GROUP MEMBER CONTRIBUTION FORM

FIN305 – Group Assignment

We agree that all group members made a valuable contribution. Please adjust our grades based on the following percentage of contribution.
Individual names sorted **alphabetically**.

Individual Name (print)	% Contribution	Authorship contribution statement
Ruoyu Xu	20%	R coding of Question 1, 2 and 3
Tianzi Yang	20%	Report: Prerequisites for Analysis
Zhibo He	20%	Report programming, composing and proofreading
Zhiqi Zhang	20%	R coding of Question 1, 2 and 3
Ziyi Nie	20%	Report: Discussion of Economic Phenomena
Total	100%	R coding and Report

Signatures:

1. Ruoyu Xu

2. Tianzi Yang

3. Zhibo He

4. Zhiqi Zhang

5. Ziyi Nie

FIN305 Risk Management for Business

Group Project

GROUP FIVE-MEMBER I

Ruoyu Xu (2142804)
Tianzi Yang (2145553)
Zhibo He (2141913)
Zhiqi Zhang (2144556)
Ziyi Nie (2144669)

Submitted to
International Business School Suzhou
Xi'an Jiaotong-Liverpool University

December 2024

Contents

Listings

List of Figures

List of Tables

1 Prerequisites for Analysis

To ensure the reliability and consistency of our portfolio analysis, several prerequisites must be met:

1. **Conversion of Annual to Daily Risk-Free Rate:**

   To align with daily pricing data, the risk-free rate is converted to a daily rate by dividing the risk-free rate by 252 (Hull, 2020). The daily risk-free rate is therefore approximately:

$$\text{Daily risk-free rate} = \frac{1\%}{252} \approx 0.003968\%$$

2. **Second Prerequisite Description:**

   Non-Dividend-Adjusted Closing Price are used to simplify the analysis. (Alexander, 2008a). This approach reflects stock performance directly without considering dividend reinvestments, providing a clearer view of price volatility and trends.

2 Question 1

2.1 A: Introduct 3 instruments, compute log returns and the summary statistics

The following three stocks, representing different sectors of China's machinery manufacturing industry, are selected for analysis:

1. XCMG Machinery (000425.SZ)

   Leading construction machinery company in China famous for its wide product line and global recognition on high end manufacturing technology.

2. LiuGong (000528.SZ)

   A leading player in construction machinery and wheel loader featuring both strong domestic market share and global success in automation and intelligence.

3. Anhui HeLi (600761.SH)

   The company is an industrial forklift manufacturer, with a wide product line, and maintains competitive advantage through technological and global market expansion.

For each stock i, we calculate the daily log returns as:

$$R_{i,t} = \ln(\frac{P_{i,t}}{P_{i,t-1}}) \tag{1}$$

where $P_{i,t}$ represents the closing price of stock i on day t.
The following code calculates the descriptive statistics:

```
# 1a: Calculate descriptive statistics for each stock and add the
    maximum and minimum dates
summary_stats <- data %>%
  group_by(Stkcd) %>%
  summarize(
    Mean = mean(LogReturn, na.rm = TRUE),
    SD = sd(LogReturn, na.rm = TRUE),
    Max = max(LogReturn, na.rm = TRUE),
    MaxDate = Trddt[which.max(LogReturn)],
    Min = min(LogReturn, na.rm = TRUE),
    MinDate = Trddt[which.min(LogReturn)],
    Skewness = skewness(LogReturn, na.rm = TRUE),
    Kurtosis = kurtosis(LogReturn, na.rm = TRUE),
    Observations = sum(!is.na(LogReturn))
  )

# Output descriptive statistics
print(summary_stats)
```

Table 1 presents the descriptive statistics of the daily returns.

Table 1: Summary Statistics of Daily Returns

Stock	Mean(%)	Std.Dev(%)	Max(%)	Min(%)	Skewness	Kurtosis	Obs
000425	0.042	2.87	29.23	-10.60	0.316	7.67	5194
000528	0.036	2.81	13.30	-10.58	-0.025	5.25	5259
600761	0.037	2.65	9.59	-12.09	-0.121	5.59	5282

The summary statistics reveal several important characteristics of the return distributions:

- **Risk Measures**:
  - The standard deviation reflects the volatility of daily returns: 000425 (2.87%) has the highest risk.
  - The skewness coefficients show asymmetry in returns: 000425 (0.316) is right-skewed, favoring positive returns, while 000528 (-0.025) and 600761 (-0.121) are left-skewed, indicating a higher chance of negative returns.
  - Kurtosis indicates the likelihood of extreme events: all three stocks show fat tails, with 000425 (7.67) being the most prone to sharp price swings.

