ECON1670: Bayesian Analysis of S&P 500 Returns

Will Chen

May 12, 2023

Abstract

The aim of this project is to use Bayesian posterior simulation methods to analyze the returns of the S&P 500 using a Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model (discussed below in the **Model Description** section).

To simulate the model, I will employ a MCMC method, specifically a Metropolis-Hastings random walk, to sample from the posterior. I will be simulating the model parameters of interest using the random walk.

Another component of the project is model checks. I will be employing prior predictive analysis to simulate from the priors for our parameters and demonstrate that they are reasonable priors to use. I will also compare log-scores of the GARCH model vs. a simpler ARCH model. Finally, I will use joint distribution tests to plot the prior and posterior means together to show posterior simulator convergence.

1 Data Description

The dataset used in this project is the S&P 500 returns, which contains the daily percentage movement of the price of the index fund from 1972 to 2005.

The S&P 500 (Standard and Poor's 500), is a stock market index tracking the stock performance of 500 of the largest companies listed on stock exchanges in the United States. The S&P 500 is generally used as an index for the current state of the US equity market.

2 Model Description

For this dataset, I will use a **Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model**, which aims to model the conditional volatility of a time series. For financial data where the variance error/volatility is believed to be serially autocorrelated, it is best to use a model such as GARCH, which assumes that the variance of the error term follows an autoregressive moving average process. In other words, in the GARCH model, the volatility at a current point in the time series depends on both the volatility and value at a previous time series.

Specifically in this project, I will be using a **GARCH(1,1)** model. This means that we will be using the volatility and value of the time series at 1 previous timestep lag to determine the current volatility.

General GARCH model outline

An equation for the GARCH(1,1) is as follows:

$$\sigma_t^2 = \omega + \alpha r_{t-1}^2 + \beta \sigma_{t-1}^2 \quad [2.1]$$

Where:

- σ_t^2 — the volatility being modelled at a particular time t
- ω — a model parameter representing the long-run volatility
- r_t — log returns at a particular time t (in our case, of the SP 500)
- α — model parameter, the weight for the returns lag term.
- β — model parameter, the weight for the volatility lag term.

To estimate the model parameters ω, α, β, we would need to maximize (using MLE) the following equation:

$$-0.5 \sum_{t=1}^{T} \left[\ln(2\pi\sigma^2) + \frac{r_t^2}{\sigma^2} \right] \quad [2.2]$$

In our case, we will be using Bayesian methods to simulate the model posterior, which is described below.

3 MCMC Algorithm

Our MCMC posterior simulation algorithm will be a **Metropolis-Hastings** random walk.

Priors

For α and β, we can use an improper prior that follows the following constraint:

$$\text{Prior}(\alpha, \beta) \propto 1\{\alpha > 0, \beta > 0, \alpha + \beta < 1\} \quad [3.1]$$

For ω, we will use an inverse-gamma distribution under the constraint $\omega > 0$:

$$\text{Prior}(\omega) = \text{Inverse Gamma}(\omega; \alpha_\omega, \beta_\omega) \cdot 1\{\omega > 0\} \quad [3.2]$$

$$\propto \omega^{-\alpha_\omega - 1} e^{-\beta_\omega / \omega} \cdot 1\{\omega > 0\} \quad [3.3]$$

We will use log-priors in our calculations for numerical precision.

Likelihood

We will use the following log-likelihood function:

$$-0.5 \sum_{t=1}^{T} \left[\ln(2\pi\sigma^2) + \frac{r_t^2}{\sigma^2} \right] \quad [3.4]$$

Metropolis-Hastings Random Walk

- Start with an initial state, which can be any value within the parameter space.
- Choose a proposal distribution, which is a probability distribution that specifies how to generate a new candidate state given the current state. In our case, we used a normal distribution.
- Generate a candidate state by drawing a sample from the proposal distribution, centered at the previous sample.
- Compute the acceptance probability, which is the ratio of the posterior probability of the candidate state to the posterior probability of the current state.
- Accept or reject the candidate state based on the acceptance probability.
- In the first 10% of iterations, adjust the covariance matrix used for random draws by calculating the covariance of all draws up to current.
- Adjust/scale covariance if acceptance rates are too high/too low.

