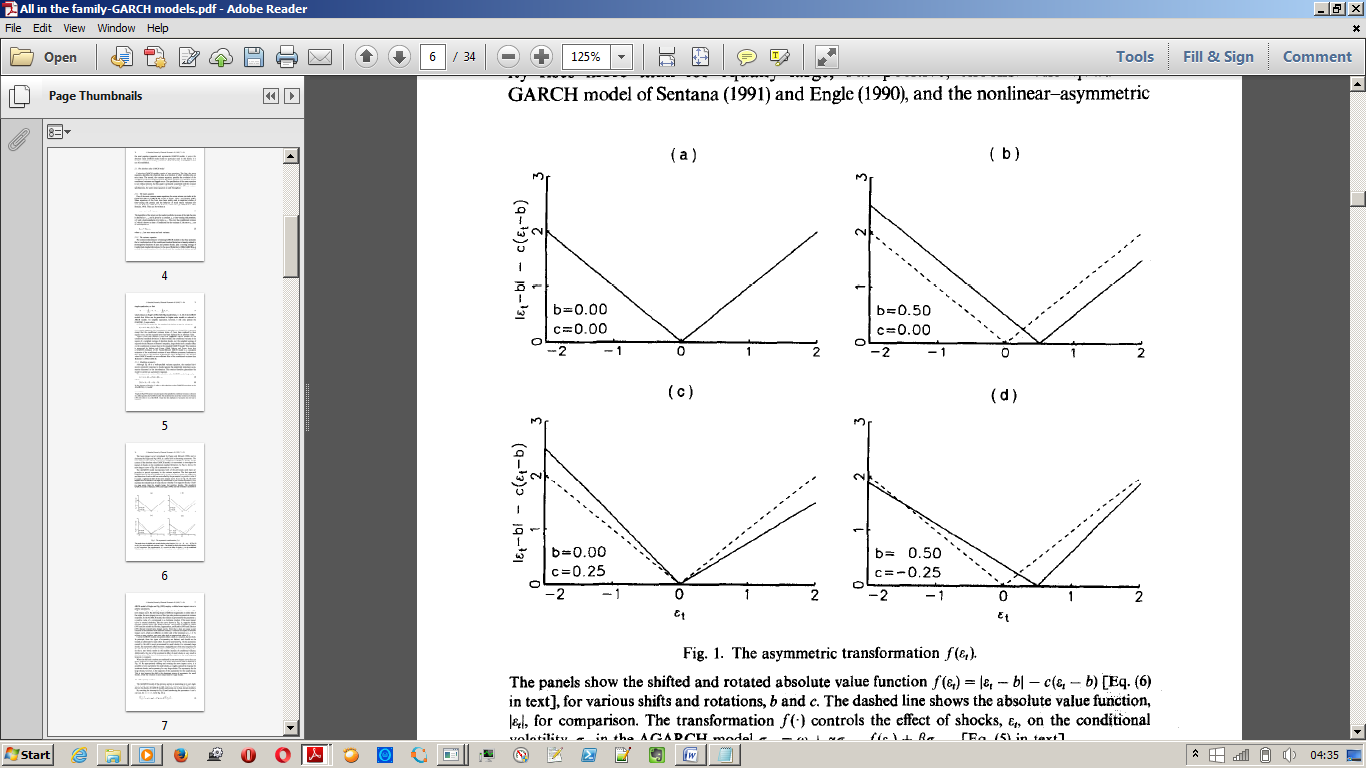
For instance:

where

captures the asymmetry in the conditional variance of the *et*. When the value of the (*υt - b)* is positive the equation reduces to

When the value of the (*υt - b)* is negative the equation reduces to

So, for the same magnitude of difference between b and , the value of will be more in the case when . The asymmetry can be clearly showed using the ‘news impact curve’ introduced by Pagan and Schwert.



x-axis =   
y-axis =

The asymmetry is captured by two approaches:

