STAT440 - Final Project

Alvin Chow and Jerry Hu

April 15, 2018

Abstract:

Financial markets are very volatile in modern days and it is difficult to determine the return value of an asset. An example to tackle the problem is by using ARCH models - Autoregressive conditional Heteroskedasticity. The GARCH model (Generalized Autoregressive Conditional Heteroskedascity) is when the error variance in a time series follows an ARMA (Autoregressive moving average model) model. These models are univariate, which models an asset as a time, which another model (FF-GARCH) explores modelling multiple assets at the same time. This can further explain variance between each asset and between each sector.

More specifically, the parameters $\alpha 0$, $\alpha 1$, $\alpha 2$, $\beta 1$, $beta2$ will be explored more in depth in the GARCH(1,1) and GARCH(2,2) models. r_out will be also compared between our univariate models and the FF-GARCH model. Posterior and Prior distributions will be created for r_out variable (prediction variable for T_out time periods), and we will show how the models converge to the posterior distribution.

The objective of our implementation of the following STAN model was to estimate certain parameters, the sum of which could hopfully describe the process underlying the returns of certain fianncial assets on the S&P500 index with some degree of aaccuracy (or, at least better than a univariate GARCH process could manage). The implicit assumption here is that

In this report, we will explore the dynamics and parameter estimates for the GARCH(1,1), GARCH(2,2), and Multivariate Garch (FF-GARCH) across different sector of stocks using Stan and R.

Stan is a programming language developed in C++ backend and interfaced with R front-end. It is a more efficient and faster way to sample from the prior distribution.

Full-Factor Generalized Autoregressive Conditional Heteroskedasticity (FF-GARCH) model:

The multivariate GARCH model we used was the the Full-Factor Multivariate GARCH, so called because it models N factors (where N is the total number of assets per time period) at each time step, each of which can be defined as a separate univaraite GARCH process. The equations that describe how the model is parameterizedare:

$$y_t = \mu(\theta) + \varepsilon_t$$
$$\varepsilon_t = H^t z_t$$
$$H_t = W \Sigma_t W'$$

where $t \in [1, ...T]$ denotes the index of the particular time interval, y is a vector of dimensions Nx1 denoting the returns of all N stocks at time t, H^t and W are square matrices of size NxN where W is lower triangular with ones along the diagonal and H_t is dense, $\Sigma_t = diag(\sigma^2_{1,t}, \sigma^2_{2,t}, ..., \sigma^2_{N,t})$ and z_t is an Nx1 random vector such that $E[z_t] = 0_N$ and $Var(z_t) = I_N$. For our purposes and for simplicity's sake, we'll be using a basic $z_t \sim N(0, I)$ multivariate normal distribution.

It should be noted that the first two equations are not unique to the FF-MGARCH; only the equation $H^t = W \Sigma_t W'$ uniquely identifies Full-Factor, as various other multivariate GARCH models (e.g. VEC or BEKK) are paramterized very similarily. The key differentiator between them is how they calculate the H_t matrix.

Modelling the FF-GARCH is greatly simplified by the fact that the individual σ^2_{nt} (or 'factors') can be modelled separately and independently of each other, only being constituted into H_t just before it's needed.

Because of this, FF-MGARCH is described as a 'generalization' of the univaraite GARCH models, with the implication being that it's a logical extension of the univariate version. And indeed, at first glance, it seems the FF-MGARCH equation is just a basic LDL' decomposition of a symmetric, positive definite (read: covariance) matrix. Nontheless, as we'll see, FFGARCH is potentially very powerful.

Though any univariate GARCH would suffice for modelling the factors, we settled for GARCH(1,1) for reasons of both simplicity and computational efficiency; looking back beyond one time step would likely prove very costly given that there are essentially N sepatrate models running in parallel.

The final layer of equations that describes FF-GARCH are thus:

$$\sigma^2_{n,t} = \alpha_n + \beta_n \varepsilon^2_{n,t-1} + \gamma_n \sigma^2_{n,t-1}$$

It is here that restrictions upon parameters are introduced. All σ values are retricted to positive positive values as they represent variance terms; that α, β and γ are also necessarily positive is a logical conclusion. Less obvious are two other constraints; un upper bound for β of one and an upper bound for γ of $1 - \beta$. Together, they ensure that the process this factor models is stationary, an important property in the pricing of financial instruments. For various reasons, not all of these restrictions could be sucessfully implemented in the STAN model, a fact that will be discussed in greater detail under the section on the models shortcomings.

