GARCH(1,1) for index returns

Giulio Galizio
William Junghahn Almqvist
Gian Martin Keller

MSc in Finance
Università della Svizzera Italiana
Lugano, Switzerland
16/01/2023

Introduction

The project we chose was based on implementing a program in R that would allow the estimation of a GARCH(1,1) model to check whether it was a meaningful model for checking its a valuable tool for financial predictions. Specifically, the points requested were as follows:

- Write a general function that estimates a GARCH(1,1) for a time series, and that returns the parameters, standard errors and the filtered variance process.
- Download at least 15 years of daily SP 500 data and estimate the GARCH model
- Use the estimated parameters and the filtered volatility to simulate a 95% confidence interval for a 30 day prediction period. Do this for every day in your sample.
- Verify how often the realizations 30 days ahead violate the confidence interval. Make a nice plot.

1 How to run the program

To run the program simply select all the lines of code and hit run, or do CTRL+ALT+B to run the whole program.
The graphs and data can be analysed in the DATA section at the top right and in the VALUES section, using the arrows below you can scroll through the various plots produced in the output.
The main file is called "Programming_Project.R" and can be read via any IDE that supports the R language, in our case we used "R studio".

2 Economic Problem and Theory

2.1 GARCH(1,1)

The Introduction of the Autoregressive Conditional Heteroscedasticity model (ARCH) and the generalized ARCH (GARCH) by Bollerslev (1986), contributed greatly to advanced econometric models. These models found positive feedback due to their ability to capture financial time series volatility clustering. The major difference in GARCH is that it makes a current conditional variance dependent on lags of its previous variance.

2.2 Definition of GARCH(1,1): Log-returns

A GARCH(1,1) is composed of daily log-returns and the variance associated with each return.

$$r_t = log(Price_t/Price_{t-1}) = \sigma_t * z_t \tag{1}$$

- r_t is the log-return of the current day, it's expected return is 0.
- $Price_t$ is the adjusted closing price of the current day.
- σ_t^2 is the volatility of the current day.
- z_t is an independent and identically distributed shock that has a standard normal distribution. Also called Innovations.

2.3 The GARCH(1,1) equation

$$\sigma_t^2 = \omega + \alpha * r_{t-1}^2 + \beta * \sigma_{t-1}^2 \tag{2}$$

- σ_t^2 is the conditional variance of the current day.
- r_{t-1}^2 is the squared return of the day before.
- σ_{t-1}^2 is the conditional variance of the day before.
- ω, α and β are the parameters that we want to optimize in order to estimate the filtered variance.

2.4 Factors that can affect the accuracy of predictions

There are many factors that can affect the accuracy of predictions made using GARCH (1,1) or any other statistical model.

- The quality and completeness of the data used to fit the model.
- The complexity of the model and its ability to capture relevant patterns and trends in the data.
- The presence of unexpected or extraneous factors that may affect the data.
- Financial markets are inherently unpredictable, and there is always some level of uncertainty in making predictions about the future performance of financial instruments.

3 Code description

First of all we download the data for the S&P500 index from Yahoo finance using the 'quantmod' package. The 'getsymbols()' is used to retrieve the data for the S&P500 index from 1988 to 2022.

3.1 Log returns

```
SP500return <- NULL
for( i in 2:length(SP500_1988_2022[,6])){
SP500return[i-1] <- log(SP500_1988_2022[i,6]/SP500_1988_2022[i-1,6])}
```

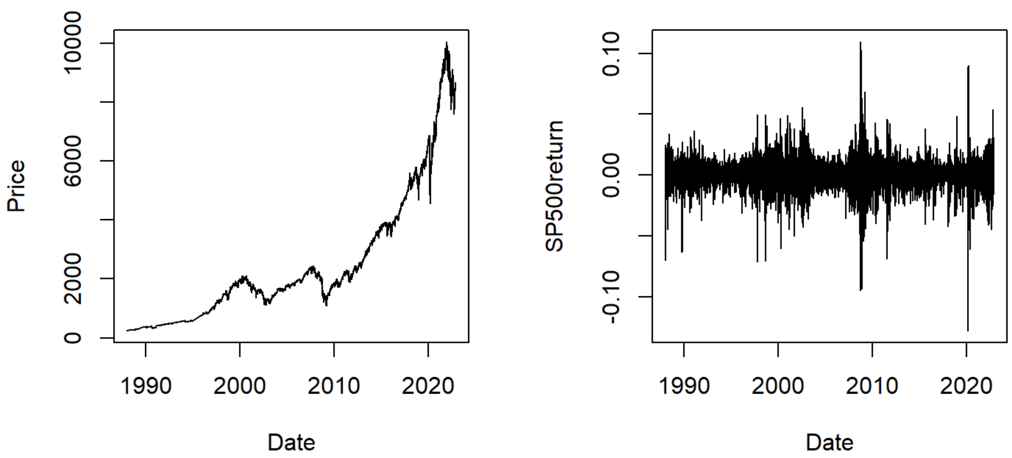
To compute the log returns, we divide the current day's adjusted closing price with the previous adjusted closing price. The for loop iterates over the adjusted closing prices of the S&P 500 index, starting at the second closing price, and calculates the log return for each day. The loop ends when it has iterated through the last closing price. The log returns are stored in the S&P500 return object, with the log return for the first day of the index being stored in the first element of S&P500 return, the log return for the second day being stored in the second element, and so on. We have to add that the adjusted closing prices are located on the 6th column of our dataset (SP500_1988_2022).