1. Magnitude of ‘b’ captures the shift and direction of the movement of ‘news impact curve’. The positive value causes shift in the right direction, and vice versa. the news impact curve is shifted to the right by the distance b, one obtains asymmetry that matches the stylized facts of stock return volatility. For negative shocks, volatility rises more than for equally large, but positive, shocks. Eg: ARCH model of Engle and Ng (1993) achieve asymmetry this way.
2. The magnitude of ‘c’ captures the rotation of the news impact curve. By allowing slopes of different magnitudes on either side of the origin, the news impact curves of this type also produce asymmetric variance responses. A positive value of c corresponds to a clockwise rotation. If the news impact curve is rotated clockwise, negative shocks increase volatility more than positive shocks. Eg: EGARCH model of Nelson (1991) and the models by Glosten, Jagannathan and Runkle (1993) use this technique to capture asymmetry. However, c does not cause a pure rotation of the ‘news impact curve’, but it changes the slope of the news impact curve on the two sides.

The presence of both shift and rotation in a model, can lead to either reinforcement or offsetting. By appropriately shifting and rotating the news impact curve, it is possible to have asymmetry for small shocks, a roughly symmetric response for moderate shocks, and asymmetry for very large shocks, or other combination using the knowledge that the rotation causes small asymmetry at low shocks, shift causes equal amount of asymmetry irrespective of shock intensity.

**Nestled Model specification**

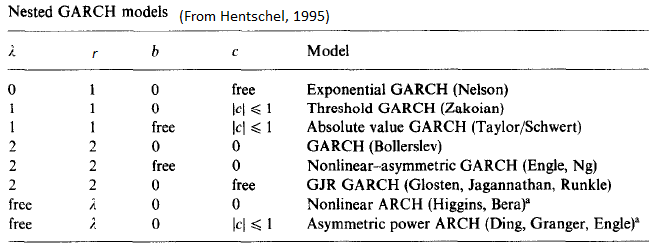
Consider the parent model specification for conditional variance:

According to Henstshel, the conditional variance, the lagged variance and the error term are related as:

This equation nests all popular GARCH models. The special cases are obtained by appropriately choosing the parameters *λ,r,b* and *c.*

Here, *f(υt)* term controls impact of the shocks *υt* on the transformed conditional variance *ht-1*.

For instance, in case of b-GARCH *fr(υt)* = *υt2*, where *r* = 2. The value of ‘*λ*’ modifies the second derivative of the news impact curve. For *λ* > 1 the transformation of *f(υt)* is convex, while for *λ* < 1 it is concave. ‘*r’* serves to transform the (potentially shifted androtated) absolute value function *f(.)*.

****Table 1

**Hyperbolic approximation**

We have :

However, the absolute value function is not differentiable. This obviates the numerical optimization of the likelihood function. For computation of the gradient in the optimization algorithms, the absolute value is approximated as:

**Models used for Octave code**

In this project, three models have been written into codes for Octave program. With simple modifications, the script can be also run in MatLab. The three models are:

1. b-GARCH
2. t-GARCH
3. GJR-GARCH

For all these 3 models, b = 0.

Following are the steps for running the code:

1. The working directory should contain the data filed in .csv format named ‘Data.csv’.
2. Data.csv file should contain the data points in a row format.
3. Run the script ‘TS’ by either typing in command window.
4. The results appear in the command window, giving calculated values of ω, α, β, δ and c, whichever is applicable for the respective model for all the three models, along with the variance and t-values calculated. The calculations are for an AR(1) mean equation and for single period lag for *ht* and *et*.
5. For all the three models, same nested common equation is used. It is converted into respective model by specifying the value of the parameters pertaining to that specific model according to Table 1.
6. Optimization algorithm used is BFGS(Broyden–Fletcher–Goldfarb–Shanno) algorithm, which approximates Newton’s method. It is a type of hill climbing algorithm which seeks for stationary point in a differentiable function.

**Calculations and Results**

The calculation proceeding according to the following steps:

1. The ‘Data.csv’ file is read and stored in variable z as a ‘Tx1’ vector.
2. The data is regressed with a lag on itself and initial value of error for an AR(1) model is generated. The value of *δ* estimated is consistent, but inefficient. This is just to get the initial values for the error values. ‘T’ is the size of the Data i.e. total number of observations.