Putting it all together, the final model equation is

$$y_t \sim MVN(0, H^{1/2} z_t)$$

Coding Methodology in Stan

The basic idea behind this implementation of FF-GARCH is quite similar to univariate GARCH(1,1); we pass an inital starting state of $\sigma^2_{N,0}$ squared values (in this case, an array of size N) for the model to build additional $\sigma^2_{N,t}$ recursively, with both the previous σ^2 and ε contributing the random component to the following σ^2 values. The coefficients that determine exactly how much influence the random components have against the constant components are the α_t, β_t and γ_t terms. These components are time-invariant and thus are critical in modeling the process as a whole. Part of the objective of our STAN model is to determine these values experimentally via random sampling and Bayesian inference.

The other component that our model attempts to estimate is the W matrix that determines how the σ^2 terms interact with each other. This is an NxN lower diagonal Cholesky factor matrix with ones along the diagonal that determines how the critical H_t matrix will behave once the factors start interacting with each other.In very general terms, the W matrix can be thought of as the parameter providing the covariance between the factor, the variance terms of this MVN GARCH process.

```
data {
  int<lower=0> T;                       // Number of time intervals
  int<lower=2> N;                       // Number of assets
  vector[N] ret[T];                     // Stock return data
  real<lower=0> sigma_init[N];          // Initial values of Sigma squared
}
transformed data{
  int<lower=1> N_choose_2;              // Number of strictly lower-than-diagonal elements of W
  N_choose_2 = (N*(N - 1)) / 2;
}
parameters {
  vector[N] mu;                         // Vector of mean returns
  real<lower=0> alpha[N];               // Alpha parameters for N assets
  real<lower=0,upper=1> beta[N];        // Beta parameters for N assets
  real<lower=0,upper=1> gamma[N];       // Gamma parameters for N assets
  vector[N_choose_2] W_lower;           // Vector of strictly lower-than-diagonal elements of W
}
```

Here, the number of time periods passed into the model T and the number of different stocks N are bother required data inputs into the model as to many other parts depend on these values (a decision was made to impose a minimum lower bound of 2 assets for stability reasons). The returns data was called as an array of T Nx1 vectors for computational efficiency down in the models block.

One noteworthy thing about these parameter declarations is the incorrect upper limit on gamma (specifically, the fact that it's not 1-beta). Unfortunately, STAN does not allow for varing bounds on the elements of arrays (since each alpha corresponds to a gamma); therefore, the <... upper=1> restriction represents an absolute bound on gamma since this only occurs when beta = 0.

Another rather bizarrre declaration is the value N_choose_2 and the vector W_lower of that length, owing again to the pecularities and limits of STANS's system of variable declaration. Needless to say, this situation is not ideal, but again, essentially unavoidable at this point. The use of these values will become clear later on.

```
transformed parameters {
  cholesky_factor_cov[N] W;             // Cholesky Factor Matrix of an NxN covariance matrix
  real<lower=0> sig[N,T];               // N sigma squared values for each time interval of T
  for (n in 1:N) {
    W[n,n] = 1;
  }
  for (i in 1:N_choose_2) {
    for(j in 2:N) {
      for(k in 1:(j-1)) {
        W[j,k] = W_lower[i];
        W[k,j] = 0;
      }
    }
  }
  for (n in 1:N)
    sig[n,1] = sigma_init[n];
  for (t in 2:T) {
    for (n in 1:N) {
      sig[n,t] = alpha[n] + beta[n]*pow((to_array_1d(ret[t-1] - mu)[n]), 2) +
        gamma[n]*sig[n,t-1];
    }
  }
}
```

The Cholesky factor matrix W is declared as a transformed parameter because it needs to be modified; STAN does not allow for the direct definition of a unit-triangular matrix. Therefore, the diagonals have to be defined manually and the below-diagonal non-vero elements paramaterized separately (the W_lower vector) and entered into the matrix using for loops. This is somewhat inefficient; major savings in computationally efficiency could be had if a dedicated unit-diagonal matrix type was accessbile.

