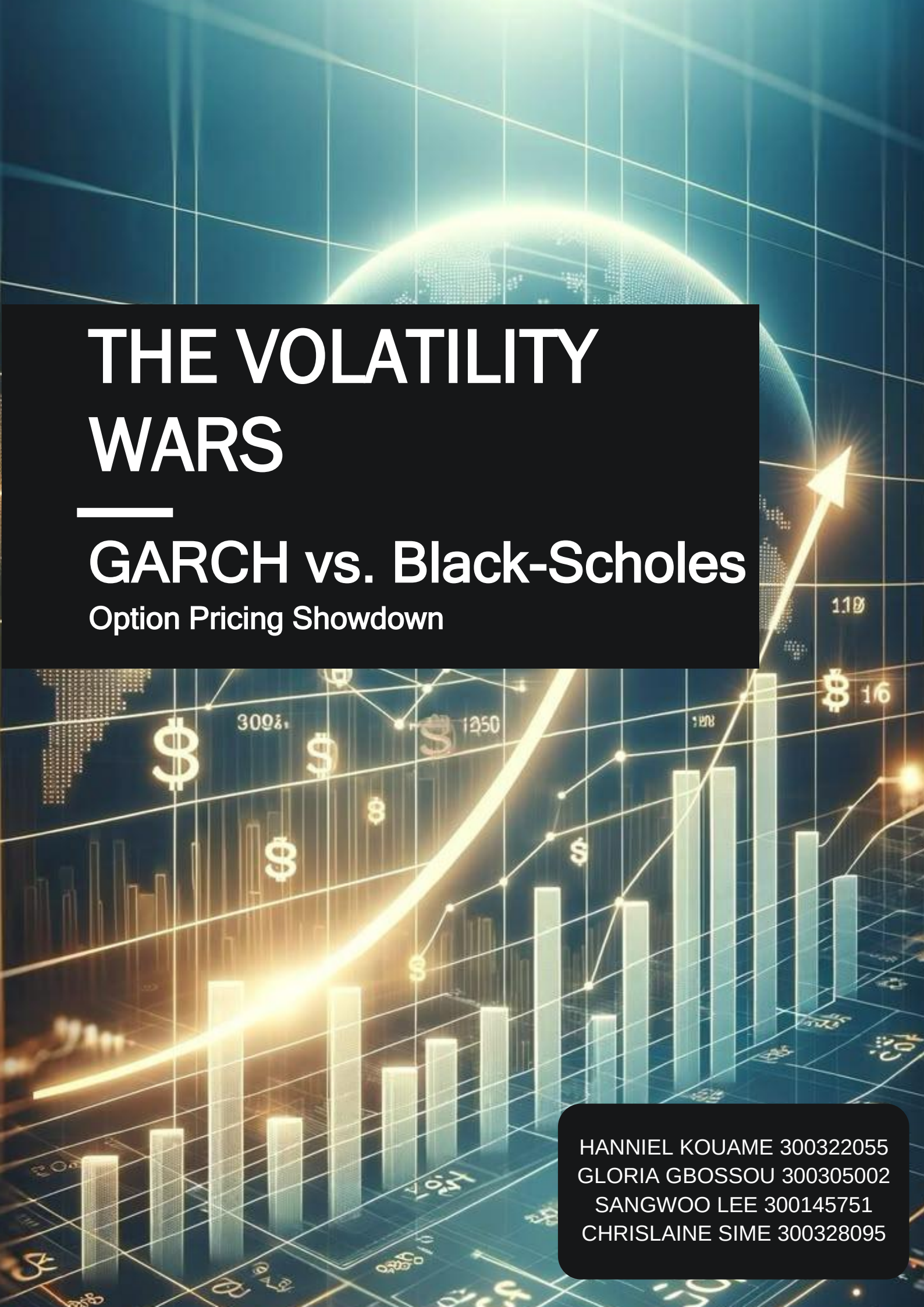


THE VOLATILITY WARS

GARCH vs. Black-Scholes

Option Pricing Showdown

HANNIEL KOUAME 300322055
GLORIA GBOSSOU 300305002
SANGWOO LEE 300145751
CHRISLAINE SIME 300328095

Contents

Introduction

The objective of this analysis is to explore the volatility patterns of the S&P 500 index and evaluate how different models such as Black-Scholes and GARCH affect the accuracy of option pricing. To do so, we begin with an examination of the underlying data, focusing on the structure and characteristics of the time series before moving on to more advanced modeling.

For this study, we use the daily closing prices of the S&P 500 index, obtained through the tidyquant package in R. This package retrieves financial data from Yahoo Finance, which itself compiles information from various exchanges and financial data providers. The data spans a ten-year period from March 15, 2015, to March 15, 2025, and consists of daily observations that reflect the index's market capitalization-weighted value.

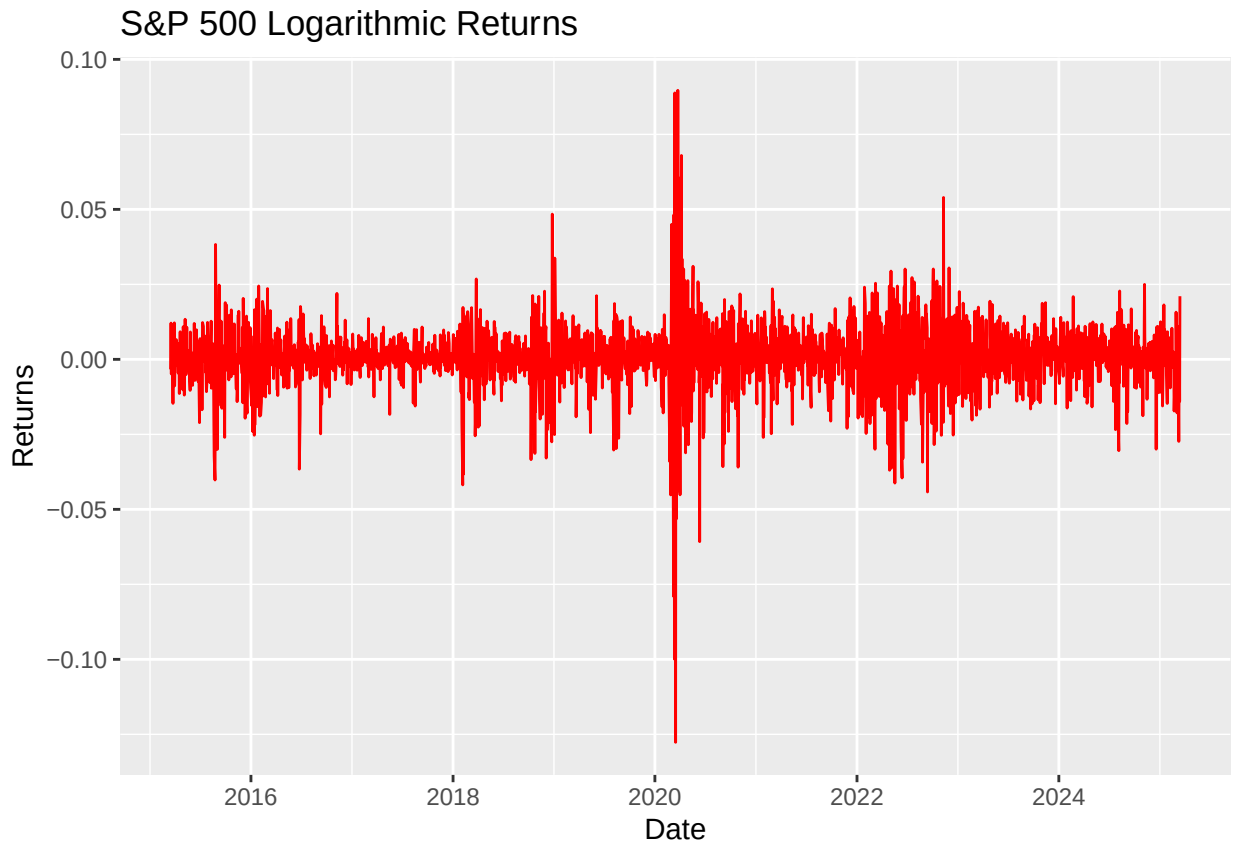
To prepare the data for volatility modeling, we compute the log returns of the daily closing prices. This transformation helps stabilize the variance and makes the time series more suitable for analysis.

Any resulting missing values were removed to ensure consistency throughout the sample. No further transformations were applied, preserving the original behavior of the series, especially the volatility dynamics that are central to this study.

The S&P 500 is one of the most widely followed equity indices in the world, making it a relevant and insightful candidate for financial modeling. Its behavior not only reflects the dynamics of the U.S. equity market but also serves as a benchmark for numerous financial instruments, including derivatives. Understanding its volatility structure is thus of both academic and practical interest.

This plot of S&P 500 closing prices effectively highlights periods of high volatility, particularly during economic downturns and crises. The general upward trend reflects long-term market growth, but noticeable sharp declines indicate moments of economic distress. For instance, the significant drop around 2020 likely corresponds to the COVID-19 market crash, a period of extreme uncertainty and rapid fluctuations. Similarly, other dips, such as those seen around 2022, may be linked to inflation concerns, interest rate hikes, or geopolitical tensions. These fluctuations emphasize the market's sensitivity to economic events, where investor sentiment shifts dramatically in response to financial instability. The presence of such volatility underscores the importance of market resilience and long-term investment strategies, as historical data suggests that despite temporary downturns, the S&P 500 tends to recover and continue its upward trajectory over time.

Financial markets are characterized not only by changes in asset prices but also by fluctuations in volatility over time. Traditional models like Black-Scholes assume that volatility is constant; however, empirical evidence from financial data, such as the S&P 500, suggests otherwise.