Results

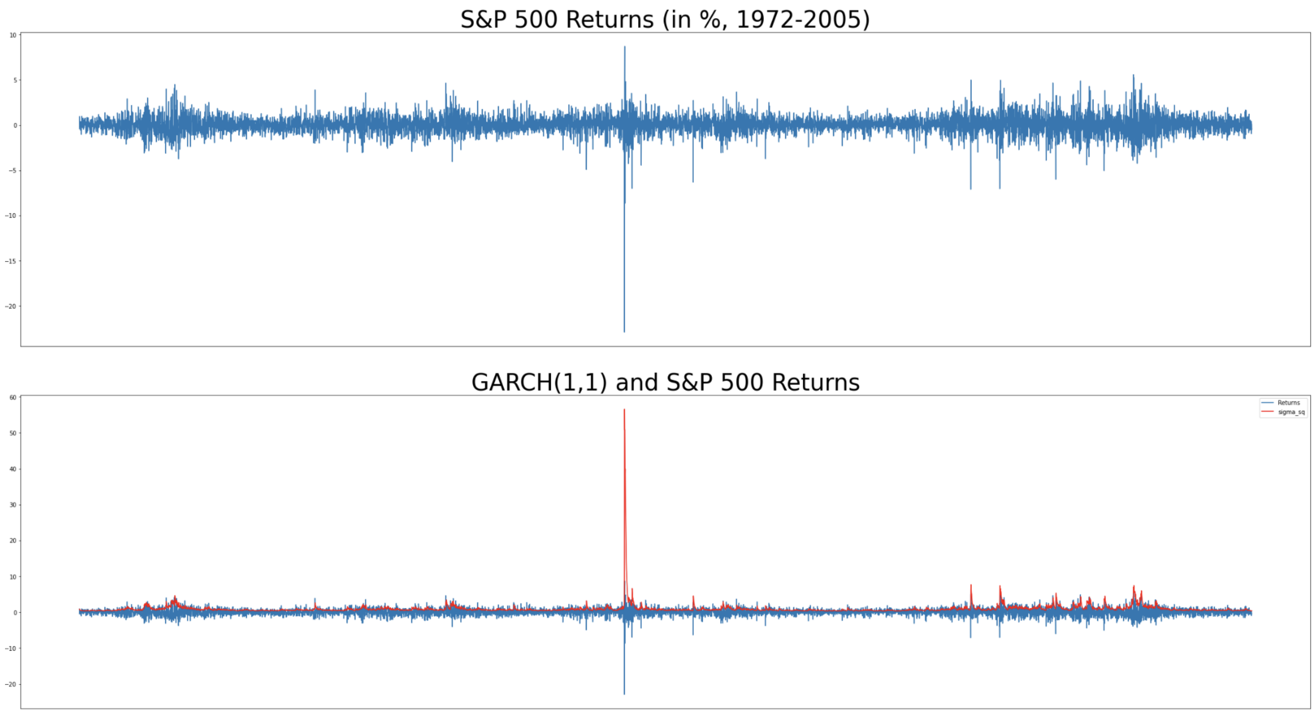

Figure 1: Plotted returns and estimated volatility

4 Model Checks

Prior Predictive Analysis

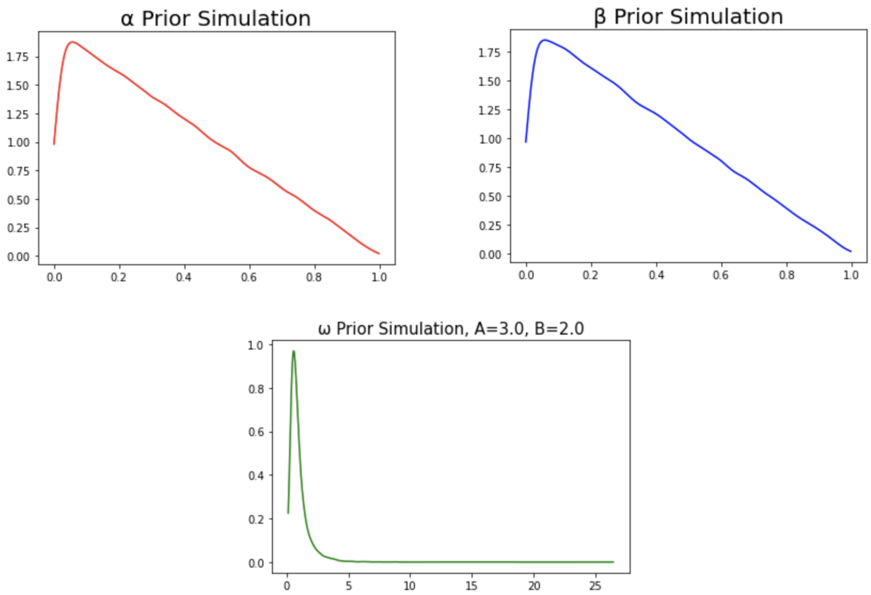

Figure 2: α, β, ω prior distribution densities

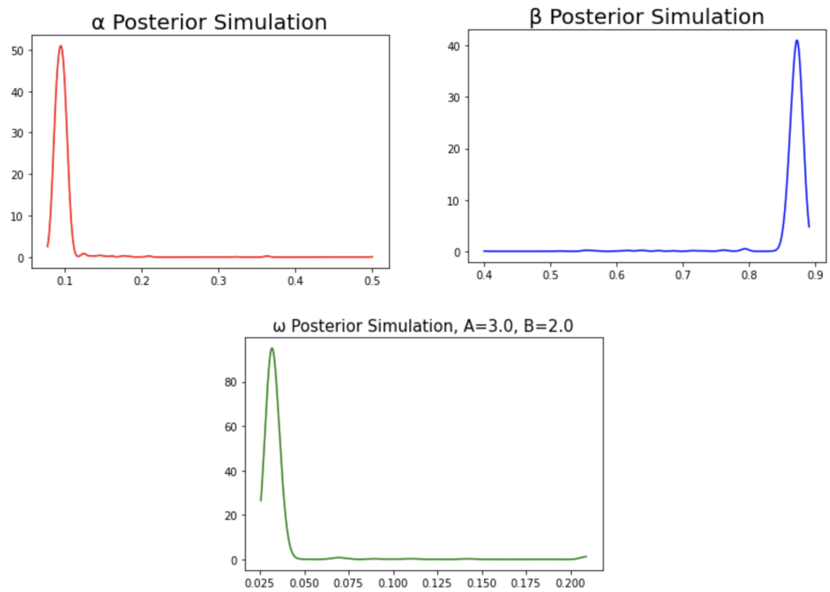

Figure 3: α, β, ω posterior distribution densities

Log-scores comparison to ARCH model

I utilized a simpler version of GARCH, the ARCH model, which only has an autoregressive component for the values of r. Specifically, I used an ARCH(1) model, which depends on one previous timestep.

$$\sigma_t^2 = \omega + \alpha$$

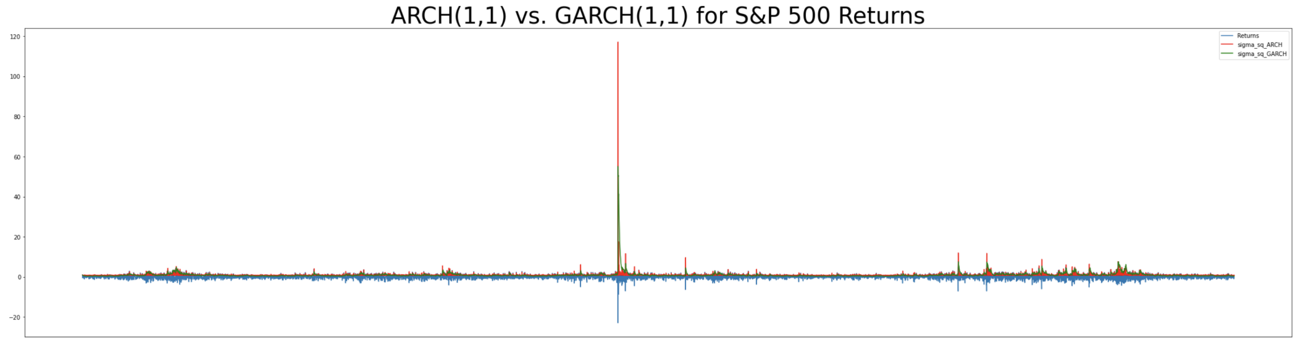

Figure 4: A baseline comparison of the ARCH vs. GARCH performance

In order to evaluate predictive performance, I split up the dataset into two portions:

- Training set (first 85% of data)
- Test set (remaning 15% of data)