T = length (z);   
znolag = z(1:T-1,:);  
z1plus = z(2:T,:);  
z2plus = z(3:T,:);  
[d, sigma, e0] = ols (z1plus, znolag);  
e0sq = e0.^2;

1. Other parameters are given some random initial values. The conditional variance is generated using the random number generator.

h0 = 10\*rand(length(z)-1,1);

1. The e0 (error vector) and h0 obtained is used to calculate the initial values of (called eta in the script).

eta = sqrt(e0sq./h0);

1. Initial *ht-1* is generated by removing the last value from h0 vector.
2. The conditional observations in *et* are conditionally normally distributed, and joint density is the product of all the conditional densities, and therefore the likelihood is the sum of conditional normal likelihoods corresponding to the given process. So, joint log likelihood can be written as:

which has to be maximized with respect to ω, α, β, δ and c. It can be written after removing the negative sign:

One cannot define or write a function in Octave in the way it can be defined in specific Statistical or Time-Series packages. Hence, a ‘1x(T-1)’ matrix is defined and is premultiplied with the vector of log likelihoods. It essentially sums up the log likelihoods written for the expression to be used for minimization algorithm. For instance, for b-GARCH model.

bgarch = @(x)((ones(1,length(hT\_1)))\*(log(2\*x(1) + 2\*x(2)\*hT\_1.\*eta.^2 + x(3)\*hT\_1) + ((z1plus - x(4)\*znolag).^2)./(2\*x(1) + 2\*x(2)\*hT\_1.\*eta.^2 + x(3)\*hT\_1)));

1. Here x is the vector of ω, α, β, δ and c. Initial values of the x is generated using random number generator.
2. The ‘fminunc’ function is called for minimization. It is a standard optimization available in the Octave installation. It returns the x(ω, α, β, δ, c) vector, and the hessian matrix, from which the variance and t-values are calculated.

A sample of the results for fitting a Data file is as follows. The size of the sample was 302 observations.

omega\_bgarch = 0.52552  
alpha\_bgarch = 0.53674  
beta\_bgarch = -0.082153  
delta\_bgarch = 1.0010

var\_bgarch =  
9189.7044 2.6337 1878.0966 75.5799

t\_val\_bgarch =  
 5.7186e-005 2.0380e-001 -4.3742e-005 1.3245e-002

omega\_tgarch = 0.023644  
alpha\_tgarch = -330.37  
beta\_tgarch = -0.0028593  
delta\_tgarch = 1.0011  
c\_tgarch = 0.49210

var\_tgarch =  
 2.0937e-007 4.9339e+002 1.6229e-009 2.3172e-009 1.0000e+000

t\_val\_tgarch =  
 1.1293e+005 -6.6959e-001 -1.7619e+006 4.3203e+008 4.9210e-001

omega\_GJR\_garch = 0.91637  
alpha\_GJR\_garch = 0.73993  
beta\_GJR\_garch = -3.2991e-004  
delta\_GJR\_garch = 0.94465  
c\_GJR\_garch = 0.56565

var\_GJR\_garch =  
 1.0000e+000 1.0002e+000 7.5920e-005 1.0028e+000 1.0000e+000

t\_val\_GJR\_garch =  
 9.1637e+000 7.3978e-001 -4.3454e+000 9.4201e-001 5.6565e-001

Data is based on the return of the BSE sensex, for which δ is expected to be near 1, which is so for all the 3 models. However, the variance values for all the 3 models seem to be too high sometimes. The credibility of these variance values for the parameters is in question. Moreover, reruns of the script frequently gives other values for parameters, which makes the result of the program not reliable and prone with errors.