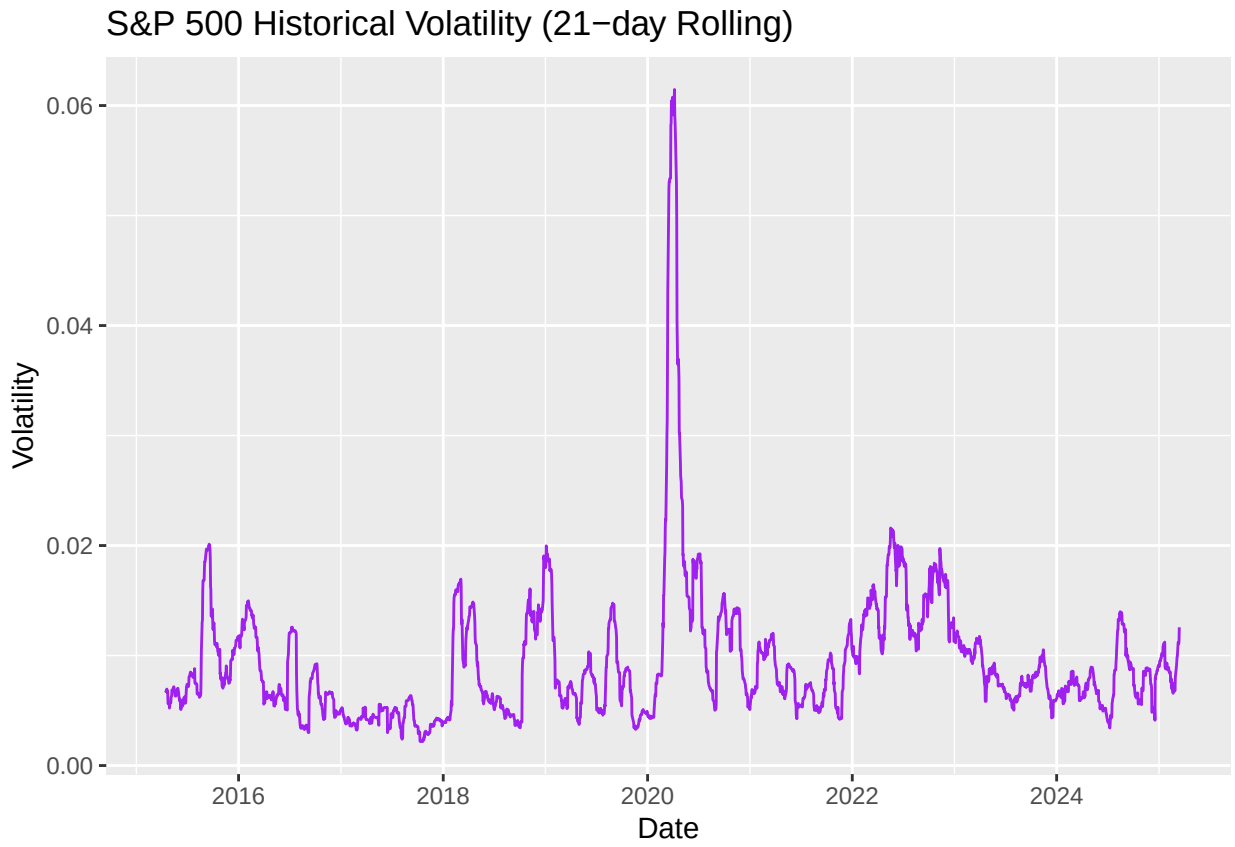
The S&P 500 returns exhibit clear volatility clustering

This plot of S&P 500 logarithmic returns clearly illustrates how the amplitude of returns fluctuates over time, reflecting changes in market volatility. Periods of relative stability are characterized by smaller fluctuations, while episodes of heightened uncertainty, such as the sharp spikes around 2020, indicate times of extreme market stress. The significant increase in return amplitude during that period aligns with the onset of the COVID-19 pandemic, when uncertainty led to dramatic price swings. Similarly, other visible fluctuations may correspond to economic events such as interest rate hikes or geopolitical tensions. This variation in return magnitude suggests that market volatility is not constant but rather evolves in response to economic conditions, investor sentiment, and external

shocks. This suggests volatility is time-varying, not constant.

Historical Volatility Analysis

Volatility is a key concept in finance, representing the degree of variation in asset returns over time. A common way to estimate it is through historical volatitily. Historical volatility is a common measure of return variability. Show volatility changes over time without modeling.



Historical volatility fluctuates significantly, confirming that volatility is not constant.

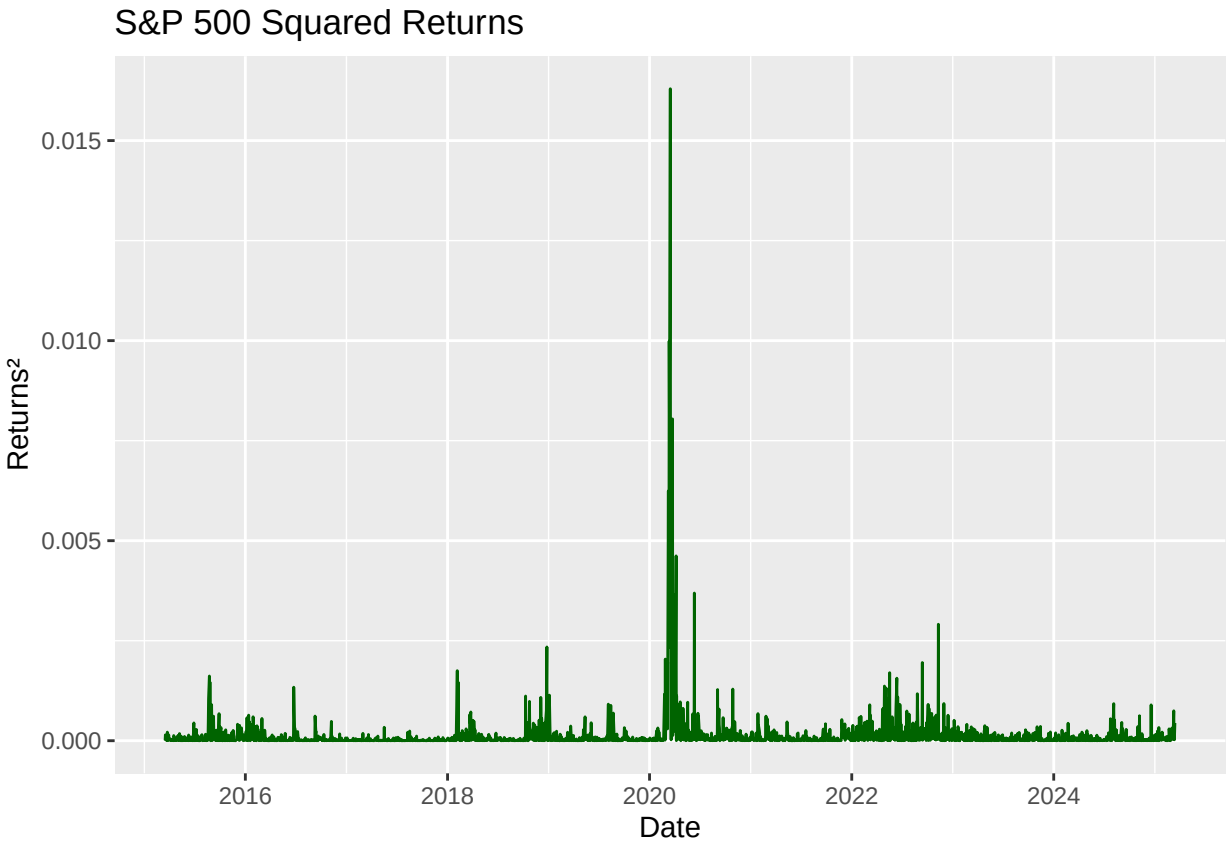
While the general volatility levels remain low during stable market conditions, sudden surges indicate periods of heightened uncertainty and risk. The most prominent spike, seen around 2020, aligns with the onset of the COVID-19 pandemic, when market reactions were extreme due to unprecedented uncertainty. Other smaller peaks, such as those observed in 2018 and 2022, may be linked to economic slowdowns, policy changes, or geopolitical tensions. These fluctuations highlight the dynamic nature of financial markets, where external shocks and investor sentiment play crucial roles in shaping price movements.

This motivates modeling volatility dynamically.

Now, we want to check whether volatility changes over time, which would justify using an ARCH model.

Detecting Heteroskedasticity and ARCH Models

Squared Returns and Volatility Clustering



This plot of S&P 500 squared returns illustrates clusters of volatility, suggesting the presence of an autoregressive conditional heteroskedasticity (ARCH) effect. Periods of low volatility are interspersed with episodes of heightened market fluctuations, indicating that volatility is not constant over time but instead exhibits persistence. The most noticeable spike occurs around 2020, likely corresponding to the financial turbulence caused by the COVID-19 pandemic. Other smaller clusters of volatility are visible throughout the timeline, reflecting market reactions to economic events, policy changes, or geopolitical uncertainties. The presence of volatility clustering implies that large price movements are often followed by further large movements, either positive or negative, which is a key characteristic of financial time series.