The rest of the model is unremarakable; the σ^2 values are obtained recursively using previous σ^2 and ε^2 values (obtained here by subtracting the previous returns from the mu average vector). One thing to note however, is the use of a 2D array to store the time series σ^2; this was not an accident but a design choice.

```
model {
  for (t in 1:T) {
    ret[t] ~ multi_normal_cholesky(mu, W*diag_matrix(sqrt(to_vector(sig[1:N,t]))));
  }
}
generated quantities {
  vector[N] ret_out[T];
  for (t in 1:T) {
    ret_out[t] = multi_normal_cholesky_rng(mu, W*diag_matrix(sqrt(to_vector(sig[1:N,t]))));
  }
}
```

By contrast, ret and ret_out are both arrays of vectors and thus, operate more efficiently due to STAN favouring the vectorization of parameters when compiling. These awkward equations involving subarrays and vectors necessitate a rather cumbersome process, wherby a subarray containing σ values were converted into a vector, called with sqrt() elementwise, then converted into a diagonal matrix and multiplied with W. The reason for this process is rather mundane and is one seriously frustrating thing about STAN.

However, this cumbersome code is at least partially offset by the choice to use multi_normal_cholesky() rather than a standard multi_normal() distribution. The efficiency savings here are hard to overstate, and they are possible due to the pecularities of FF-MGARCH.

Choice of multivariate normal functions

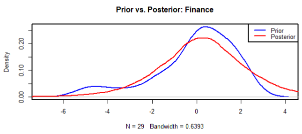
While one method for obtaining the ε_t values (i.e. the "variablilty" of the process at time t) is obvious and simple to figure out from $y_t = \mu(\theta) + \varepsilon_t$ and $\varepsilon_t = H^{1/2} z_t$ (i.e. just calculate $H^{1/2}$ directly and multiply it with a randomly generated variable z_t), this is very much a "brute-force" method. Notably, the cost of obtaining $H^{1/2}$ can be quite expensive, even though H is symmetric and positive definite by construction as calculating $H^{1/2}$ involves taking its Cholesky decomposition which is still an $O(n^3)$ operation. In addition, round-off error can quickly compound and lead to wild inaccuracies since the calculations involve recursing over the same parameters over many factors for each time period.

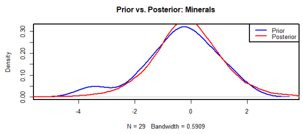
Thus, a chance to avoid calculating $H^{1/2}$ each iteration is highly desirable; this was done by exploiting the nature of the FF-MGARCH H_t matrix. From Bauwens, it was noted that $Var_{t-1}(y_t) = Var_{t-1}(\varepsilon_t) = H_t^{1/2} Var_{t-1}(z_t)(H_t^{1/2})' = H_t$ or in other words, the conditional variances of the process are equal $H_t^{1/2}\Sigma_t H_t^{1/2}$ which when taken toger ther with $H_t = W\Sigma_t W'$ impies that $H_t^2 = W_t$. Furthermore, H_t can be expressen in the form LL' since $H = W\Sigma W' = W\Sigma^{1/2}\Sigma^{1/2}W' = LL'$ since $Sigma^{1/2} = diag(\sigma_1, \sigma_2 ... \sigma_N)$ is still diagonal and thus, $W\Sigma^{1/2}$ is lower triangular. This allows for the use of multi_normal_cholesky(μ, L) where L is the cholesky factor of the variance matrix, a much more efficient approach in this scenario.

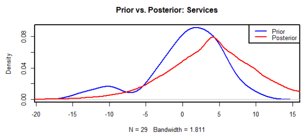
Results

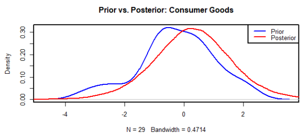
The data set that this FFGARCH model was applied to was the S&P500 stock data provided on the course website Learn. For computational time reasons, the time frame over which calculations took place were limited to the last 30 days. Preprocessing for sector level modelling of stock returns was achieved by first averaging out all the company stock prices for a particular sector (e.g. Financial, Tech, etc. . .) over every one of those last 30 days, then calculating the change in those sector-averaged prices for those 30 days (giving us (T = 29) * (N = 7) = 203 pieces of data to work with).