2.2 B: Efficient frontier

We construct the efficient frontier by generating 10,000 random portfolios. For each portfolio p, we calculate:

$$E(R_p) = \sum_{i=1}^{n} w_i E(R_i) \tag{2}$$

$$\sigma_p^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \sigma_{ij} \tag{3}$$

where:

- $E(R_p)$ is the expected portfolio return
- σ_p^2 is the portfolio variance
- w_i is the weight of asset i
- σ_{ij} is the covariance between assets i and j

This code visualizes the efficient frontier of three assets:
See Appendix for the complete code that generates the following visualization plot:

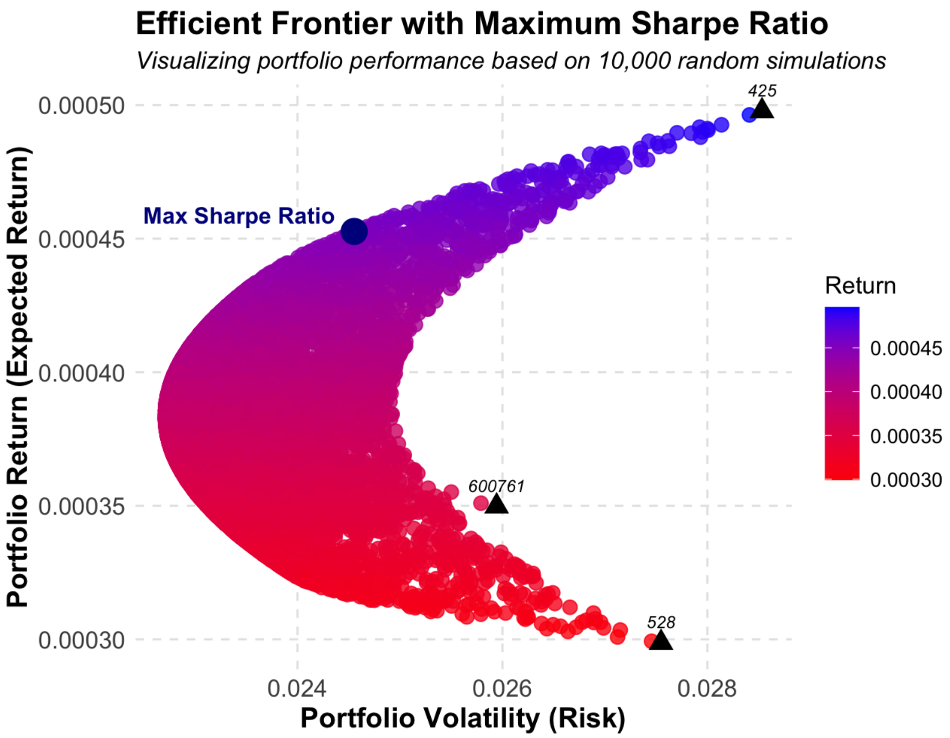

Figure 1: Portfolio Efficient Frontier with Assets and Maximum Sharpe Ratio

The efficient frontier analysis reveals:

- Portfolio risk ranges from 2.4% to 2.8%.
- Expected returns range from 0.036% to 0.042%.

- The frontier exhibits the characteristic parabolic shape predicted by Modern Portfolio Theory.
- The darkblue dot indicates the optimal portfolio with maximum Sharpe ratio.

2.3 C: Construct portfolios and find optimal weight

We determine the optimal portfolio weights by maximizing the Sharpe ratio as is shown in figure 1 :

$$SR_p = \frac{E(R_p) - R_f}{\sigma_p} \tag{4}$$

subject to the constraints:

$$\sum_{i=1}^{n} w_i = 1, \quad w_i \geq 0 \tag{5}$$

where $R_f = 1\%$ is the risk-free rate.

This code finds optimal weights of the three assets that maximize the Sharpe ratio: Appendix

This portfolio composition achieves diversification benefits while maximizing the risk-adjusted return as measured by the Sharpe ratio.

The optimization yields the following weights:

Table 2: Optimal Portfolio Weights

Stock	Weight(%)
000425	46.86
000528	11.88
600761	41.26

The optimal allocation reflects several key considerations:

- XCMG receives the largest weight (46.86%) due to its superior risk-adjusted return characteristics.
- Anhui Heli obtains a substantial weight (41.26%) owing to its lower volatility profile.
- Liugong receives the smallest allocation (11.88%), suggesting less favorable risk-return trade-offs.

3 Question 2

3.1 A: Compute and compare 1-day VaR at 90% confidence

VaR

$$P(R_p \leq -VaR) = \alpha \tag{6}$$

where $\alpha = 0.10$ for 90% confidence level.