Engle's ARCH Test

The ARCH test formally checks for conditional heteroskedasticity:

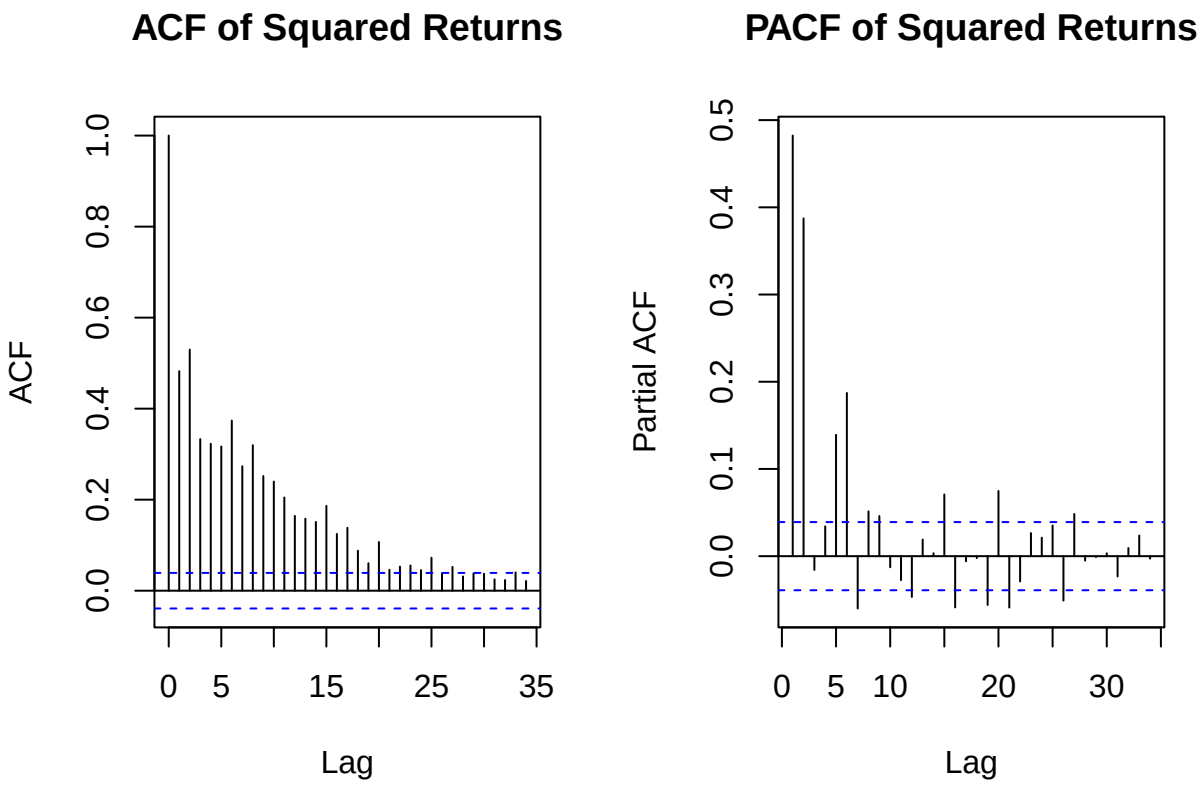
To formally test for volatility clustering, we conduct Engle's ARCH LM test (Engle, 1982). The null hypothesis assumes no ARCH effects (i.e., constant volatility). A rejection implies that volatility dynamics are predictable from past squared returns.

The test yields chi-square(10) = 974.58 ($p < 0.001$), overwhelmingly rejecting the null. This confirms that S&P 500 returns exhibit significant ARCH effects, validating the need for conditional

volatility models like ARCH/GARCH.

This aligns with our earlier observations:

Volatility clusters in crises (e.g., 2008, 2020).



The ACF/PACF of squared returns shows significant lags, implying past squared returns predict future volatility. This is called an ARCH effect : evidence that volatility depends on past shocks.

Introducing ARCH Model

We choose ARCH(1) because it's the simplest model that captures the core idea of volatility clustering : the observation that large price movements tend to be followed by more large movements (and calm periods by more calm). The "1" in ARCH(1) means volatility today depends only on yesterday's squared return, which directly translates the empirical fact we see in the ACF/PACF of squared returns (where lag 1 is significant). It's a minimal but meaningful starting point: it proves volatility reacts to past shocks without overcomplicating the story.

After preparing the S&P 500 returns data by removing missing values, we fit an ARCH(1) model to capture the time-varying volatility dynamics. The ARCH(1) specification assumes that today's conditional volatility depends on the magnitude of yesterday's squared return, formalized as:

$$X_n = \mu + \epsilon_n$$

where:

$$\epsilon_n = \sigma_n Z_n,$$

with:

- $\{Z_n\} \sim \text{WN}(0, 1)$, a white noise process with unit variance,
- and the conditional variance evolving according to:

$$\sigma_n^2 = \alpha_0 + \alpha_1 \epsilon_{n-1}^2,$$

where:

- $\alpha_0 > 0$ ensures positive variance,
- $\alpha_1 \geq 0$ measures the influence of past shocks on today's volatility.

In the ARCH(1) model:

- A large shock yesterday (ϵ_{n-1}) increases today's volatility (σ_n^2).
- A small shock yesterday reduces today's volatility.

Thus, today's volatility depends only on the squared return shock from the immediately preceding period.

The estimated coefficients for the S&P 500 returns are:

$\mu = 9.465971 \times 10^{-4}$

$\omega = 6.535707 \times 10^{-5}$

$\alpha_1 = 5.130371 \times 10^{-1}$

So the full model becomes:

$$\sigma_n^2 = 6.535707 \times 10^{-5} + 5.130371 \times 10^{-1} \epsilon_{n-1}^2,$$

The positive and statistically significant coefficient $\alpha_1 = 0.513$ suggests that large past shocks captured by squared returns lead to increased current volatility. This behavior reflects the phenomenon of volatility clustering, where periods of high volatility tend to persist over time, as seen in the raw return data during turbulent market phases.

The ω term, estimated at 6.535707×10^{-5}, represents the baseline level of volatility in the absence of recent shocks. It sets the minimum variance level the model expects when past returns have little influence.

The ARCH(1) model confirms that volatility is not constant. However, ARCH(1) has limitations: it only reacts to the most recent shock and ignores longer-term volatility persistence. This motivates extending the model to GARCH(1,1), which accounts for both recent shocks and past volatility (via the β_1 term).

To improve the model, we will test for residual autocorrelation (Ljung-Box test) to verify no ARCH effects remain and compare AIC/BIC values with GARCH(1,1) to justify the more flexible model.

Extending to Garch(1,1)

While the ARCH(1) model captures how shocks affect volatility, it ignores the persistence of volatility over time : a critical feature of financial markets. The GARCH(1,1) model addresses this by adding a term (β_1) that accounts for past volatility itself:

$$X_n = \mu + \epsilon_n$$

where:

$$\epsilon_n = \sigma_n Z_n,$$

with:

- $\{Z_n\} \sim \text{WN}(0, 1)$, a white noise process with unit variance,
- and the conditional variance evolving as:

$$\sigma_n^2 = \alpha_0 + \alpha_1 \epsilon_{n-1}^2 + \beta_1 \sigma_{n-1}^2,$$

where:

- $\alpha_0 > 0$ ensures positive variance,
- $\alpha_1 \geq 0$ captures the impact of past squared returns (shocks),
- $\beta_1 \geq 0$ captures the persistence effect of past volatility.

The GARCH effect ($\beta_1 \sigma_{n-1}^2$) captures the gradual decay of volatility over time.

The estimated GARCH(1,1) model for S&P 500 returns yields the following coefficients:

$\omega = 3.866763 \times 10^{-6}$

$\alpha_1 = 0.1835699$

$\beta_1 = 0.7878797$

the conditional variance equation becomes:

$$\sigma_n^2 = 3.866763 \times 10^{-6} + 0.1835699 \epsilon_{n-1}^2 + 0.7878797 \sigma_{n-1}^2,$$

The $\alpha 1$ coefficient of 0.1835 indicates that a 1% daily return shock increases next day's volatility by 0.1835%. This moderate reaction suggests the market absorbs shocks without excessive overreaction in normal conditions.

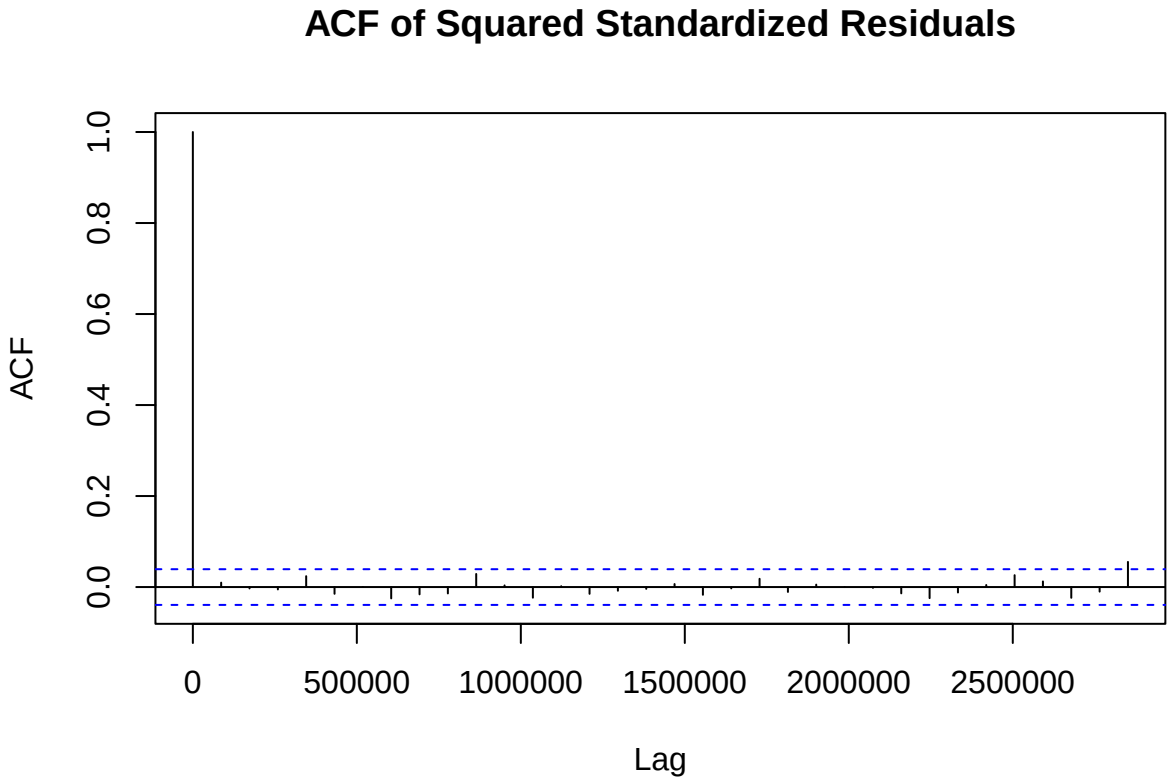
The high $\beta 1$ value of 0.7878 shows strong volatility persistence. Combined with $\alpha 1$, the persistence measure ($\alpha 1 + \beta 1 = 0.9714496$) approaches the non-stationarity boundary of 1, implying shocks decay slowly with a half-life of:

Half-life= $\log(0.5)/\log(0.9714496) = 24$(approx.) trading days. This explains why volatility clusters typically last about 1 month.