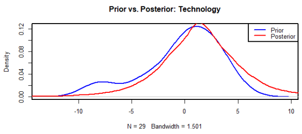
Calling FF-MGARCH on those 7 'assets' over 29 time periods gave us a model that produced the following posterior distribution among predicted values:

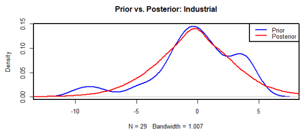


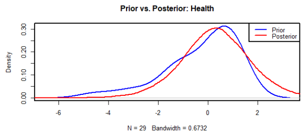




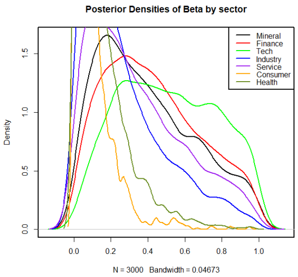


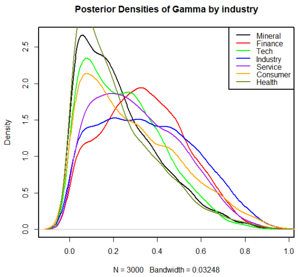


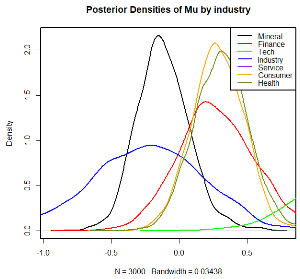


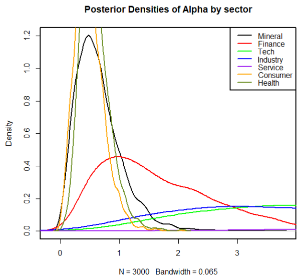


When compared with each other, the posterior density functions for the different parameters show some variablilty, but are otherwise rather well clustered for the most part (with the exception of the alpha parameter)









While the Beta and Gamma parameter graphs seem to share the same geneal shape (suggesting that past volatility seems to have the same effect on present volatility for every company, regardless of sector), the mu and alpha parameters exhibit much more variation. To me, this is a fairly unsuprising result; it suggests that regardless of past instances of volatility in the stock returns, some sector are simply far more volatile than others on a structural level (i.e. base level of sigma squared or alpha is higher). Tech. and to a lesser extent, the finance sector exhibited large right tails in their alpha parameter graph, suggesting a higher liklihood og high volatility trading days (no suprise for these industries); however, industrial sector stocks also exhibited high degrees of "base volatility" which really suprised me". I can not say for certain, but I would speculate that perhaps the events behind the swings in stock returns in industrial companies is political in nature.

For a more detailed description on how FF-GARCH performed on a particular sector, I have included additional plots showing how well the sampler did in the Financial sector in particular. These plots include a plot of the values of alpha,beta,gamma and mu over all the iteration of the algorithm, as well as a pairs plot showing how the sampled data clustered.

Univariate GARCH approach:

Knowing that this is a time series problem, applying univariate GARCH to each individual asset isn't a terrible starting ground. We decided to code the modelling in Rstan, as the sampling is much faster and more accurate when done with C++. First approach is looking at the basic GARCH(1,1) model.

This model only looks at one lagged time term for the error and the standard deviation terms. Intuitively, this seems like a simple and naive approach to modelling the dataset, but it didn't give us bad results. The MCMC sampler explored the sample space appropriately as shown through simple diagnostics such as the trace and autoregressive-correlation plots. The Autoregressive correlation plot shows that the next time interval will have very low correlation with the current time step. The trace plot demonstrates excelent mixing. Due to it's simplistic "look at t-1" approach, this model did not take long to converge an run.

Looking at the GARCH(2,2) model, it took a bit longer to converge and run, because it is a more complicated model. Intuitively this makes sense because it requires more computation to calculate both "t-1" and "t-2" time periods.

To properly diagnose the MCMC, extraction of the alphas and betas were done and plotted with each lagged time term on the x-axis. A good trace plot is one which mixes very well; That is one which fluctuates up and down dramatically. This shows the sampler is exploring new areas of the space and isn't staying in one spot. A good Autoregressive correlation plot is one where the next lagged time term is not correlated at all with the previous lagged term. In both of the GARCH(1,1) and GARCH(2,2) models, it demonstrates that the alphas and betas contain excellent sampling.

