Università della Svizzera Italiana

Department of Economics

Programming in Finance and Economics I

GARCH and index returns

Alessandro Dodon, Matteo Paesanti, Maddalena Stella

a.y. 2024/2025

Submission date: January 17, 2025

Contents

1 Planning

The first step involved careful study of the econometric theory of the model, consulting online material and tutorials. After downloading and preparing the SP 500 data, calculating log returns and de-meaning and assuming a zero mean ($\mu = 0$) for simplicity, we started building the functions for the GARCH(1,1) model. During this step, we faced several challenges in deciding whether to scale the data or not, and coming up with accurate standard errors (SE) and parameter estimates. The second biggest challenge was implementing the nested loop structure for simulating 30-day returns and aligning the outputs correctly. However, after receiving guidance from professors and AI, we finally obtained results that matched our theoretical expectations. Finally, we expanded our work including the Student's t-distribution to test a more advanced model. Different checks, debugging statements, and comparisons of our results with those generated by the ARCH library in Python were computed in the process.

2 User Guide

- Requirements: Ensure Python is installed with `numpy`, `pandas`, `matplotlib`, `statsmodels`, `arch`, `seaborn`, `scipy`, and `yfinance`. Missing libraries can be installed by removing the `#` in front of the `!pip install` commands at the beginning of the notebook. Internet connection is needed.
- Running the Program: Open the `Garch.ipynb` file in Jupyter Notebook or a compatible environment and execute the command "run all".
- Customizing Parameters: The user can, if desired: modify initial guesses (`omega`, `alpha`, `beta`, `nu`) in the GARCH function; adjust iterations or periods in simulation loops; change the start and end

dates in the `yfinance` section. Please note that all modifications are optional, as the program runs automatically.

- Results and Visualizations: Displayed directly below each cell using `print()` statements and in-line plots. No results or plots are stored externally.
- Data Handling: SP 500 data is automatically downloaded via `yfinance`. No manual file inputs or additional setup are needed.

Note: The notebook contains all necessary clarifications about major passages, in case some technical issues should not be clear.

3 Economic Problem and Motivation

The purpose of this project is to evaluate whether the GARCH(1,1) model is a suitable tool for modeling financial index returns. Using 16 years of SP 500 daily data, we aim to capture key features of financial time series, such as periods of clustered volatility and extreme returns, which are difficult to handle with traditional models. The GARCH(1,1) model addresses these challenges by accounting for changes in volatility over time. Our analysis involves estimating model parameters, creating 95% confidence intervals (CIs) for a 30-day forecast horizon, and checking how often actual returns fall outside these intervals. To better capture the extreme values in financial returns, we test the model under both normal and Student's t-distributed shocks. Comparing its performance under these two assumptions allows us to evaluate the model's ability to predict returns and assess financial risk.

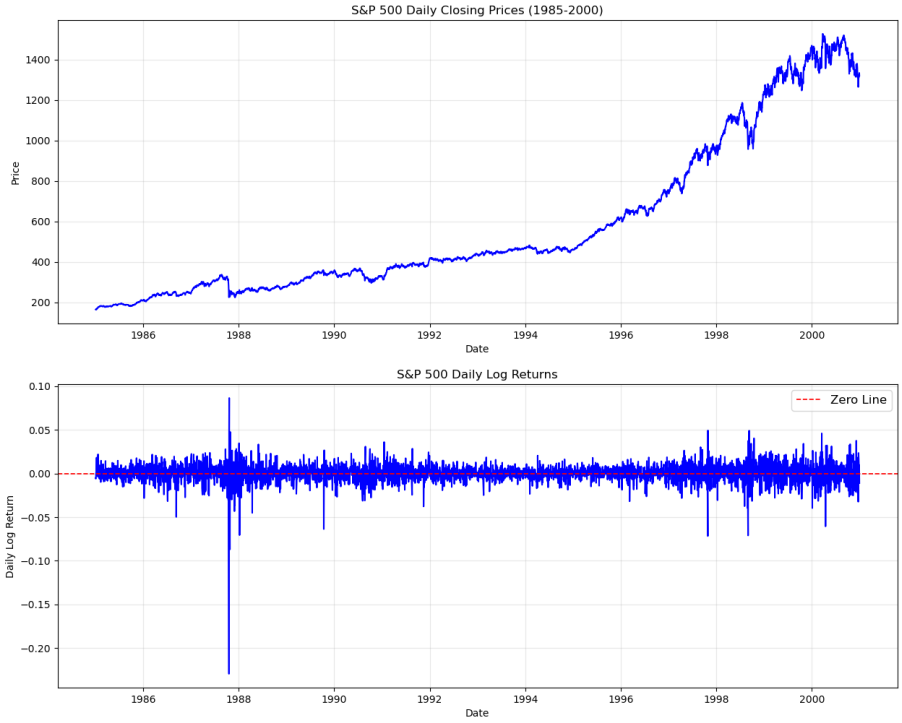
4 Data Preparation for GARCH Analysis

The data preparation for the GARCH(1,1) model involved the following steps:

- Data Download: SP 500 index daily adjusted closing prices from January 1, 1985, to December 31, 2000 were directly dowloaded via the yfinance library. No manual file handling.
- Data Cleaning, Validation and De-meaning: A double-check was performed to ensure no missing values. After calculating log returns, the dataset was validated to confirm that the mean of the de-meaned returns was approximately zero.
- Calculating Log Returns: To prepare the data for modeling, log returns were computed as:

$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right),$$

  where r_t represents the daily log return, P_t is the adjusted closing price at time t, and P_{t-1} is the price at $t-1$.
- Autocorrelation: Autocorrelation of both closing prices and log returns were plotted to evaluate dependencies.



5 GARCH(1,1) Model under Normality Distribution

- The GARCH(1,1) model assumes the following structure for returns r_t at time t:

$$r_t = \sigma_t z_t, \quad z_t \sim N(0,1)$$

where z_t is a standard normal random variable, and σ_t^2 represents the conditional variance of r_t. The conditional variance is modelled as: $\sigma_t^2 = \omega + \alpha r_{t-1}^2 + \beta \sigma_{t-1}^2$

The parameters ω, α, β must satisfy the following constraints to ensure stability and non-negativity:

$$\omega > 0, \quad \alpha \geq 0, \quad \beta \geq 0, \quad \alpha + \beta < 1$$

- Derivation of the PDF of $r_t|\mathcal{F}_{t-1}$: Since $z_t \sim N(0,1)$, the conditional distribution of r_t, given the information set $\mathcal{F}_{t-1}$, is:

$$r_t|\mathcal{F}_{t-1} \sim N(0, \sigma_t^2)$$

- General PDF of a Normal Distribution: The probability density function of r_t is defined as:

$$f(r_t|\mathcal{F}_{t-1}) = \frac{1}{\sqrt{2\pi\sigma_t^2}} \exp\left(-\frac{r_t^2}{2\sigma_t^2}\right)$$

- Log-Likelihood Function: For T observations, the likelihood function is defined as:

$$L(\theta) = \prod_{t=1}^{T} f(r_t|\mathcal{F}_{t-1}), \quad \theta = (\omega, \alpha, \beta)$$

Taking the logarithm to simplify computations and substituting the expression for $f(r_t|\mathcal{F}_{t-1})$, the log-likelihood can be written as:

$$\ell(\theta) = \sum_{t=1}^{T} \left(-\frac{1}{2}\ln(2\pi) - \frac{1}{2}\ln(\sigma_t^2) - \frac{r_t^2}{2\sigma_t^2} \right)$$

The parameters ω, α, and β are estimated by maximizing the log-likelihood function using numerical optimization, as closed-form solutions are unavailable. The conditional variance σ_t^2 is recursively calculated using initial values derived from the sample variance of returns.

Description of the Code

The GARCH(1,1) estimation is performed in two connected steps: first, the negative log-likelihood function is computed using the GARCH recursion; second, this function is minimized to estimate the parameters, standard errors, and conditional variances. Both functions are defined in the code with clear docstrings, enclosed by triple quotes (`"""`) to document their purpose, inputs, and outputs.
1) `garch_normal_log_likelihood`: This function computes the negative log-likelihood of a GARCH(1,1) model under the normality assumption. It recursively estimates the conditional variance starting from the sample variance and sums up the log-likelihood. The negative log-likelihood is then returned for optimization.
2) `fit_garch_normal_model` : This function estimates the GARCH(1,1) parameters $[\omega, \alpha, \beta]$ using maximum likelihood estimation (MLE). The main steps are as follows:

- Initialization: Parameters ω, α, and β are initialized to starting values ($\omega = 0.01, \alpha = 0.08, \beta = 0.89$).
- Optimization: The negative log-likelihood is minimized using SciPy's `minimize` function with the L-BFGS-B algorithm, ensuring the required constraints.
- Standard Errors: Computed from the inverse Hessian matrix of the optimized parameters.
- Variance Update: The conditional variance σ_t^2 is recursively updated using the GARCH(1,1) equation.