Model Diagnostics

The high p-value 0.7164 (» 0.05) indicates we fail to reject the null hypothesis of no autocorrelation in squared residuals.

This means the GARCH(1,1) model has successfully captured the volatility clustering present in the original data, leaving no significant ARCH effects in the residuals.



The plot should show no significant lags beyond the 95% confidence bounds.

Model Comparison and Validation : GARCH(1,1) vs. ARCH(1)

Our results show that the GARCH(1,1) model outperforms the ARCH(1) model, as indicated by lower AIC, BIC, and Hannan-Quinn values.

AIC (-6.627098 vs. -6.391740): The GARCH(1,1) model has a lower AIC, meaning it provides a better fit with fewer unnecessary parameters.

BIC (-6.613097 vs. -6.384739): Since BIC penalizes complexity more than AIC, the lower value confirms that GARCH(1,1) is the preferred model.

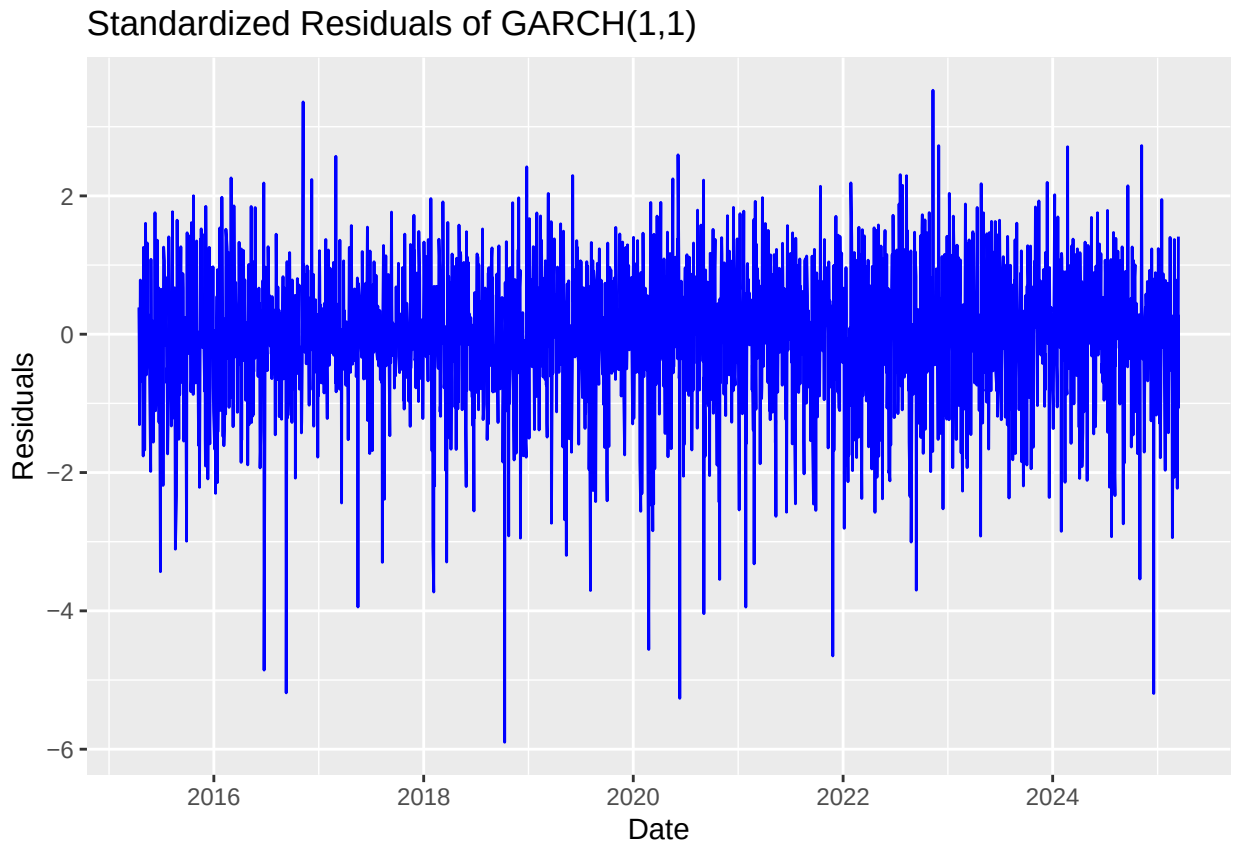
Hannan-Quinn (-6.627109 vs. -6.391743): Again, GARCH(1,1) performs better.

ARCH models assume volatility depends only on past squared returns, which may be insufficient.

GARCH models add an autoregressive term, allowing volatility to depend on both past squared returns and past variances.

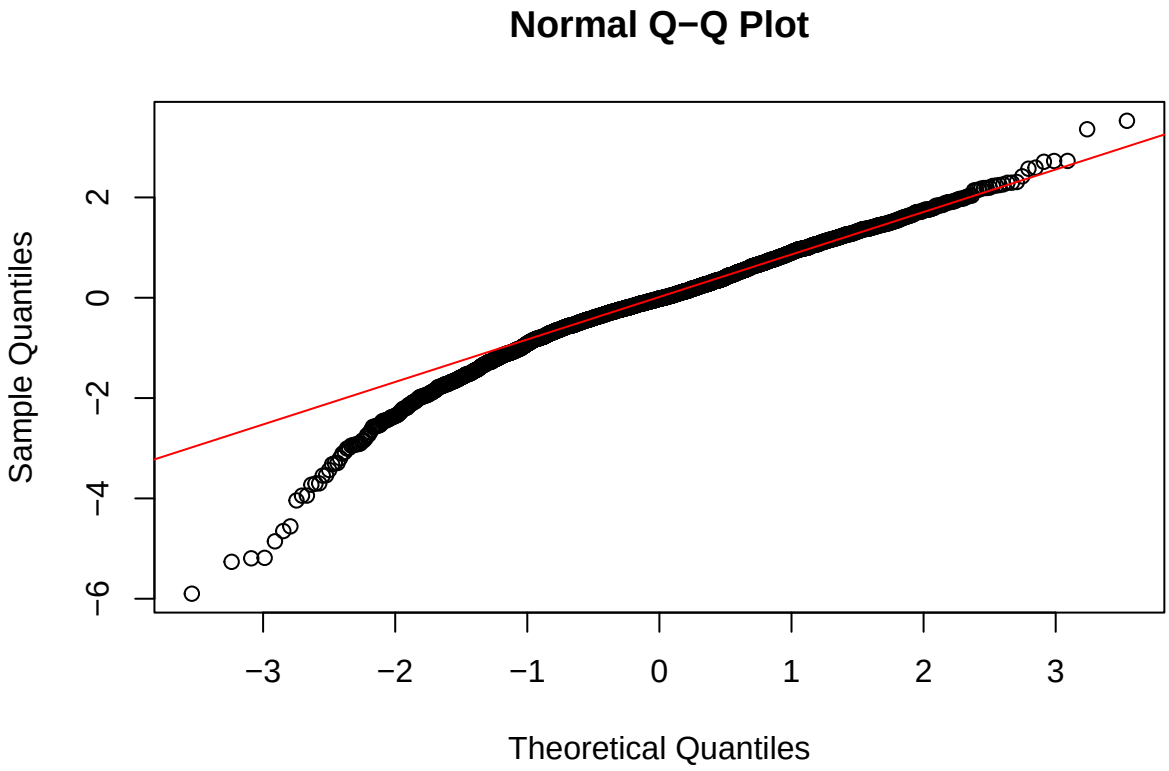
The AIC/BIC results confirm that GARCH(1,1) captures volatility dynamics more effectively than ARCH(1).

Residual Diagnostics



Volatility clustering is reduced but still present. This suggests that the GARCH(1,1) model has improved the fit but might not fully capture extreme volatility.