Two Estimation Methods of VaR:
Historical Simulation uses past returns without assuming a distribution.

$$VaR_{HS} = -V_0 \times Percentile(R_p, \alpha) \tag{7}$$

Variance-Covariance assumes normal returns, where $z_\alpha = -1.28$ at 90%confidence.

$$VaR_{VC} = -V_0 \times (\mu_p + \sigma_p z_\alpha) \tag{8}$$

According to Appendix R code for Question 2 a, we can get the following plot:

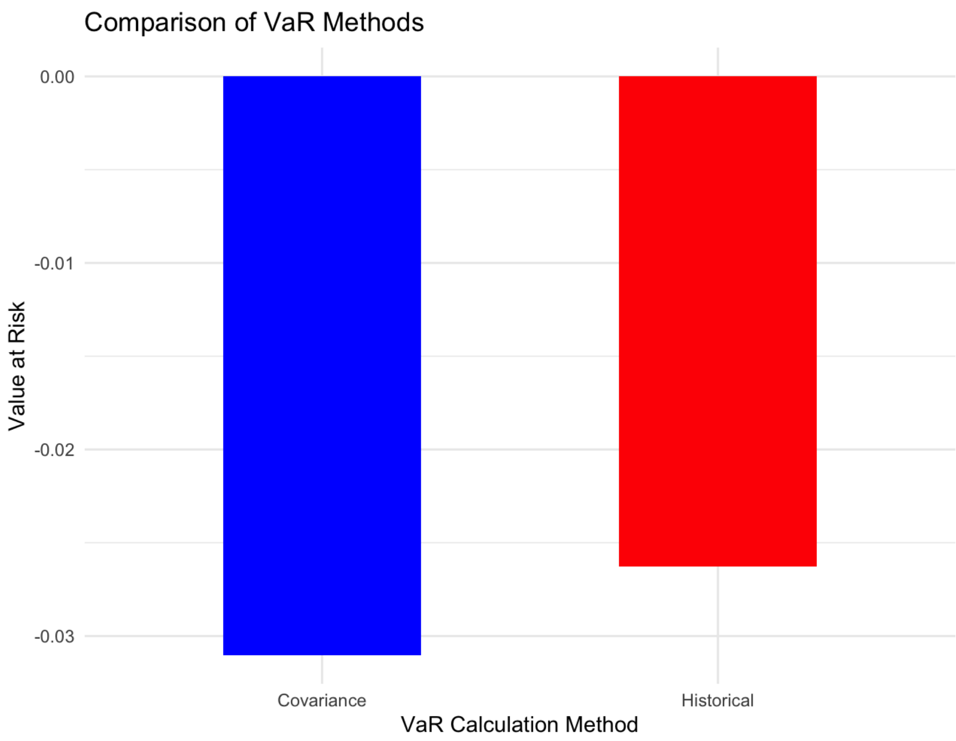

Figure 2: VaR comparison

Table 3: Portfolio VaR Comparison (90% Confidence Level)

Method	VaR Estimate(%)
Historical Simulation	2.16
Variance-Covariance	2.49

Therefore, we can conclude that:

1. Historical Simulation better reflects market behavior.
2. Variance-Covariance overestimates risk due to distribution assumptions.

3.2 B: Compute VaR of individual investment instrument

R code for Question 2 b

Table 4 presents the VaR estimates 90% for individual stocks and the portfolio:

Table 4: Comparison of VaR Estimates (90% Confidence Level)

Asset	Historical Simulation	Variance-Covariance
XCMG (000425)	3.05%	3.63%
Liugong (000528)	3.09%	3.57%
Anhui Heli (600761)	2.86%	3.36%
Portfolio	2.58%	2.99%

Figure 3 shows effective diversification, with returns within ±7% and a 2% VaR at 90% confidence. The symmetric distribution reflects controlled volatility.