Results: The final outputs include the estimated parameters (ω, α, β), their associated standard errors, and the complete series of filtered variance.

Simulation of the 30 Days Returns

Our approach was as follows: first, we simulated 30-day return paths for each day in the sample, corresponding to approximately 4042 days over 16 years. The simulated returns were then aligned so that

each actual return could be directly compared to its corresponding simulations. From these aligned returns, we estimated the 95% confidence intervals (CIs) for each day and calculated the violation rate. The nested loop we used works as follows: the outer loop repeats the simulation process for each day in the dataset, starting with the day's filtered variance as the initial variance. For each day, the inner loop simulates a 30-day return path. At each step t of the 30-day path, a random shock $z_t \sim N(0,1)$ is drawn from a normal distribution, and the return is calculated as:

$$r_t = \sigma_t z_t, \quad \text{where } \sigma_t = \sqrt{\sigma_t^2}$$

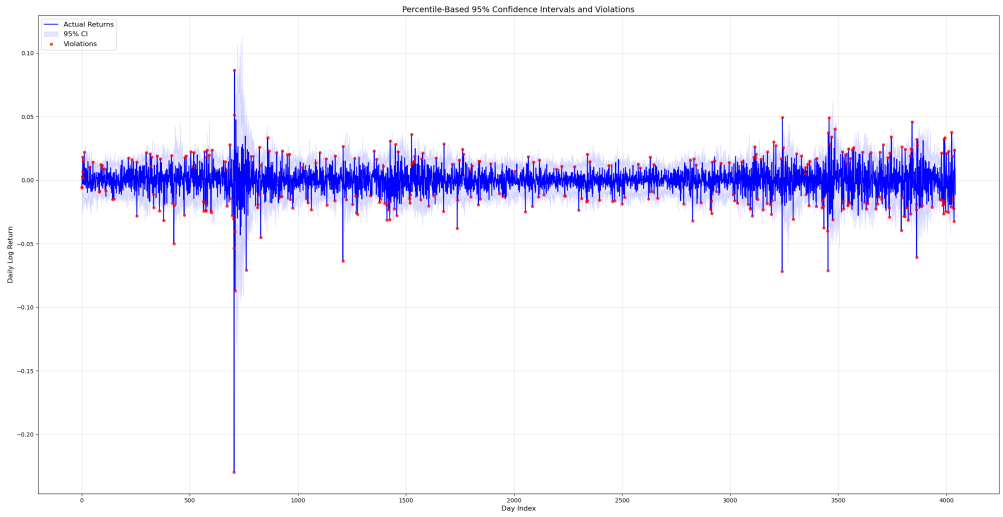
The variance is then updated recursively using the GARCH(1,1) equation.

Confidence Intervals and Violation Rate

To evaluate the reliability of our model, we construct confidence intervals (CIs) for simulated returns using a percentile-based approach, defining the 95% CI by the 2.5th and 97.5th percentiles of the empirical distribution. This captures 95% of simulated outcomes, providing a robust range for expected returns. A violation occurs when the observed return (r_t) falls outside the CI, specifically when $r_t <$ Lower Bound or $r_t >$ Upper Bound. For each day, violations are counted by iterating through the dataset, and the violation rate is calculated as:

$$\text{Violation Rate} = \frac{\text{Total Violations}}{\text{Total Sample Days}} \times 100$$

This process produces arrays for the daily lower and upper bounds, along with the total violations and the violation rate. In our analysis, the violation rate is 7.89%, reflecting the proportion of days where observed returns fall outside the 95% CI.



6 GARCH(1,1) Model under Student's t-Distribution

The GARCH(1,1) model can be extended to incorporate Student's t-distributed shocks, introducing an additional parameter ν, representing the degrees of freedom. This extension allows the model to better capture the heavy tails typically observed in financial time series.

Key Adjustments

- Log-Likelihood Function: The log-likelihood function under Student's t-distribution is:

$$\ell(\theta, \nu) = \sum_{t=1}^{T} \left[\ln \Gamma\left(\frac{\nu+1}{2}\right) - \ln \Gamma\left(\frac{\nu}{2}\right) - \frac{1}{2} \ln \pi(\nu-2) - \frac{1}{2} \ln \sigma_t^2 - \frac{\nu+1}{2} \ln \left(1 + \frac{r_t^2}{(\nu-2)\sigma_t^2}\right) \right],$$

  where $\nu > 2$ ensures finite variance.