However, the result for the t-GARCH model is much consistent even after repeated runs. The error in the results could be due to following issues:

1. The optimization function is not able to reach the global minima (negative of maxima), and is struck at local minima. To obviate the problem, each iteration was run with different initial value as the initial values are generated by random number generator function, making sure the selection of initial point is random.
2. The iteration is struck at some plateau and is not able to get out of it. For this issue, the program was instructed to carry a fresh iteration if the sum of absolute value of errors was greater than 1, with new initial values. However, the objective function being a huge expression, which may be having a lot number of minima, causing the optimization to get stuck in some local minima again.
3. The optimization function is not able to be handle huge data sets. The ‘fminunc’ function during diagnosis sometimes return vector with all values as ‘NaN’(Not a number), which leads to hessian matrix forming with only ‘1’(s).
4. The Hessian calculation is not being calculated properly. As ‘fminunc’ is an approximation of the Newton’s method, and for the sake of calculation, the approximation may be causing the program to give an erroneous Hessian (a noninvertible Hessian). Another algorithm called BHHH can be used. However, the algorithm is not available for MatLab or Octave openly. The code for the same might be needed to be compiled before it can be used.

**Conclusion**

The code written for running and testing the Nested models and making the appropriate choice is unfortunately unsound. It does solve and provide values for parameters according to the results obtained from the optimization. However, the variances obtained are not reliable. The diagnosis of the exact problem and corrective action is underway.

**Appendix**

The computer code for the Octave program is as follows:

#######################################################################################

clear all

z = csvread("Data.csv");

%prompt = ['p:'];

%dlg\_title = 'Please specify the lags i.e. p';

%defaultans = {'1'};

%p = str2double(cell2mat(inputdlg(prompt,dlg\_title,1,defaultans)));

%prompt = ['q:'];

%dlg\_title = 'Please specify the lags i.e. q';

%defaultans = {'1'};

%q = str2double(cell2mat(inputdlg(prompt,dlg\_title,1,defaultans)));

%Get initial values of stochastic errors using OLS

T = length (z); %Use OLS to get the inefficent, but consistent estimates

znolag = z(1:T-1,:);

z1plus = z(2:T,:);

z2plus = z(3:T,:);

[d, sigma, e0] = ols (z1plus, znolag);

e0sq = e0.^2;

omega = alpha = beta = 0.5;

almostzero = 0.000000001;

%h0 = ones(length(z)-1,1)./4;

h0 = 10\*rand(length(z)-1,1);

eta = sqrt(e0sq./h0);

h = h0;

########################################################################################################################

%%%%for b garch model

%update the value of h according to bgarch formula

hT\_1=h(1:(length(h)),:);

sumofdiff = 0;

while abs(sum(eta)) > 1

%minimise using fminunc; x(1) = omega; x(2)=alpha; x(3)=beta; x(4) = d

hT\_1;

eta(1);

bgarch = @(x)((ones(1,length(hT\_1)))\*(log(2\*x(1) + 2\*x(2)\*hT\_1.\*eta.^2 + x(3)\*hT\_1) + ((z1plus - x(4)\*znolag).^2)./(2\*x(1) + 2\*x(2)\*hT\_1.\*eta.^2 + x(3)\*hT\_1)));

x0 = rand(1,4);

[x,fval,info, output, grad, hess] = fminunc(bgarch,x0);

hT\_1 = [2\*x(1) + (2\*x(2)\*hT\_1.\*eta.^2 + x(3)\*hT\_1)];

eta = (z1plus-x(4)\*znolag)./hT\_1;

%sumofdiff = sum(abs(hT\_1 - h));

endwhile

omega\_bgarch = x(1)

alpha\_bgarch = x(2)

beta\_bgarch = x(3)

delta\_bgarch = x(4)

var\_bgarch = (diag(hess^(-1)))'

t\_val\_bgarch = x./var\_bgarch

%bgarch model finished

###########################################################################################################################

%%%%for t garch model

%update the value of h according to tgarch formula

hT\_1=h(1:(length(h)),:);

lambda = 1;

% hT\_1 = (x(1) + x(2)\*hT\_1.\*f(eta) + x(3)\*hT\_1.^lambda).^(2/lambda)

% f(eta) = ((eta.^2 + almostzero^2).^0.5 - x(5)\*eta)