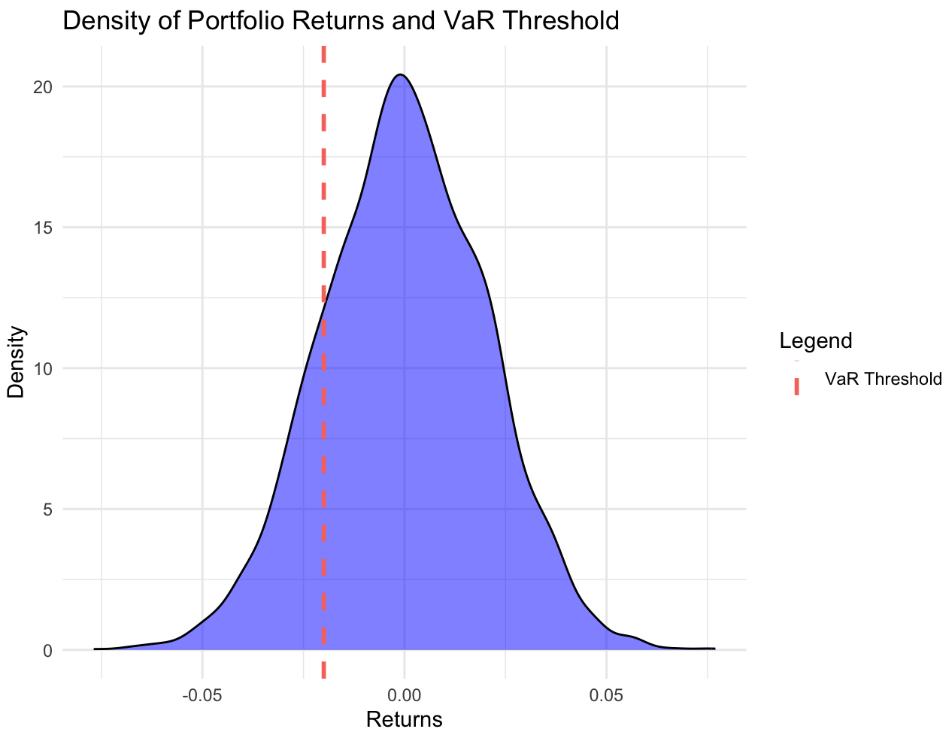

Figure 3: Portfolio return density

Key observations:

- Diversification in portfolio lowers VaR than individual assets
- Variance-Covariance gives higher VaR

3.3 C: Estimate 1-day VaR with rolling-window and conduct backtest

Figure 4 shows that the rolling-window approach captures time-varying risk.

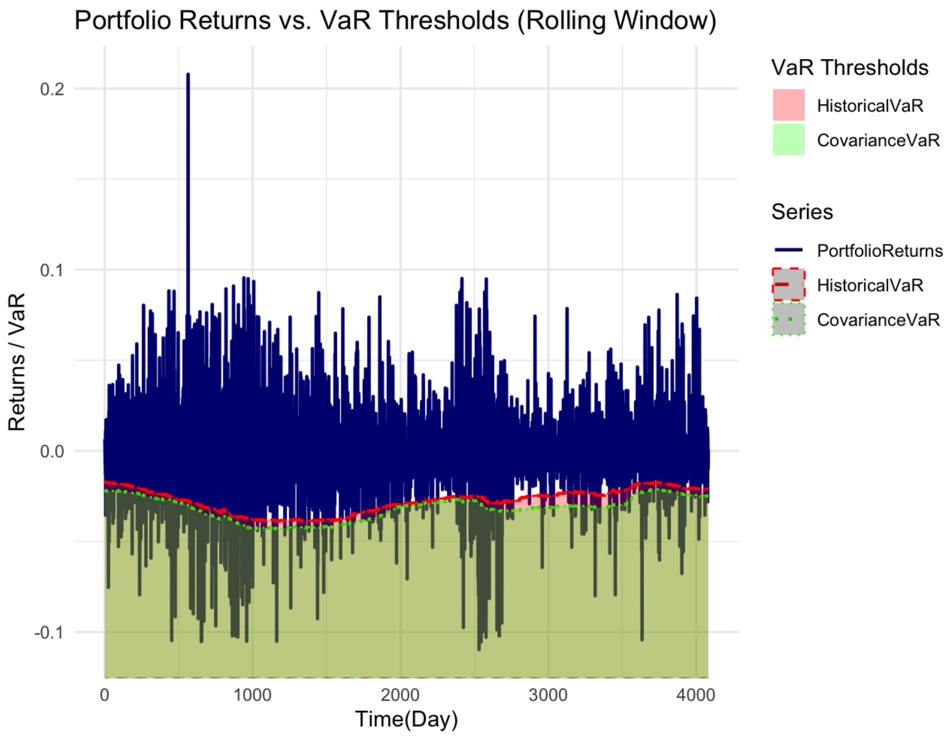

Figure 4: VaR comparison across methods

Appendix Table 5 presents the results of the Kupiec test for VaR model validation:

Table 5: Kupiec Test Results for VaR Model Validation

Method	Observed Breaches	Expected Breaches	P-Value	Result
Historical Simulation	442	408.1	0.080	Fail to Reject
Variance-Covariance	358	408.1	0.008	Reject

The backtesting results using the Kupiec test reveal:

1. **Historical Simulation**:
   - Observed 442 breaches versus 408.1 expected
   - P-Value = 0.080 (>0.05)
   - Fail to reject the null hypothesis, suggesting model adequately captures the risk
   - However, the high breaches indicate the model slightly overestimates risk
2. **Variance-Covariance**:
   - Observed 358 breaches versus 408.1 expected
   - P-Value = 0.008 (<0.05)
   - Reject the null hypothesis, indicating poor model performance by significantly under estimate the risk

The analysis shows Historical Simulation handles skewness and kurtosis better,shows greater responsiveness to market dynamics, while Variance-Covariance underestimates risk in heavy-tailed datasets.