Residual Normality Test

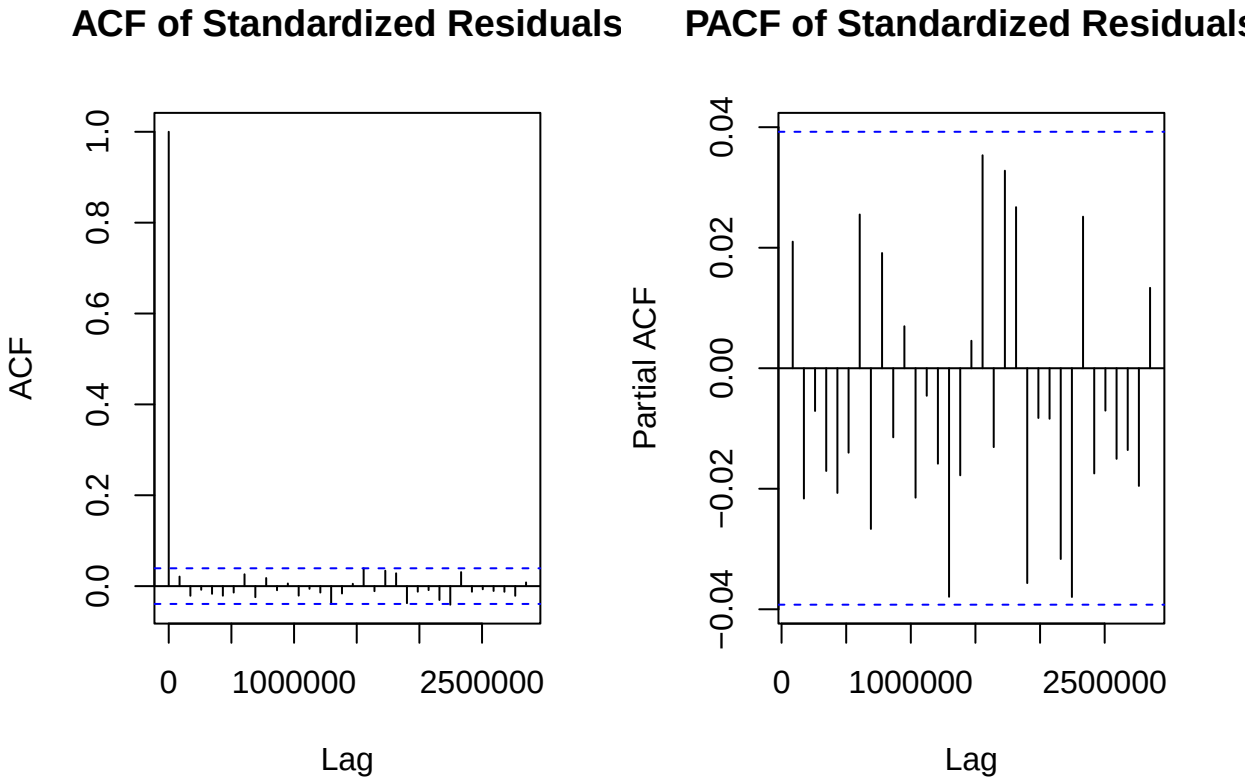


The QQ-plot shows that the right tail deviates from normality, while the left tail aligns well with the theoretical line. This suggests mild asymmetry and some fat-tail behavior on the negative side, indicating the model underestimates the probability of large negative returns.

Jarque-Bera Test

Jarque-Bera Test ($p < 2.2e\text{-}16$) : The residuals are not normally distributed. This is common in financial returns, which often exhibit fat tails.

Autocorrelation Check



The residuals behave like white noise, indicating that the model captures time dependencies well.

ARCH-LM Test

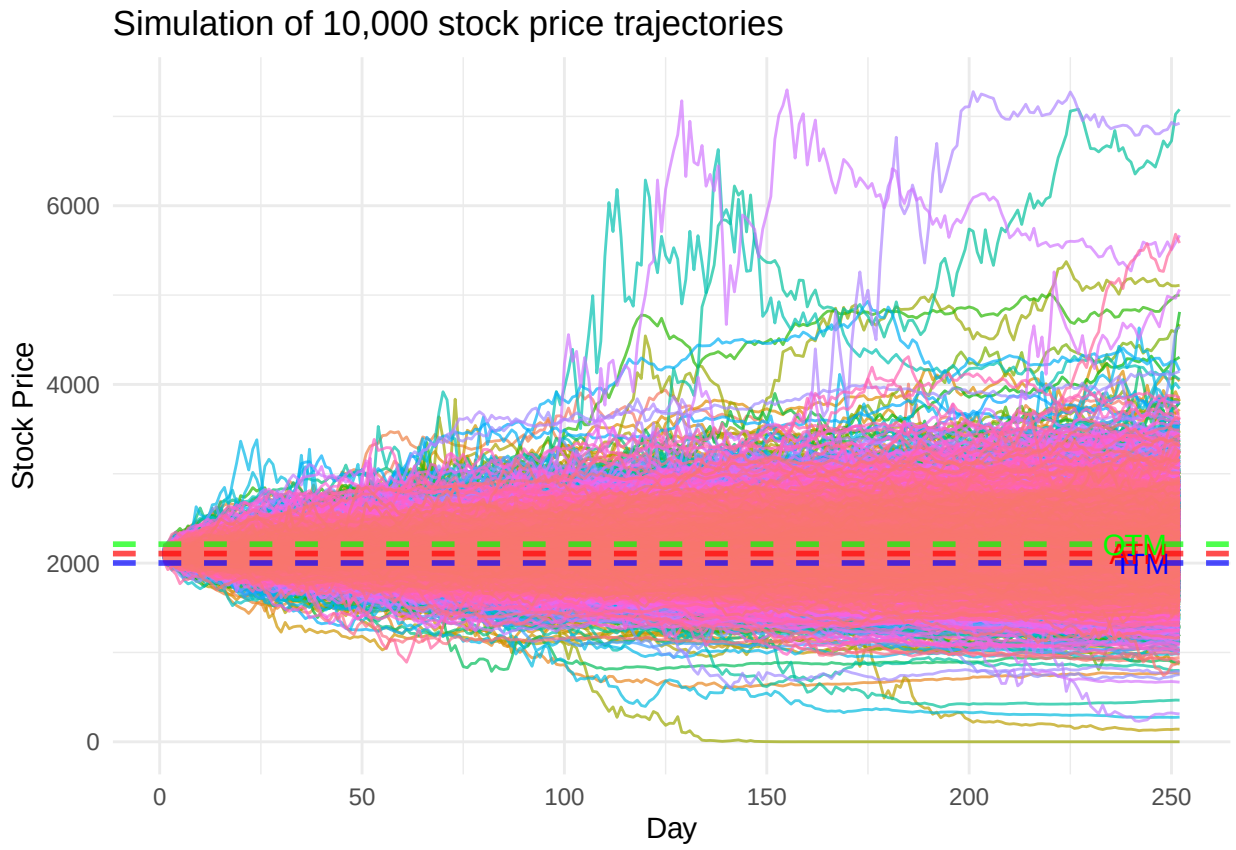
No significant ARCH effects remain, meaning the GARCH model sufficiently captures volatility clustering.

Since ARCH effects are gone, GARCH(1,1) is doing a good job.

Compute Implied Volatility from GARCH(1,1)

Simulate Future Volatility Paths

Once we have the estimated GARCH parameters, we can simulate future volatility paths based on these parameters.



Price the Option

GARCH(1,1)

With the simulated stock price paths generated from the GARCH(1,1) model, we can now proceed to estimate the fair prices of European call options. By evaluating the option payoffs at maturity across the different simulated scenarios and discounting them at the risk-free rate, we obtain Monte Carlo estimates of option prices for various moneyness levels. This approach captures the dynamic nature of volatility and allows for more realistic pricing compared to models assuming constant volatility.

```
##    Moneyness Call_Garch Put_Garch
## 1 ITM (0.95)   226.7380  79.02424
## 2 ATM (1.00)   166.7036 122.23565
## 3 OTM (1.05)   119.0726 177.85042
```

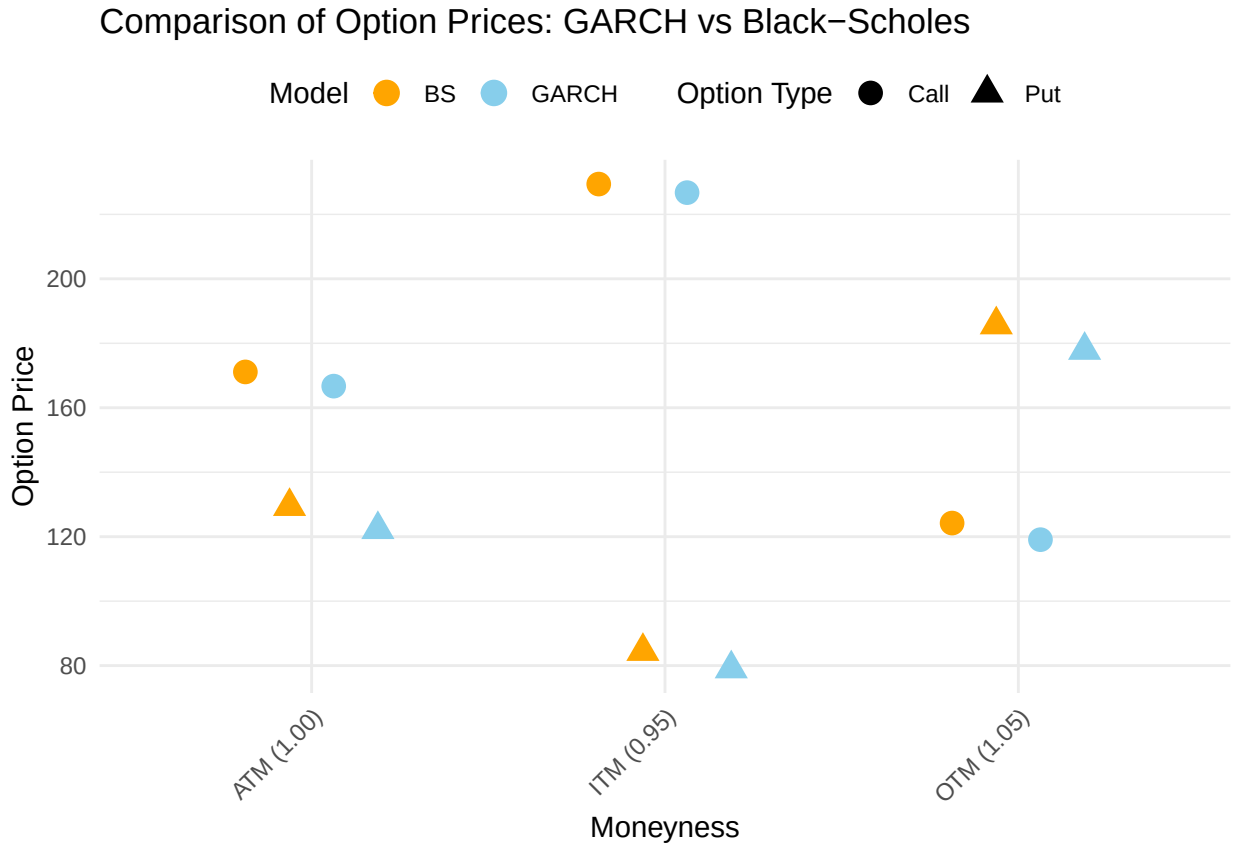
Black-Scholes

Additionally, we'll compute the option price using the Black-Scholes model, assuming constant volatility.

```
##    Moneyness  Call_BS    Put_BS
## 1 ITM (0.95) 229.3970  84.43711
## 2 ATM (1.00) 171.1154 129.40131
## 3 OTM (1.05) 124.2214 185.75314
```

Comparative Pricing Analysis : Black-Scholes vs. GARCH(1,1)

We compare the option prices derived from the Black-Scholes model and the GARCH(1,1) model to evaluate the impact of volatility assumptions on pricing accuracy. We aim to understand how each model handles market dynamics, and which approach provides a more realistic estimate of option prices under conditions of fluctuating volatility.