% hT\_1 = (x(1) + x(2)\*hT\_1.\*((eta.^2 + almostzero^2).^0.5 - x(4)) + x(3)\*hT\_1.^lambda).^(2/lambda)

sumofdiff = 0;

eta = sqrt(e0sq./h0);

while abs(sum(eta)) > 1

%minimise using fminunc; x(1) = omega; x(2)=alpha; x(3)=beta; x(4) = d; x(5) = c

hT\_1;

eta(1);

tgarch = @(x)((ones(1,length(hT\_1)))\*(log((x(1) + x(2)\*hT\_1.\*((eta.^2 + almostzero^2).^0.5 - x(4)\*eta) + x(3)\*hT\_1.^lambda).^(2/lambda)) + ((z1plus - x(4)\*znolag).^2)./((x(1) + x(2)\*hT\_1.\*((eta.^2 + almostzero^2).^0.5 - x(4)\*eta) + x(3)\*hT\_1.^lambda).^(2/lambda))));

x0 = rand(1,5);

[x,fval,info, output, grad, hess] = fminunc(tgarch,x0);

hT\_1 = [(x(1) + x(2)\*hT\_1.\*((eta.^2 + almostzero^2).^0.5 - x(5)\*eta) + x(3)\*hT\_1).^(2)];

eta = (z1plus-x(4)\*znolag)./hT\_1;

%sumofdiff = sum(abs(hT\_1 - h));

endwhile

omega\_tgarch = x(1)

alpha\_tgarch = x(2)

beta\_tgarch = x(3)

delta\_tgarch = x(4)

c\_tgarch = x(5)

var\_tgarch = (diag(hess^(-1)))'

t\_val\_tgarch = x./var\_tgarch

%tgarch model finished

#######################################################################################################

%%%%for GJR garch model

%update the value of h according to GJRgarch formula

hT\_1=h(1:(length(h)),:);

lambda = 2;

% hT\_1 = (x(1) + x(2)\*hT\_1.\*f(eta) + x(3)\*hT\_1.^lambda).^(2/lambda)

% f(eta) = ((1+x(5)^2)\*(eta.^2) - x(5)\*eta.\*(eta.^2 + almostzero^2).^0.5)

% hT\_1 = (x(1) + x(2)\*hT\_1.\*((1+x(5)^2)\*(eta.^2) - x(5)\*eta.\*(eta.^2 + almostzero^2).^0.5) + x(3)\*hT\_1.^lambda).^(2/lambda)

sumofdiff = 0;

eta = sqrt(e0sq./h0);

while abs(sum(eta)) > 1

%minimise using fminunc; x(1) = omega; x(2)=alpha; x(3)=beta; x(4) = d; x(5) = c

hT\_1;

eta(1);

GJRgarch = @(x)((ones(1,length(hT\_1)))\*(log((x(1) + x(2)\*hT\_1.\*((1+x(5)^2)\*(eta.^2) - x(5)\*eta.\*(eta.^2 + almostzero^2).^0.5) + x(3)\*hT\_1.^lambda).^(2/lambda)) + ((z1plus - x(4)\*znolag).^2)./((x(1) + x(2)\*hT\_1.\*((1+x(5)^2)\*(eta.^2) - x(5)\*eta.\*(eta.^2 + almostzero^2).^0.5) + x(3)\*hT\_1.^lambda).^(2/lambda))));

x0 = rand(1,5);

[x,fval,info, output, grad, hess] = fminunc(GJRgarch,x0);

hT\_1 = [(x(1) + x(2)\*hT\_1.\*((1+x(5)^2)\*(eta.^2) - x(5)\*eta.\*(eta.^2 + almostzero^2).^0.5) + x(3)\*hT\_1.^lambda).^(2/lambda)];

eta = (z1plus-x(4)\*znolag)./hT\_1;

%sumofdiff = sum(abs(hT\_1 - h));

endwhile

omega\_GJR\_garch = x(1)

alpha\_GJR\_garch = x(2)

beta\_GJR\_garch = x(3)

delta\_GJR\_garch = x(4)

c\_GJR\_garch = x(5)

var\_GJR\_garch = (diag(hess^(-1)))'

t\_val\_GJR\_garch = x./var\_GJR\_garch

%GJR\_garch model finished

#######################################################################################