RStan code is shown below GARCH(1,1)

```
data {
  int<lower=2> T;
  real r[T];
  real<lower=0> sigma1;
  int<lower=0> T_out;
}
parameters {
  real mu;
  real<lower=0> alpha0;
  real<lower=0,upper=1> alpha1;
  real<lower=0, upper=(1-alpha1)> beta1;
}
transformed parameters {
  real<lower=0> sigma[T];
  sigma[1] = sigma1;
  for (t in 2:T)
    sigma[t] = sqrt(alpha0 +
                      + alpha1 * square(r[t - 1]-mu)
                    + beta1  * square(sigma[t - 1]));
}
model {
  r ~ normal(mu,sigma);
}
generated quantities {
  real r_out[T_out];
  r_out[1] = normal_rng(mu, sigma[1]);
  for (t in 2:T_out) {
    r_out[t] = normal_rng(mu, sigma[t]);
  }
}
```

Sample trace plots and autoregressive lag plots are shown in the appendix.

After diagnosing the individual MCMC sampler, next steps would be to model the interactions between each of the sectors of stocks. These sectors include Financial, Basic Minerals, Tech, Industrial, Services, Healthcare, etc…

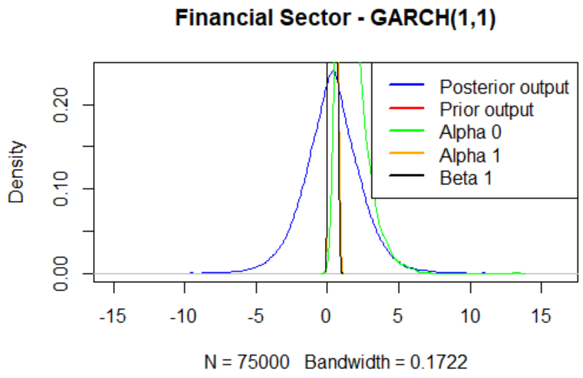
To tackle modelling between sectors, an average across each sector for each time period is computed. Furthermore, GARCH(1,1) and GARCH(2,2) are built on the new "average" dataset for each time period. The following parameter values are shown below when ran with the entire dataset:

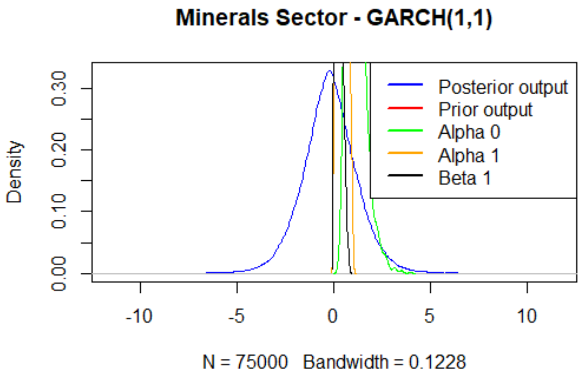
```r
library(rstan)
set.seed(440)
source("GARCH_func_code.R")
df = read.csv("snp500-adj_close_2004-2018.csv")
df = price_to_return(df)

# FINANCIAL SECTOR (Columns 1-7)
fin_param = parameter_extraction(df[,1:7],0.3,0.3,30)
print (fin_param[1:8])
#GARCH11 - (alpha0 = 0.0128, alpha1 = 0.0981, beta1 = 0.899)
#GARCH22 - (alpha0 = 0.0186, alpha1 = 0.0852, alpha2 = 0.0696, beta1 = 0.331, beta2 = 0.511)

# BASIC MINERALS (Columns 8-12)
minerals_param = parameter_extraction(df[,8:12],0.3,0.3,30)
print (minerals_param[1:8])
#GARCH11 - (alpha0 = 0.00452, alpha1 = 0.0639, beta1 = 0.932)
#GARCH22 - (alpha0 = 0.00833, alpha1 = 0.0354, alpha2 = 0.0761, beta1 = 0.2648, beta2 = 0.6161)

# TECHNOLOGY (Columns 13-21)
```