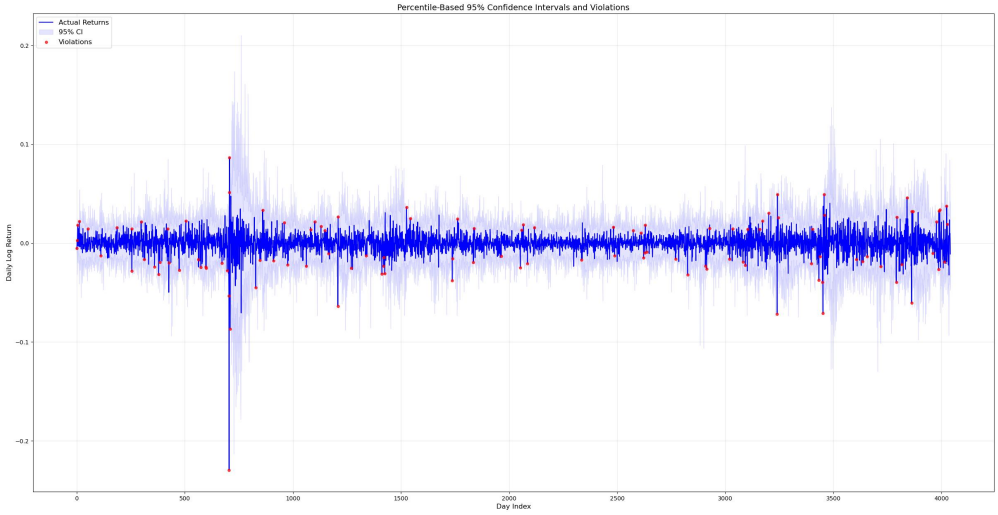
- Parameter Estimation: Parameters $(\omega, \alpha, \beta, \nu)$ are estimated by maximizing the log-likelihood function with additional constraint: $\nu > 2$.

Simulation of 30-Day Returns

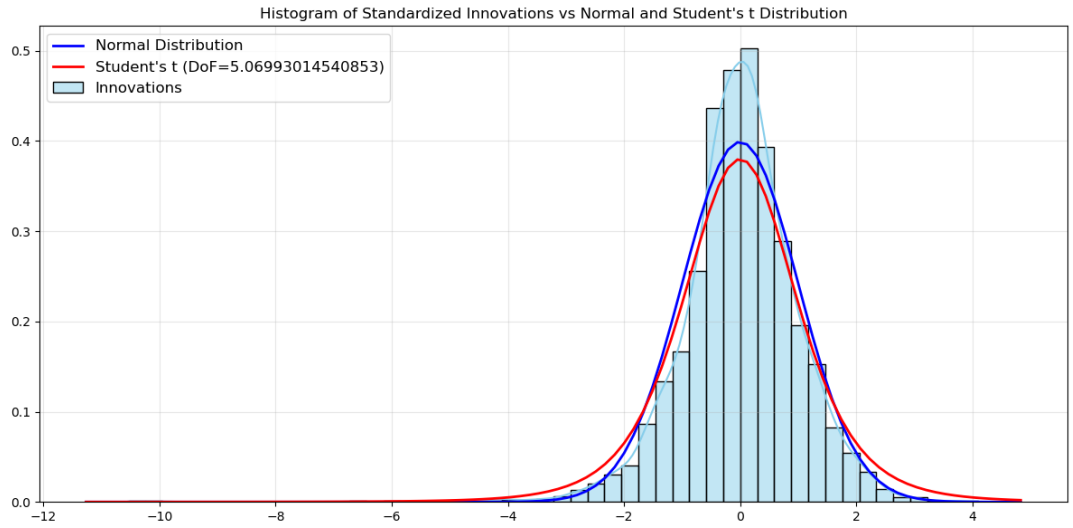
The simulation process for 30-day returns follows the methodology used for the normal GARCH model, the key difference is that shocks z_t are drawn from a Student's t-distribution with ν degrees of freedom.

Confidence Intervals and Violation Rate

The process for constructing confidence intervals (CIs) and calculating the violation rate is consistent with the normal GARCH model. For the Student's t-GARCH model, the violation rate is 2.82%.