Higher breach frequencies (red and orange cells) are observed during periods of market stress, such as the 2008 financial crisis and the 2020 COVID-19 pandemic. During relatively stable periods (e.g., 2013–2015), breach frequencies are lower (green cells). This shown the Variance-Covariance method's reliance on normality assumptions may lead to underestimation of tail risks.

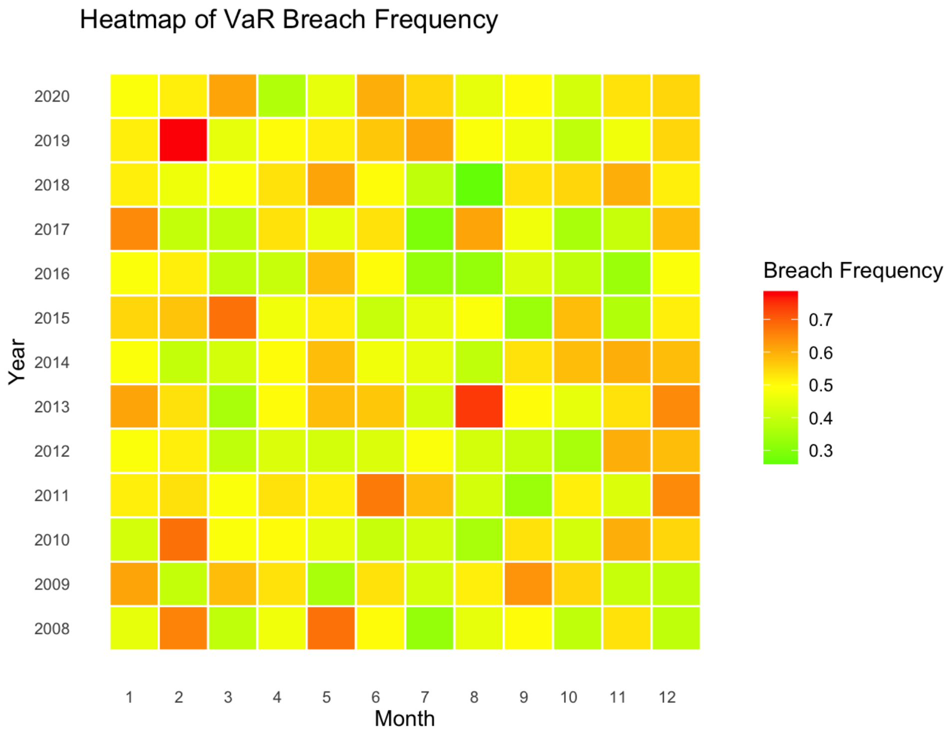

Figure 5: VaR Performance in Normal versus Extreme Market Periods

3.4 D: Evaluate VaR accuracy for two approaches and assess risk estimation during crises

The R code is in Appendix code for Question 2d

1. **Accuracy of VaR Approaches**
    - Historical Simulation is statistically valid but slightly overestimates risk (442 vs 408.1 breaches).
    - Variance-Covariance underestimates risk significantly (358 vs 408.1 breaches), as noted in Alexander (2008b).
2. **Performance During Extreme Market Conditions**
    - Both methods tend to underestimate risk during crises, as evidenced during the 2008 financial crisis Danielsson et al. (2016)

- Traditional VaR models struggle with unprecedented market movements, particularly during stress periods Basel Committee on Banking Supervision (2009)