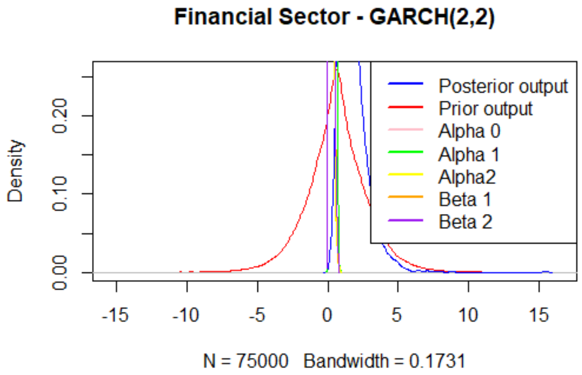
```
tech_param <- parameter_extraction(df[,13:21],0.3,0.3,30)
print (tech_param[1:8])
#GARCH11 - (alpha0 = 0.00529, alpha1 = 0.0649, beta1 = 0.933)
#GARCH22 - (alpha0 = 0.00755, alpha1 = 0.0792, alpha2 = 0.0152, beta1 = 0.3891, beta2 = 0.5144)

# INDUSTRIAL (Columns 22-26)
industry_param = parameter_extraction(df[,22:26],0.3,0.3,30)
print (industry_param[1:8])
#GARCH11 - (alpha0 = 0.00229, alpha1 = 0.0589, beta1 = 0.9396)
#GARCH22 - (alpha0 = 0.00435, alpha1 = 0.0494, alpha2 = 0.0595, beta1 = 0.15689, beta2 = 0.7314)

# SERVICES (Columns 27-33)
services_param = parameter_extraction(df[,27:33],0.3,0.3,30)
print (services_param[1:8])
#GARCH11 - (alpha0 = 0.00055, alpha1 = 0.0255, beta1 = 0.9743)
#GARCH22 - (alpha0 = 0.0008, alpha1 = 0.0319, alpha2 = 0.00914, beta1 = 0.3567, beta2 = 0.6019)

# CONSUMER GOODS (Columns 34-38)
consumer_param = parameter_extraction(df[,34:38],0.3,0.3,30)
print (consumer_param[1:8])
#GARCH11 - (alpha0 = 0.00055, alpha1 = 0.0255, beta1 = 0.97433)
#GARCH22 - (alpha0 = 0.0008, alpha1 = 0.0319, alpha2 = 0.009144, beta1 = 0.35676, beta2 = 0.60191)

# HEALTHCARE (Columns 39-45)
health_param = parameter_extraction(df[,39:45],0.3,0.3,30)
print(health_param[1:8])
#GARCH11 - (alpha0 = 0.0015, alpha1 = 0.1052, beta1 = 0.889)
#GARCH22 - (alpha0 = 0.0028, alpha1 = 0.0802, alpha2 = 0.1143, beta1 = 0.1424, beta2 = 0.6529)
```

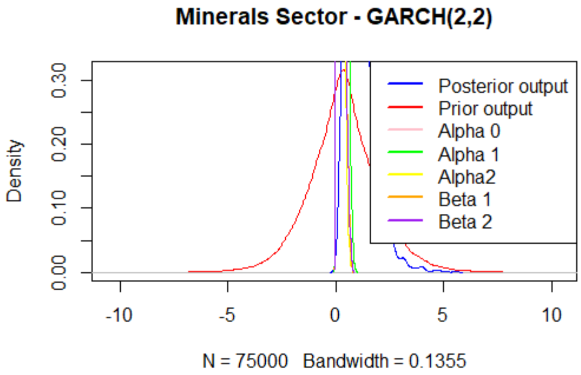
Results: As shown from above for Univariate models, we can see that the parameters for Consumer Goods sector is around the same as the parameters for Services parameters. This means that the returns are very similar in the two sectors. All other parameters are slightly different from each other. To hedge the risk of Consumer goods sector doing poorly, one must choose to NOT invest in the Services sector. If one believes that consumer goods sector is booming, one should also invest in the services sector.In addition, we see that a high parameter value for Betas is present in comparison to alphas. This further concludes that the next day's volatility is highly dependant on the previous day's volatility more so than it's error term.