3.2 Daily closing price & daily log returns

Now we created 'Date' as an object converting the row names of the 'SP500_1988_2022' object to dates and 'Price' is created by the conversion of the closing price column to numeric values. The first Plot of the function 'Date_Price' representing the daily adjusted closing price and the second using 'Date_Return' is the Daily log returns.

```
Date_Price <- data.frame(Date, Price, stringsAsFactors = FALSE)
plot(Date_Price, type="l", xlab="Date", ylab="Price")

Date_Return <- data.frame(Date[-1], SP500return, stringsAsFactors = FALSE)
plot(Date_Return, type="l", xlab="Date")
```



We can see in the second graph that the returns are centered around 0, which confirms the assumption that the expectation of the returns is 0.

3.3 Garch Recursion and Parameters

The function takes three arguments: omega, alpha, beta, and return. The function returns a vector of "Garch values", which are the predicted variances of the returns.

The function first initializes an empty vector called Sigma2 to store the "Garch values". It then loops through the elements of the return vector and calculates the "Garch value" for each return using the Garch(1,1) model that we previously defined.
The if statement at the beginning of the loop is used to handle the case where i is 1, which is the first element of the return vector. We chose to use the variance of the returns as the initial "Garch value".

```
if (i == 1){
    Sigma2[i] <- var(return)
}
else
      # garch(1,1) model:
      Sigma2[i] <- omega + alpha*(return[i-1])^2 + beta*Sigma2[i-1]
}
return(Sigma2)
```

The purpose of this function is to avoid having to much code inside the main function, so we divided the steps and made separate secondary functions.

3.4 Garch loglikelihood function

We create a new function called Garch_Loglikelihood with a given set of parameters and returns. The goal is to compute the loglikelihood value for n daily conditional variances, with a given set of starting Garch parameters. We use the Garch recursion function to store the conditional variances in a vector, which are then going to be used in the loglike formula.

The loglikelihood function is obtained from the joint density of the log-returns, which have a Normal distribution with mean 0 and variance equal to the conditional variance:

$$Loglike = -0.5 * \sum_{t=1}^{n} [log(2\pi) + log(\sigma_t^2) + r_t^2/\sigma_t^2] \tag{3}$$

The log-likelihood formula is used to measure the fit of the GARCH model for the given returns, we want that value to be optimized, which means that we have to find the best Garch parameters. This is done with the Maximum Likelihood Estimator(MLE), which maximizes the loglike by alternating the parameters.

3.5 Main GARCH(1,1) function

Now, we start with our main function. This function uses the previously defined functions. First, we find the MLE by minimizing the negative Log-Likelihood function using 'optim'. By default, optim minimizes, which is why we use the negative Loglike. We gather the maximum likelihood estimates(the optimized parameters) to calculate the conditional variance for our given returns using our previously mentioned 'GARCH recursion function'. Then we calculate the mean of the conditional variance, which is the expectation of the conditional variance.

Next, we take the square root of the expected future variance to get the filtered volatility, note that we set the innovations to zero for the initial value. The equation for the Filtered Volatility is the following:

$$\sigma_{t+h} = \sqrt{\omega + \alpha \cdot r_t^2 + \beta \cdot \sigma_t^2} \tag{4}$$

Omega, alpha and beta are the GARCH parameters, and r_t is the return at time t.

3.6 Simulating volatility and confidence interval

Using the equation from before (4), we computed the future variance for a period of 30 days, this corresponds to the loop:

```
Future_variance <- NULL
for (i in 1:30){
  if(i==1){
    Future_variance[i] <- Future_sigma2
  }
  else{
    Future_variance[i] <- Estimated_omega + Estimated_alpha*mean(return^2)
                          + Estimated_beta*Future_variance[i-1]
  }
}
```

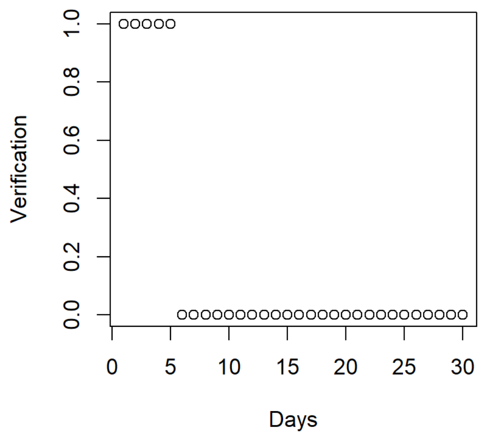
We then created a 95% C.I for the filtered volatility:

```
Margin <- qt(0.975, df= length(return)-1)*sqrt(var(return))/sqrt(length(return))
  for (i in 1:30) {
    Lower[i] <- c(sqrt(Future_variance[i]) - Margin)
    Upper[i] <- c(sqrt(Future_variance[i]) + Margin)
  }
  Confidence_interval <- cbind(Lower, Upper)
```

The purpose of this C.I is to see how many realizations violated it, we found that 16.6% of the realizations did not violate the C.I, which is low but understandable because it is hard to be very accurate in a 30 day horizon. For a lower horizon, we could say that GARCH(1,1) can be more accurate, as the 1st five predictions did not violate the C.I. We have to add that this percentage increased when we added more returns, but we couldn't get more prices below 1988.

```
Verification <- c(Filtered_volatility >= Confidence_interval[, 1] &
                  Filtered_volatility<=Confidence_interval[, 2])
"Percentage that does not violate CI: 16.67 %"
```

Verification plot:



3.7 Conclusion

To conclude, the GARCH(1,1) may be a good way to model the volatility of daily returns of the S&P500 index. Five out of thirty of our predictions do not violate the confidence interval. This seems like a relatively decent number considering the low amount of observations(8798). However, other models may be even more accurate for predicting volatility, so GARCH(1,1) may not be the best option.

4 References

- Loriano Mancini, *Asset Returns: Stylized Facts and ARCH/GARCH models.*
- Bionic Turtle, *Volatility: GARCH 1,1 (FRM T2-23)*, Youtube