The differences in option pricing between the GARCH(1,1) and Black-Scholes models can be understood by examining how each approach handles volatility, particularly across moneyness levels. Black-Scholes assumes constant volatility and normally distributed returns, while GARCH incorporates time-varying volatility that responds to past shocks. This distinction becomes critical when pricing options with varying degrees of intrinsic and extrinsic value.

For in-the-money (ITM) options, where intrinsic value dominates, the pricing gap between the two models is relatively small. For example, ITM calls are priced at 226.74 (GARCH) vs. 229.40 (Black-Scholes) a difference of just 2.66. This narrow margin reflects the reduced reliance of ITM options on volatility forecasts, as their value is already anchored by the underlying asset's favorable price. However, even here, Black-Scholes' static volatility assumption slightly inflates premiums compared to GARCH's dynamic adjustments. The gap widens for ITM puts (79.02 vs. 84.44), where Black-Scholes overestimates downside risk by failing to model how volatility clusters fade over time.

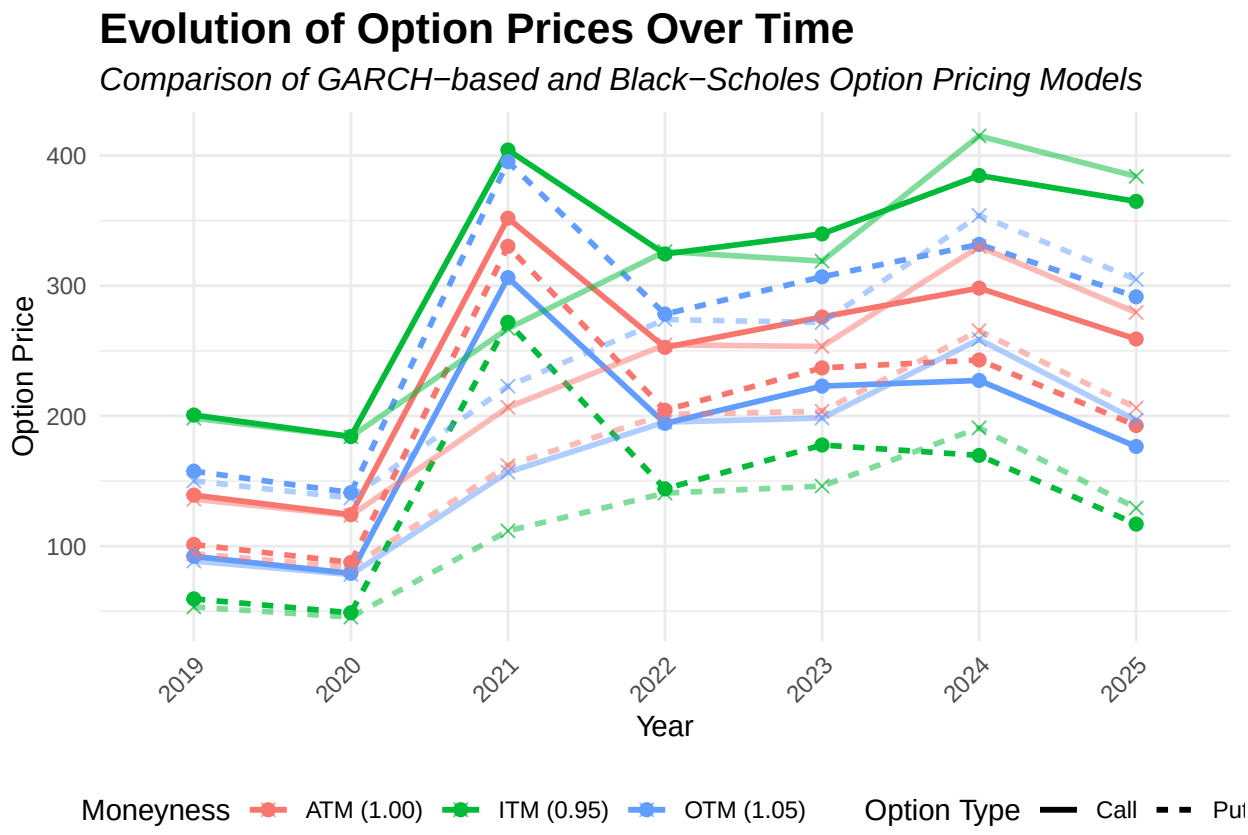
At-the-money (ATM) options, which balance intrinsic and extrinsic value, show more pronounced differences. ATM calls under GARCH (166.70) lag Black-Scholes (171.12) by 4.42, while ATM puts diverge by 7.16 (122.24 vs. 129.40). These disparities highlight Black-Scholes' rigidity: its constant volatility input averages historical turbulence, including past crises, which inflates premiums during calmer periods. GARCH, in contrast, tempers forecasts by weighting recent shocks ($\alpha 1 = 0.18$) against long-term volatility persistence ($\beta = 0.79$), resulting in leaner prices.

The largest discrepancies emerge with out-of-the-money (OTM) options, which rely entirely on future volatility to gain value. OTM calls under GARCH (119.07) trail Black-Scholes (124.22) by 5.15, while OTM puts diverge sharply by 7.90 (177.85 vs. 185.75). This asymmetry underscores Black-Scholes' tendency to overprice tail risks. By assuming constant volatility, it misjudges the likelihood of extreme market moves especially for OTM puts, which act as insurance against crashes. GARCH, however, models volatility as a decaying process, where shocks like the 2020 COVID-19 crisis influence forecasts but diminish over weeks. This leads to more conservative pricing that better reflects the market's gradual recovery from turbulence.

In summary, GARCH's dynamic volatility modeling offers a more realistic framework for option pricing, particularly for volatility-sensitive instruments like OTM options. Black-Scholes, while simpler, remains tethered to outdated assumptions, leading to systematic overpricing that reflects its inability to adapt to evolving market conditions.

Rolling Window Analysis

we perform a rolling window analysis to assess how the option pricing model evolves over time. By applying a fixed-size window that moves through the dataset, we can observe how volatility and price dynamics change as new data becomes available. This approach allows us to track the stability and adaptability of the model, highlighting potential shifts in market conditions and providing a more dynamic view of option pricing. The rolling window analysis helps evaluate the robustness of the model over different time periods and market environments.



The evolution of option prices over time, when visualized through a rolling window framework, reveals the strengths and limitations of each model's approach to volatility. In these plots, solid lines and filled circles represent GARCH-based prices, while faded lines and 'X' markers denote Black-Scholes prices

we observe how models like GARCH(1,1) and Black-Scholes adapt to shifting volatility regimes. During stable periods, both models exhibit modest divergence, particularly for at-the-money (ATM) options, where their predictions align closely. However, this harmony fractures during crises, such as the 2020 COVID-19 crash, when volatility clusters emerge. GARCH, with its ability to absorb recent shocks and decay past volatility, adjusts prices more responsively. For instance, out-of-the-money (OTM) puts, which act as crash insurance, see GARCH prices decline gradually post-crisis, reflecting its autoregressive structure. In contrast, Black-Scholes, anchored to a static volatility derived from historical averages, lags behind, persistently overpricing tail risks due to its inability to "forget" extreme events.

GARCH adapts; Black-Scholes stagnates.

Conclusion

In financial markets, volatility is the heartbeat of uncertainty, shaping how investors perceive risk and value derivatives. Our analysis of S&P 500 option pricing reveals a persistent divergence between the Black-Scholes (BS) model and the GARCH(1,1) framework, offering critical insights into why traditional models often misrepresent reality and why they remain stubbornly entrenched

in practice.

At first glance, the results appear counterintuitive. Black-Scholes, despite its well-documented flaws static volatility, normality assumptions, and ignorance of market turbulence produces higher option prices than GARCH(1,1), particularly for out-of-the-money (OTM) puts. This seems to contradict its reputation as an outdated tool. However, this discrepancy is not a validation of BS but rather a reflection of its inherent limitations. By relying on historical volatility, BS implicitly embeds past market crises, such as the 2020 COVID-19 crash, into a single constant parameter. This "one-size-fits-all" approach inflates volatility estimates, artificially amplifying the perceived risk of extreme price movements. In essence, BS overcompensates for its inability to model time-varying volatility by averaging turbulence across calm and crisis periods, leading to overstated premiums.

GARCH(1,1), in contrast, embraces the dynamic nature of markets. By allowing volatility to depend on recent shocks ($\alpha 1 = 0.18$) and past variance ($\beta 1 = 0.79$), the model captures the persistence and gradual decay of market stress. The combined persistence parameter ($\alpha 1 + \beta 1 = 0.97$) suggests that volatility shocks linger for weeks, aligning with the observed month-long clusters during crises. This dynamic adjustment results in more conservative volatility forecasts, translating to lower option prices. For OTM puts (options highly sensitive to tail risks) the difference is stark: GARCH prices are nearly 4.2% lower than BS, reflecting a tempered outlook on extreme downside risks.