To compare with the Multi-variate model, we will be comparing the last month of data in the dataset. Code is shown below: (Red graphs shows Posterior, Blue shows prior)





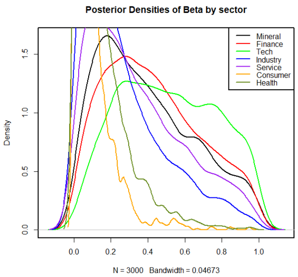


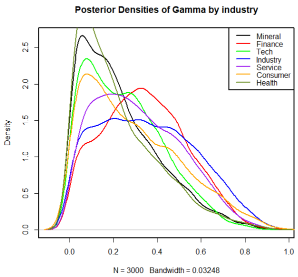


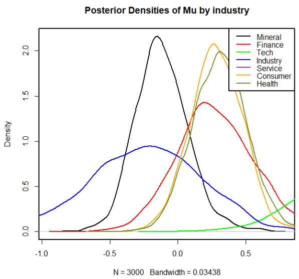
As shown above, the posterior and prior distributions are quite identical. As we obtain more and more samples, it becomes apparent that they both converge to the same distribution. Interesting to note that when the time period of the data input (to the models) is small and not the entire dataset, the betas will become smaller and smaler while the alphas will have a bigger impact to the model.

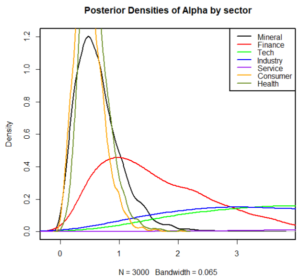
Compared to the Multi-Variate FFGARCH whos results are shown below:

As you can see, the results are quite similar for r_out afor both methods (Univariate and Multivariate). The parameter distributions itself (α_0,β_0, α_1, β_1, α_2, and $\beta 2$) are quite different for the models. Below are graphs showing the posterior probability density distributions obtained when compared to the prior density distributions.









With respect to GARCH(1,1) and GARCH(2,2), as we tested and predicted the next month's data with the models, GARCH(1,1) seems to be performing better than GARCH(2,2) models. As we are predicting more into the future, the accuracy decreases dramatically since today's volatility is dependant on yesterday's volatility. In comparison to other stock models (ARMA models) GARCH performs well under high VIX indices while ARMA performs well under low VIX indices due to heavy changes in the stock markets.

Do any models actually fit the data:

Conclude about dataset you are studying:

Discussion :

summarize contributions. Strengths / limitations of analysis. Future research?

References:

Bayesian Inference Methods for Univariate and Multivariate GARCH Models : A Survey; Audrone Virbickaite, M. Concepcion Ausin, Pedro Galeano https://arxiv.org/pdf/1402.0346.pdf

Comparison results for Garch Processes; Fabio Bellini, Franco Pellerey, Carlo Sgarra, Salimeh, Yasaei Sekeh https://arxiv.org/abs/1204.3786

Continuous-Time Garch Processes; Peter Brockwell, Edrenebaatar Chadraa, Alexander Lindner https://arxiv.org/pdf/math/0607109.pdf

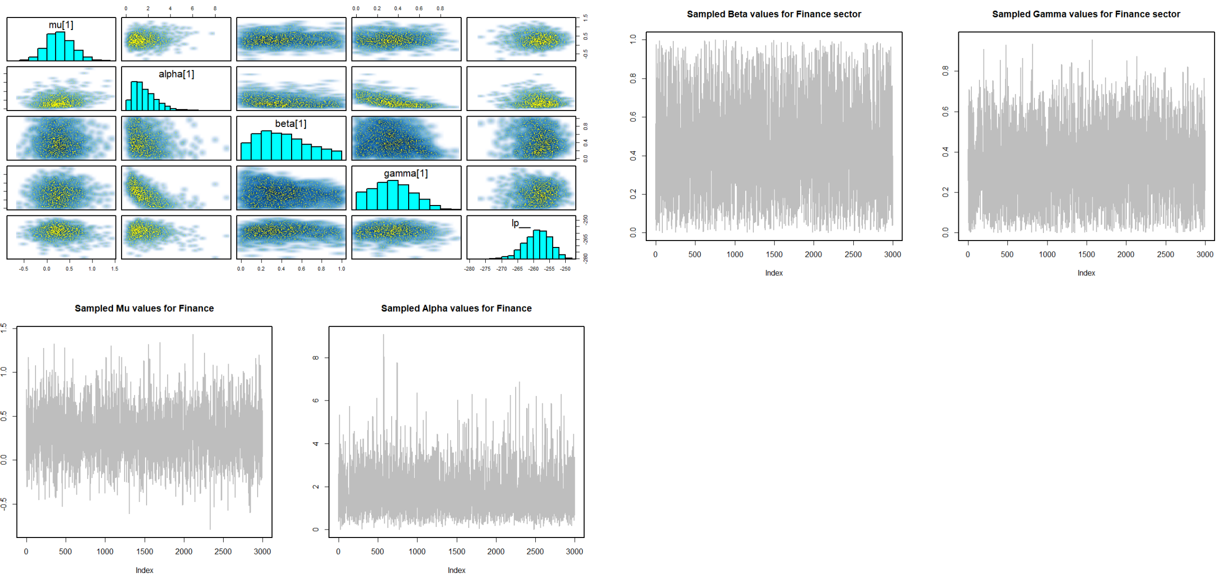
The efficiency of the Estimators of the Parameters in Garch Processes; Istvan Berkes, Lajos Horvath https://arxiv.org/pdf/math/0406432.pdf