7 Visualization and Interpretation of Results



To conclude our analysis, we evaluated the GARCH(1,1) model's performance in modeling SP 500 daily returns under two distributional assumptions. A well-calibrated model should produce a violation rate close to the theoretical 5% for a 95% confidence interval (CI), but our findings reveal notable deviations. These discrepancies can be attributed to the non-normality of actual returns. Nevertheless, it is important to note that the violations are derived from shocks simulated using a normal or Student's t distribution, with parameters estimated based on the respective model assumptions. The GARCH(1,1) model with normal shocks exhibited a violation rate of approximately 7.89%, underscoring its inability to account for extreme market movements. In contrast, the Student's t-GARCH model achieved a lower violation rate of around 2.82%, indicating better alignment with the data, though suggesting a slight overestimation of risk due to the lower-than-expected violation rate. Visually, the confidence intervals for the t-GARCH model appear broader, confirming its ability to accommodate extreme returns.

Insights from Parameter Estimates

The estimated parameters provide additional insights into the model's behavior. For the normal GARCH(1,1) model, the parameters are:

$$\omega = 0.000001, \quad \alpha = 0.086507, \quad \beta = 0.905270, \quad \alpha + \beta \approx 0.9918.$$

For the Student's t-GARCH(1,1) model, the parameters are:

$$\omega = 0.000001, \quad \alpha = 0.051163, \quad \beta = 0.943842, \quad \nu = 5.069930.$$

Validation against the ARCH package in Python showed consistent results overall. However, ω was significantly affected when scaling returns (e.g., multiplying by 100), a natural outcome due to its pro-

portionality to the square of the scaling factor. Discrepancies in standard errors likely arise from our manual methods' limitations with the Hessian matrix, while automatic ARCH packages use more robust optimization techniques.

Conclusion

The Student's t-GARCH model proves superior in capturing financial returns with heavy tails. Compared to the normal GARCH model, which underestimates risk and produces narrower confidence intervals, the t-GARCH model generates wider intervals that better reflect extreme market movements. This project provided a valuable opportunity to deepen our understanding of the GARCH model, a key topic in econometrics, while gaining practical Python programming skills. The manual implementation of functions and loops, though challenging, offered essential practice in function design, iterative programming, numerical methods, and Maximum Likelihood Estimation (MLE). Despite its classic nature, the GARCH model remains versatile, accommodating distributions like the asymmetric Student's t and Generalized Error Function (GEF), and can be extended to Bayesian frameworks or leverage effects. Its applications in risk management and derivatives pricing highlights its enduring relevance in financial econometrics.

8 Testing and Deployment

The code was validated by comparing results with the `arch` library's automated models, ensuring accurate parameter estimation, filtered variance, and confidence intervals. Each code chunk was verified by analyzing intermediate outputs and using debugging statements, along with the `%whos` command, to check matrix and vector dimensions. All debugging statements were removed in the final version for clarity. A full `Run All` confirmed seamless execution without errors. The program is standalone and does not require manual modifications before execution. Below we list all the versions used in this project (editor, extensions, environment, and libraries):

- VS Code: 1.95.3
- Python Extension: v2024.22.0
- Jupyter Extension: v2024.10.0
- Python: 3.12.4
- numpy: 1.26.4
- pandas: 2.1.4
- matplotlib: 3.7.5
- seaborn: 0.13.2
- yfinance: 0.2.50
- scipy: 0.14.2
- statsmodels: 0.14.2
- arch: 7.2.0

9 Appendix

AI Utilization

- Coding: ChatGPT 4o was primarily used for debugging and constructing specific parts of the code, and also provided general coding assistance. The instances where it was used the most are documented in the notebook.
- Testing: AI provided suggestions for handling edge cases and debugging errors.
- Report Writing: AI was occasionally used to refine report sentences, making them shorter or clearer, and to assist with Latex formatting.
- Ideation and Planning: The primary ideas and overall structure were developed by us with guidance from our professors. ChatGPT 4o was mainly used for confirmation and refinement of concepts.

Author Declaration

We hereby certify that:

- We have written the program ourselves except for clearly marked pieces of code and mentions of Generative AI.
- We have tested the program and it ran without crashing.
- We note that minor variations in results may occur depending on the computer and its system environment.
- We received guidance and suggestions from USI Professors Peter H. Gruber and Loriano Mancini during the development of this project.

References

- Data source: Yahoo Finance.
- Course materials:
  - Mancini, Loriano. Asset Returns: Stylized Facts and ARCH/GARCH Models. USI Lugano and Swiss Finance Institute, Financial Econometrics Course.
  - Barigozzi, Matteo. GARCH Models. London School of Economics, UK. CIDE–Bertinoro Course for PhD Students, 2014.
  - Kogan, Leonid. Volatility Models. MIT Sloan, Course 15.450, Fall 2010.
- Websites: Programming forums such as Stack Overflow (https://stackoverflow.com).