Yet the story does not end here. While GARCH corrects BS's rigidity, it too faces challenges. Residual diagnostics reveal fat-tailed residuals, a hallmark of financial returns that neither model fully addresses. BS sidesteps this issue through its flawed normality assumption, while GARCH acknowledges it but lacks explicit mechanisms to price tail risks. Paradoxically, BS's reliance on historical volatility which inadvertently bakes past crises into its inputs grants it a veneer of robustness. It is a clumsy workaround, but one that masks its theoretical shortcomings in practice.

Why, then, does BS persist as an industry standard? The answer lies in pragmatism. Its simplicity and interpretability make it a lingua franca for traders, who often "fix" its flaws by using implied volatility a market-driven parameter that retrofits BS to reality. GARCH, while theoretically superior, demands more computational effort and nuanced calibration, limiting its adoption outside specialized contexts.

Our findings underscore a broader truth: no model is perfect, but some are more honest about their imperfections. BS's artificial performance in this analysis is a relic of its static worldview, while GARCH's lower prices reflect a more nuanced, adaptive understanding of volatility. For practitioners, the choice hinges on context. In stable markets, BS suffices as a quick approximation. In turbulent times or for tail-sensitive instruments, GARCH offers a clearer lens provided we pair it with tools like Student-t innovations to better capture extreme risks.

Ultimately, the gap between these models is not just a technical detail but a reminder that financial markets are living systems, resistant to simplification. As we refine our tools, we must balance elegance with humility, recognizing that every model is a shadow of the complexity it seeks to tame.

"All models are wrong, but some are useful." – *George E. P. Box*

References

- Boire, F-M. (2025, March 28 & April 2). Cours 20 et 21 – Hétéroscédasticité: Modèles ARCH et GARCH (p. 4–7). MAT 3779 – Introduction aux séries chronologiques, Département de mathématiques et de statistique, Université d'Ottawa.

Appendix

```
#Library
library(tidyquant)
library(quantmod)
library(rugarch)
library(RQuantLib)
library(dplyr)
library(ggplot2)
library(tseries)
library(xts)
library(FinTS)
library(scales)
library(gridExtra)
library(tidyr)
library(lubridate)
library(rugarch)
library(dplyr)
```

```
# Get S&P 500 data
sp500_data <- tq_get("^GSPC",
                     from = "2015-03-15",
                     to = "2025-03-15") %>%
  mutate(log_returns = log(close / lag(close))) %>%
  na.omit()  # Remove NA from returns

# Verify
#head(sp500_data)
```

Introduction

```
# Plot closing prices
ggplot(sp500_data, aes(x = date, y = close)) +
  geom_line(color = "blue") +
  labs(title = "S&P 500 Closing Prices", x = "Date", y = "Price")
```

```
# Plot returns
ggplot(sp500_data, aes(x = date, y = log_returns)) +
```

```
  geom_line(color = "red") +
  labs(title = "S&P 500 Logarithmic Returns", x = "Date", y = "Returns")
```

Historical Volatility Analysis

```
# Compute 30-day rolling volatility
sp500_data$volatility <- rollapply(sp500_data$log_returns, width = 21, FUN = sd, align = "right

# Plot volatility
ggplot(sp500_data, aes(x = date, y = volatility)) +
  geom_line(color = "purple") +
  labs(title = "S&P 500 Historical Volatility (21-day Rolling)", x = "Date", y = "Volatility")
```

Detecting Heteroskedasticity and ARCH Models Squared Returns and Volatility Clustering

```
# Plot squared returns
ggplot(sp500_data, aes(x = date, y = log_returns^2)) +
  geom_line(color = "darkgreen") +
  labs(title = "S&P 500 Squared Returns", x = "Date", y = "Returns²")
```

Engle's ARCH Test

```
# Perform ARCH test on returns
arch_test <- ArchTest(sp500_data$log_returns, lags = 10)
#print(arch_test)
```

```
par(mfrow = c(1, 2))
# Plot ACF/PACF of squared returns
acf(na.omit(sp500_data$log_returns^2), main="ACF of Squared Returns")
#
pacf(na.omit(sp500_data$log_returns^2), main= "PACF of Squared Returns")
#
```

Introducing ARCH Model

```
sp500_data <- na.omit(sp500_data)  # Remove NAs
# Fit ARCH(1) using `rugarch`
arch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 0)),  # ARCH(1)
  mean.model = list(armaOrder = c(0, 0), include.mean = TRUE)
)
arch_fit <- ugarchfit(arch_spec, data = sp500_data$log_returns)
coef(arch_fit)  # mu omega, alpha1
```

Extending to Garch(1,1)

```r
# Fit GARCH(1,1) model - PROPERLY NESTED
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(1, 1), include.mean = TRUE))

garch_fit <- ugarchfit(garch_spec, data = sp500_data$log_returns)
coef(garch_fit)

#show(garch_spec)
```

Model Diagnostics

```r
# Check residual autocorrelation
residuals_std <- residuals(garch_fit, standardize=TRUE)
Box.test(residuals_std^2, lag=10, type="Ljung-Box")
```

```r
# Plot ACF of squared residuals
acf(residuals_std^2, main = "ACF of Squared Standardized Residuals")
```

Model Comparison and Validation : GARCH(1,1) vs. ARCH(1)

```r
# Model comparison
garch_ic <- infocriteria(garch_fit)
arch_ic <- infocriteria(arch_fit)

criteria_table <- data.frame(
  Criterion = c("Akaike (AIC)", "Bayes (BIC)", "Hannan-Quinn"),
  GARCH_11 = c(garch_ic[1], garch_ic[2], garch_ic[3]),
  ARCH_1 = c(arch_ic[1], arch_ic[2], arch_ic[3]),
  Difference = c(garch_ic[1] - arch_ic[1], garch_ic[2] - arch_ic[2], garch_ic[3] - arch_ic[3])
)

criteria_table
```

Residual Diagnostics

```r
# Extract standardized residuals
std_residuals <- residuals(garch_fit, standardize = TRUE)

# Plot standardized residuals
ggplot(data.frame(Date = sp500_data$date, Residuals = std_residuals), aes(x = Date, y = Residua
  geom_line(color = "blue") +
  labs(title = "Standardized Residuals of GARCH(1,1)", x = "Date", y = "Residuals")
```

Residual Normality Test

```r
qqnorm(std_residuals)
qqline(std_residuals, col = "red")
```

Jarque-Bera Test

```r
jarque.bera.test(std_residuals)
```

Autocorrelation Check

```r
par(mfrow = c(1, 2))
acf(std_residuals, main = "ACF of Standardized Residuals")
pacf(std_residuals, main = "PACF of Standardized Residuals")
```

ARCH-LM Test

```r
ArchTest(std_residuals, lags = 10)
```

Compute Implied Volatility from GARCH(1,1) Simulate Future Volatility Paths

```r
# Simulate future volatility paths using the GARCH(1,1) model
S0 <- as.numeric(head(sp500_data$close, 1)) # Initial stock price
r <- 0.02  # Risk-free rate
n_sim <- 10000  # Number of simulations
n_days <- 252  # Number of trading days (1 year)
T <-1 # Time to maturity in years
dt <- T/n_days
moneyness_levels <- c(0.95, 1, 1.05)  # ITM, ATM, OTM
# Compute log returns
# Extract GARCH parameters from the fitted model
omega <- as.numeric(coef(garch_fit)["omega"])
alpha1 <- as.numeric(coef(garch_fit)["alpha1"])
beta <- as.numeric(coef(garch_fit)["beta1"])

# Initialize volatility path (variance, not standard deviation)

# Simulate future volatility paths
set.seed(111217)  # For reproducibility
garch_sim <- ugarchsim(fit=garch_fit,n.sim= n_days, m.sim=n_sim, rseed=111217)
volatility_paths_sd <- sigma(garch_sim) * sqrt(252)


# Simulate stock prices
set.seed(111217)
stock_paths <- matrix(NA, nrow = n_sim, ncol = n_days)

for (i in 1:n_sim) {
```

```
  S_t <- S0  # Start price
  for (t in 1:n_days) {
    # Get the volatility for this time step
    sigma_t <- volatility_paths_sd[t, i]
    # Simulate the stock price path using GBM formula
    z <- rnorm(1)
    S_t <- S_t * exp((r - 0.5 * sigma_t^2) * (1/252) + sigma_t * sqrt(1/252) * z)
    stock_paths[i, t] <- S_t
  }
}
```

```
# Sélectionner 10 simulations aléatoires
set.seed(111217)
sample_sim <- sample(1:n_sim, 10000)

# Créer un dataframe pour le graph
stock_paths_df <- as.data.frame(stock_paths[sample_sim, ])
stock_paths_df$Simulation <- factor(1:10000)

# Ajouter les noms de colonnes pour les jours
colnames(stock_paths_df) <- c(1:n_days, "Simulation")

# Transformation au format long
stock_paths_df <- pivot_longer(stock_paths_df,
                               cols = -Simulation,
                               names_to = "Day",
                               values_to = "Stock_Price")

# Convertir Day en numérique
stock_paths_df$Day <- as.numeric(stock_paths_df$Day)


ggplot(stock_paths_df, aes(x = Day, y = Stock_Price, color = Simulation , group = Simulation))
  geom_line(alpha = 0.7) +
  geom_hline(yintercept = S0 * moneyness_levels[1], linetype = "dashed", color = "blue", size =
  geom_hline(yintercept = S0 * moneyness_levels[2], linetype = "dashed", color = "red", size =
  geom_hline(yintercept = S0 * moneyness_levels[3], linetype = "dashed", color = "green", size
  labs(title = "Simulation of 10,000 stock price trajectories",
       x = "Day",
       y = "Stock Price") +
  annotate("text", x = max(stock_paths_df$Day), y = S0 * moneyness_levels[1], label = "ITM", hj
  annotate("text", x = max(stock_paths_df$Day), y = S0 * moneyness_levels[2], label = "ATM", hj
  annotate("text", x = max(stock_paths_df$Day), y = S0 * moneyness_levels[3], label = "OTM", hj
  theme_minimal() +
  guides(color = "none")  # Supprimer la légende
```