MULTIVARIATE GARCH MODELS: A SURVEY - Journal of Applied Econometrics Luc Bauwens, Sebastien Laurent and Jeroen V.K. Rombouts

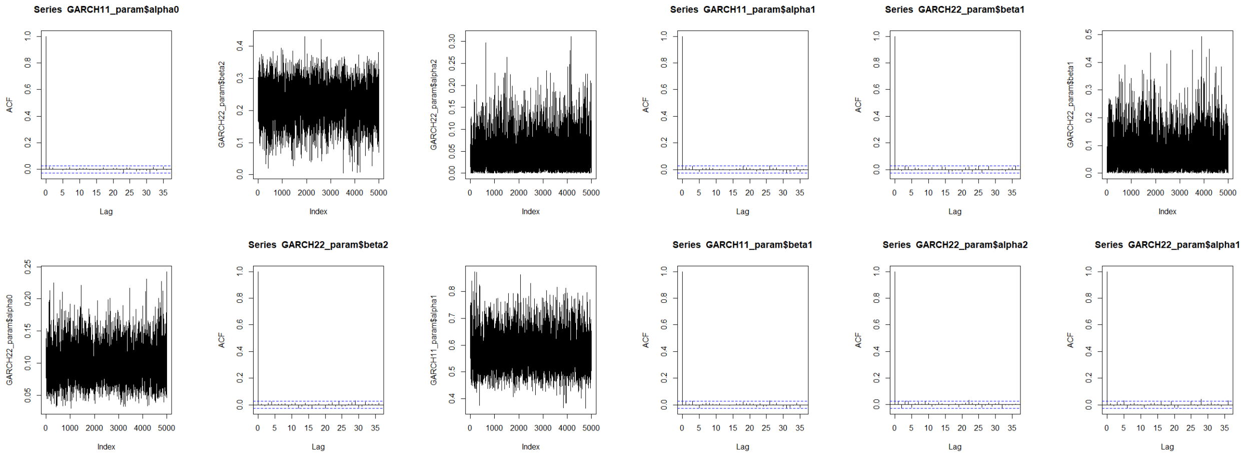
A full-factor multivariate GARCH model I.D. Vrontos, P.Dellaportas and D.N Politis

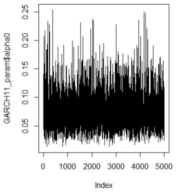
Appendix:

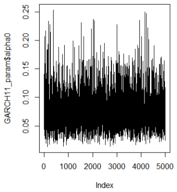
Financial Plots

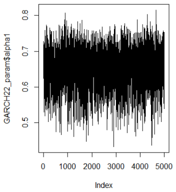


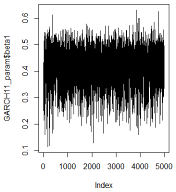
Sample trace plots and autoregressive lag plots are shown below for GARCH models




GARCH11_param$alpha0
0.05
0.10
0.15
0.20
0.25
0
1000
2000
3000
4000
5000
Index


GARCH11_param$alpha0
0.05
0.10
0.15
0.20
0.25
0
1000
2000
3000
4000
5000
Index


GARCH22_param$alpha1
0.5
0.6
0.7
0.8
0
1000
2000
3000
4000
5000
Index


GARCH11_param$beta1
0.1
0.2
0.3
0.4
0.5
0.6
0
1000
2000
3000
4000
5000
Index