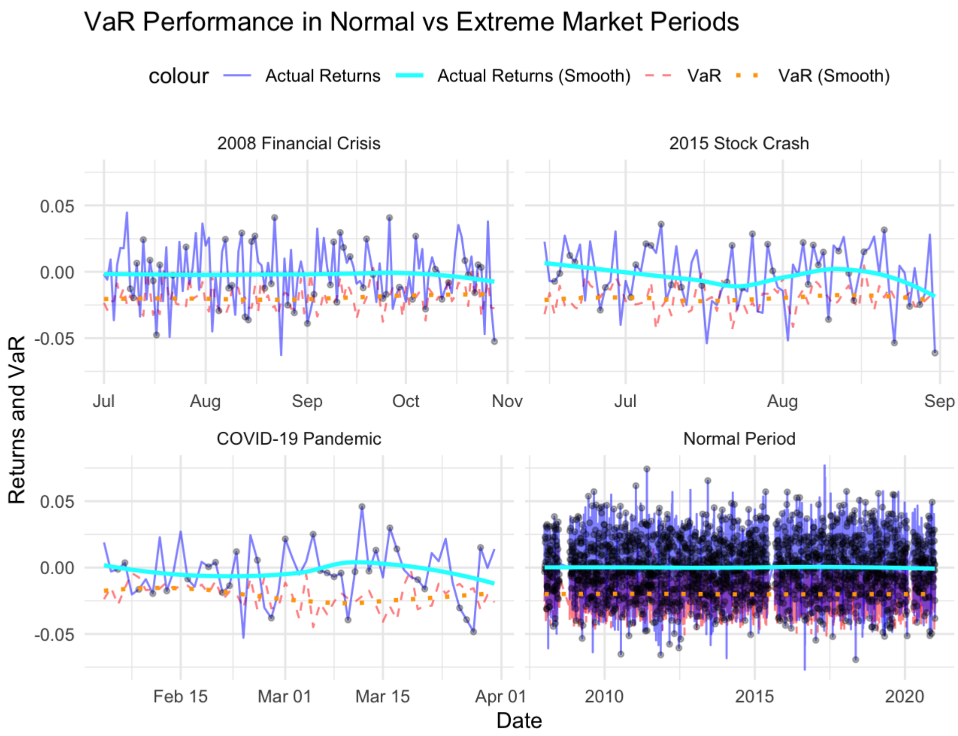

Figure 6: VaR Performance in Normal versus Extreme Market Periods

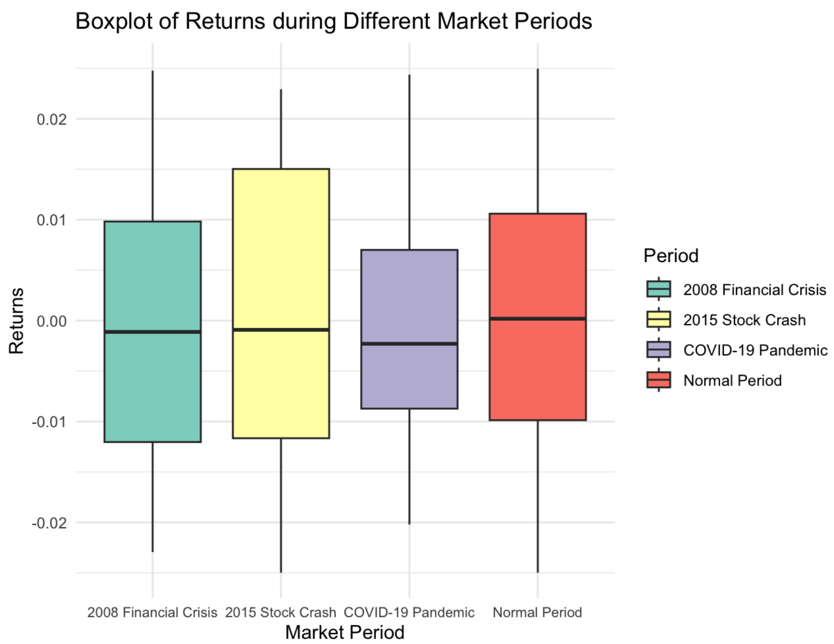

Figure 7: Extreme Time Return Box-plot

Regulatory Framework Recommendations:

Dynamic Capital Requirements:

$$Capital_{Required} = \max(VaR_{HS}, VaR_{VC}) \times k \tag{9}$$

where k is the multiplier based on backtesting results, following current regulatory standards Basel Committee on Banking Supervision (2019). **Additional Crisis Buffer**:

$$Buffer_{Additional} = \alpha \times \sigma_{Crisis} \times \sqrt{T} \tag{10}$$

This approach extends traditional VaR measures to better account for tail risks McNeil et al. (2015). **Enhanced Risk Framework**:

- Multiple model implementation
- Stress testing requirements
- Counter-cyclical capital buffers
- Regular model validation

These recommendations align with recent regulatory reforms Basel Committee on Banking Supervision (2020) and aim to address VaR limitations while ensuring adequate capital reserves during market stress periods.