Price the Option

GARCH(1,1)

```r
#2. Garch
# Initialize storage for option prices
call_prices <- numeric(length(moneyness_levels))
put_prices <- numeric(length(moneyness_levels))

# Loop over different moneyness levels
for (i in seq_along(moneyness_levels)) {
  K <- S0 * moneyness_levels[i]  # Strike price

  # Initialize payoff storage
  payoffs_call <- numeric(n_sim)
  payoffs_put <- numeric(n_sim)

  # Calculate payoffs for each simulation
  for (j in 1:n_sim) {
    S_T <- stock_paths[j, n_days]  # Final stock price for this simulation

    # Payoff for Call and Put options
    payoffs_call[j] <- max(S_T - K, 0)
    payoffs_put[j] <- max(K - S_T, 0)
  }

  # Discount expected payoffs to present value
  call_prices[i] <- exp(-r * (n_days / 252)) * mean(payoffs_call)
  put_prices[i] <- exp(-r * (n_days / 252)) * mean(payoffs_put)
}

# Store the option prices in a data frame
option_prices <- data.frame(
  Moneyness = c("ITM (0.95)", "ATM (1.00)", "OTM (1.05)"),
  Call_Garch = call_prices,
  Put_Garch = put_prices
)

# Print the option prices
print(option_prices)
```

Black-Scholes

```r
# 1. Define Black-Scholes function for Call & Put
black_scholes <- function(S, K, r, T, sigma, type = "call") {
  d1 <- (log(S/K) + (r + sigma^2/2) * T) / (sigma * sqrt(T))
  d2 <- d1 - sigma * sqrt(T)
  if (type == "call") {
    return(S * pnorm(d1) - K * exp(-r * T) * pnorm(d2))
```

```
  } else {
    return(K * exp(-r * T) * pnorm(-d2) - S * pnorm(-d1))
  }
}


# Estimate volatility as the standard deviation of log returns
sigma_bs <- sd(sp500_data$log_returns) * sqrt(252)  # Annualized volatility


# Recalculate BS prices using the adjusted GARCH volatility
bs_put_prices <- sapply(moneyness_levels, function(k) {
  black_scholes(S0, S0*k, r, n_days/252, sigma_bs, type = "put")
})
bs_call_prices <- sapply(moneyness_levels, function(k) {
  black_scholes(S0, S0*k, r, n_days/252,  sigma_bs, type = "call")
})

bs_option_prices <- data.frame(
  Moneyness = c("ITM (0.95)", "ATM (1.00)", "OTM (1.05)"),
  Call_BS = bs_call_prices,
  Put_BS = bs_put_prices
)
print(bs_option_prices)
```

Comparative Pricing Analysis : Black-Scholes vs. GARCH(1,1)

```
# Create Comparison Table
option_data <- data.frame(
  Moneyness = c("ITM (0.95)", "ATM (1.00)", "OTM (1.05)"),
  Call_GARCH = call_prices,
  Call_BS = bs_call_prices,
  Put_GARCH = put_prices,
  Put_BS = bs_put_prices
)

#option_data

# Visualization of the comparison
# Reshape the data into long format for ggplot
option_data_long <- option_data %>%
  pivot_longer(cols = c("Call_GARCH", "Call_BS", "Put_GARCH", "Put_BS"),
               names_to = c("Option_Type", "Model"),
               names_sep = "_",
               values_to = "Price")

# Create the point plot using ggplot
```

```r
ggplot(option_data_long, aes(x = Moneyness, y = Price, color = Model, shape = Option_Type)) +
  geom_point(size = 4, position = position_dodge(width = 0.5)) +
  scale_color_manual(values = c("GARCH" = "skyblue", "BS" = "orange")) +
  scale_shape_manual(values = c("Call" = 16, "Put" = 17)) +  # Solid circle for Call, triangle
  theme_minimal() +
  labs(title = "Comparison of Option Prices: GARCH vs Black-Scholes",
       x = "Moneyness",
       y = "Option Price",
       color = "Model",
       shape = "Option Type") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_y_continuous(labels = scales::comma) +
  theme(legend.position = "top")
```

Rolling Window Analysis

```r
# Load or prepare your data (assuming 'sp500_data' has 'date', 'close', 'log_returns')
# sp500_data <- ...

# Simulation Parameters
n_sim <- 10000      # Number of simulations
n_days <- 252       # Trading days in 1 year
moneyness_levels <- c(0.95, 1, 1.05)  # ITM, ATM, OTM
r <- 0.02           # Risk-free rate
window_years <- 3   # Rolling window size
forecast_horizon <- 1 # Forecast 1 year ahead
roll_step <- 1       # Annual roll
start_year <- 2015
end_year <- 2025

# Initialize results storage
results <- list()

# Rolling Window Implementation
for (current_year in start_year:(end_year - forecast_horizon)) {
  # Define window bounds
  train_start <- as.Date(paste0(current_year - window_years, "-01-01"))
  train_end <- as.Date(paste0(current_year, "-12-31"))
  forecast_start <- train_end + days(1)
  forecast_end <- forecast_start + years(forecast_horizon) - days(1)

  # Subset training data
  train_data <- sp500_data %>%
    filter(date >= train_start & date <= train_end)

  # Skip incomplete windows
  if (nrow(train_data) < 252 * window_years) next
```

```r
# Fit GARCH model
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
  mean.model = list(armaOrder = c(1,1), include.mean=TRUE)
)

garch_fit <- tryCatch(
  ugarchfit(garch_spec, data = train_data$log_returns),
  error = function(e) NULL
)

if (is.null(garch_fit)) next

# Extract parameters from current window
omega <- as.numeric(coef(garch_fit)["omega"])
alpha1 <- as.numeric(coef(garch_fit)["alpha1"])
beta1 <- as.numeric(coef(garch_fit)["beta1"])
sigma2_initial <- tail(sigma(garch_fit)^2, 1)  # Last conditional variance

# Set initial price for current window
S0 <- tail(train_data$close, 1)

# Simulate volatility paths
set.seed(111217)  # Reset seed for reproducibility
garch_sim <- ugarchsim(garch_fit, n.sim = n_days, m.sim = n_sim)
volatility_paths_sd <- sigma(garch_sim) * sqrt(252)  # Annualizing the volatility

# Set initial price for current window
S0 <- head(train_data$close, 1)

# Simulate stock price paths
stock_paths <- matrix(S0, nrow = n_sim, ncol = n_days)
dt <- 1/252

for (i in 1:n_sim) {
  for (t in 2:n_days) {
    sigma_t <- volatility_paths_sd[t, i]
    z <- rnorm(1)
    stock_paths[i, t] <- stock_paths[i, t-1] * exp(
      (r - 0.5 * sigma_t^2) * dt + sigma_t * sqrt(dt) * z
    )
  }
}

# Calculate option prices for current window
bs_vol <- sd(train_data$log_returns) * sqrt(252)
T <- n_days/252
```

```r
  # Black-Scholes prices
  bs_call_prices <- sapply(moneyness_levels, function(k) {
  black_scholes(S0, S0*k, r, T, bs_vol, "call")
})
bs_put_prices <- sapply(moneyness_levels, function(k) {
  black_scholes(S0, S0*k, r, T, bs_vol, "put")
})

# Calculate Monte Carlo prices
mc_call_prices <- sapply(moneyness_levels, function(k) {
  K <- S0 * k
  call_payoffs <- pmax(stock_paths[, n_days] - K, 0)
  exp(-r * T) * mean(call_payoffs)
})

mc_put_prices <- sapply(moneyness_levels, function(k) {
  K <- S0 * k
  put_payoffs <- pmax(K - stock_paths[, n_days], 0)
  exp(-r * T) * mean(put_payoffs)
})

# Store results with proper indexing
results[[as.character(current_year)]] <- data.frame(
  Forecast_Year = current_year + forecast_horizon,
  Moneyness = rep(c("ITM (0.95)", "ATM (1.00)", "OTM (1.05)"), each = 2),
  Type = rep(c("Call", "Put"), 3),
  BS_Price = c(
    bs_call_prices[1], bs_put_prices[1],  # ITM
    bs_call_prices[2], bs_put_prices[2],  # ATM
    bs_call_prices[3], bs_put_prices[3]   # OTM
  ),
  GARCH_Price = c(
    mc_call_prices[1], mc_put_prices[1],  # ITM
    mc_call_prices[2], mc_put_prices[2],  # ATM
    mc_call_prices[3], mc_put_prices[3]   # OTM
  )
)
}

# Combine results
final_results <- bind_rows(results) %>% na.omit()
```

```r
# Ensure final_results exists and is formatted correctly
final_results <- final_results %>%
  mutate(Forecast_Year = as.factor(Forecast_Year))

# Create the plot
```

```r
ggplot(final_results, aes(x = Forecast_Year, group = interaction(Moneyness, Type))) +
  geom_line(aes(y = GARCH_Price, color = Moneyness, linetype = Type), size = 1) +
  geom_point(aes(y = GARCH_Price, color = Moneyness), size = 2) +
  geom_line(aes(y = BS_Price, color = Moneyness, linetype = Type), size = 1, alpha = 0.5) +
  geom_point(aes(y = BS_Price, color = Moneyness), size = 2, shape = 4, alpha = 0.7) +

  labs(
    title = "Evolution of Option Prices Over Time",
    subtitle = "Comparison of GARCH-based and Black-Scholes Option Pricing Models",
    x = "Year",
    y = "Option Price",
    color = "Moneyness",
    linetype = "Option Type"
  ) +

  theme_minimal() +
  theme(
    plot.title = element_text(size = 16, face = "bold"),
    plot.subtitle = element_text(size = 12, face = "italic"),
    axis.text.x = element_text(angle = 45, hjust = 1),
    legend.position = "bottom"
  )
```

Conclusion