I simulated the ARCH and GARCH models on the training set to obtain parameters fit to the first 85%, and then I computed predictive log-scores for the remaining 15%. The predictive log-scores were evaluated as a likelihood of the returns value given the predicted volatility of the model.

$$\text{Log-score at time } t = \log p(r_t | r_1, ...r_{t-1}) \qquad [4.2]$$

Where

$$p(r_t | r_1, ...r_{t-1}) \approx \frac{1}{M} \sum_{m=1}^{M} p(r_t | \sigma_t^{2(m)}) \qquad [4.3]$$

and

$$p(r_t | \sigma_t^{2(m)}) \text{ is } \mathcal{N}(0, \sigma_t^{2(m)}) \qquad [4.4]$$

Log-score results

- ARCH predictive log-score on test data: -396.6784834570662
- GARCH predictive log-score on test data: -465.00760728987825

(It looks as though ARCH performs better. I'm not sure why this is the case, potentially error in implementation or ARCH may do better for this specific dataset.)

5 Joint Distribution Tests

Geweke's joint distribution tests comparing prior and posterior densities of the parameters of interest.

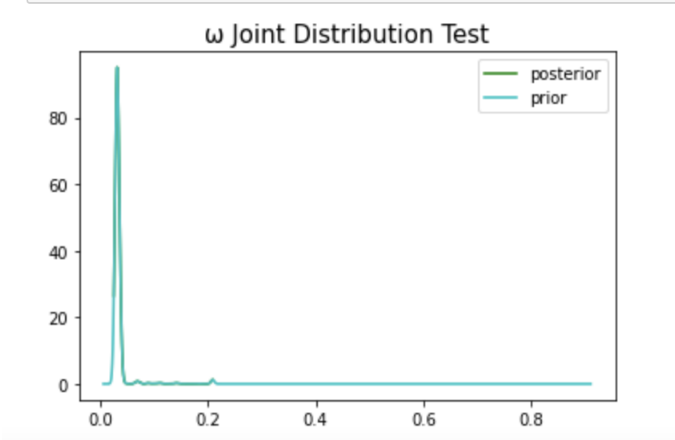

Figure 5: ω Joint Distribution Test