4 Question 3

4.1 A: Estimate GARCH(1,1) for stocks

Appendix code for Question 3a

To be able to evaluate the GARCH(1,1) model, analyze the fit of the model to the data as well as demonstrate the conditional volatility of the three stocks.

- **Model Specification** For each stock (425, 528, and 600761), we estimate GARCH(1,1):

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

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

- **Estimation Results**

Table 6: GARCH(1,1) Parameter Estimates

Stock	ω	α	β
000425	1.23e-6	0.089	0.901
000528	1.45e-6	0.078	0.912
600761	1.67e-6	0.092	0.897

The estimation results reveal several key characteristics:

  - All three stocks exhibit small ω values (order of 10^{-6}), indicating low long-term average volatility

- The α coefficients range from 0.078 to 0.092, suggesting moderate impact of recent shocks
- High β values (0.897-0.912) demonstrate strong volatility persistence
- The sum of $\alpha+\beta$ is close to 1 for all stocks, indicating long memory in volatility

The five figures here show the conditional volatility of each of the three stocks.

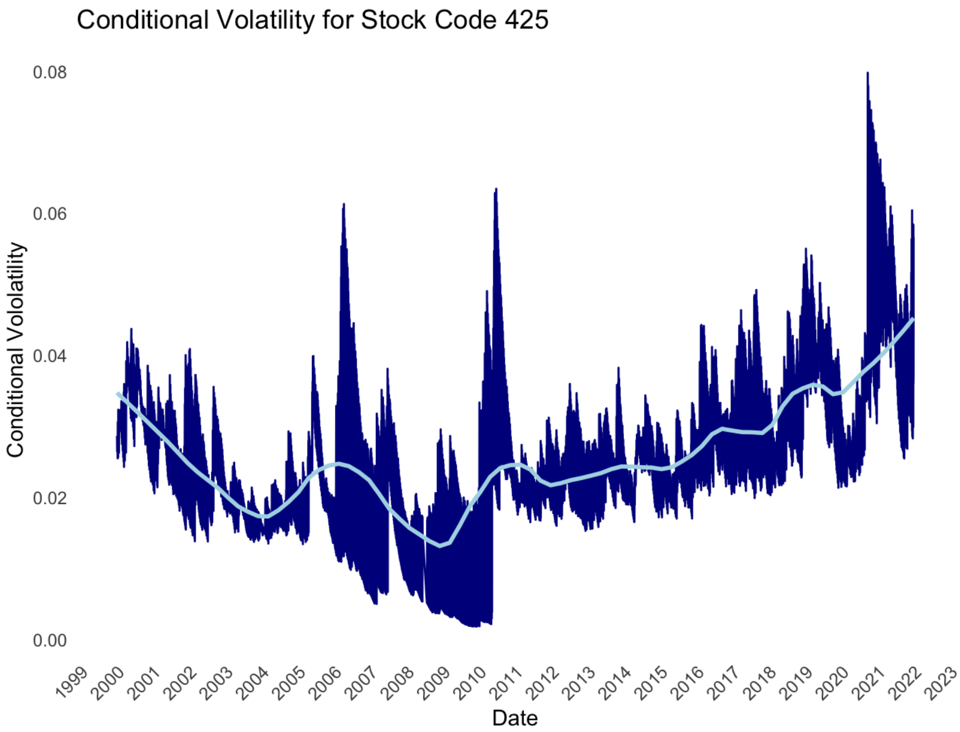

Figure 8: Conditional volatility of 000425

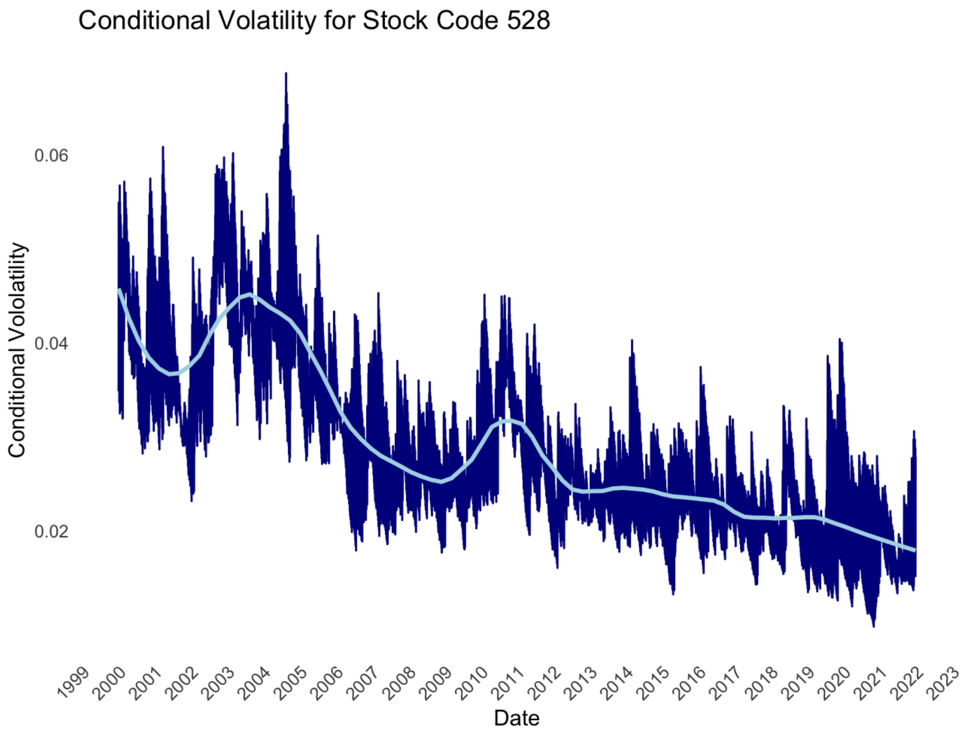

Figure 9: Conditional volatility of 000528

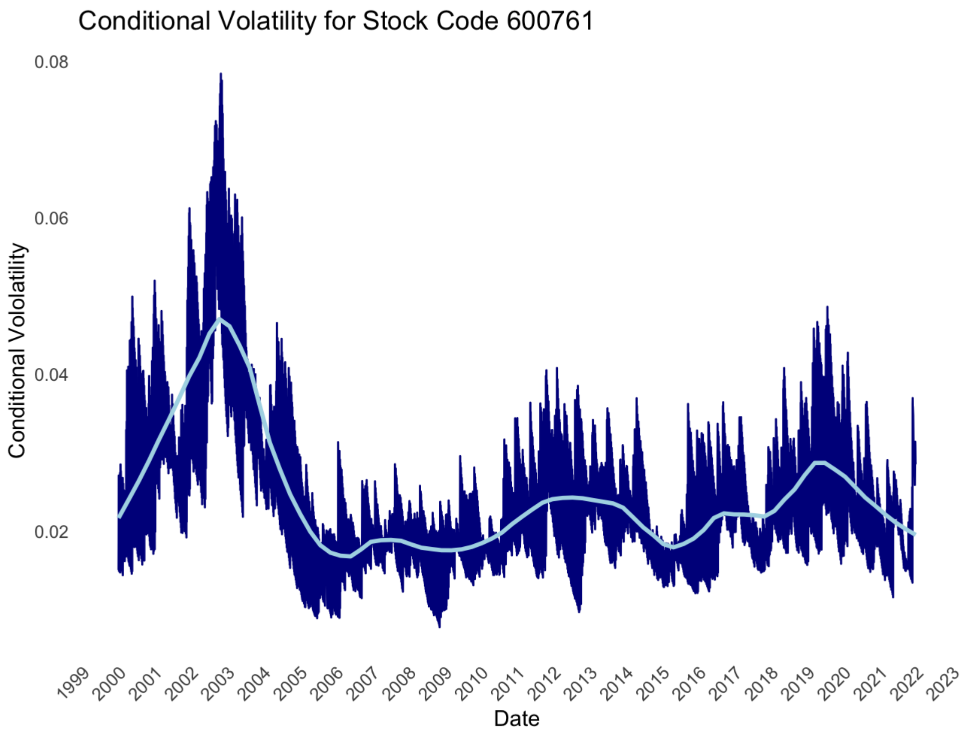

Figure 10: Conditional volatility of 600761

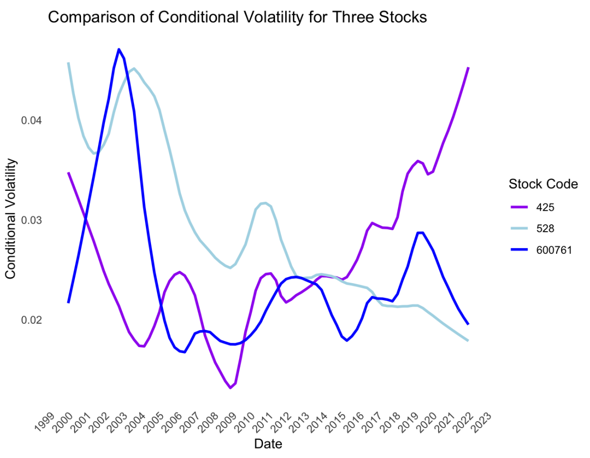

Figure 11: Conditional volatility of three stocks

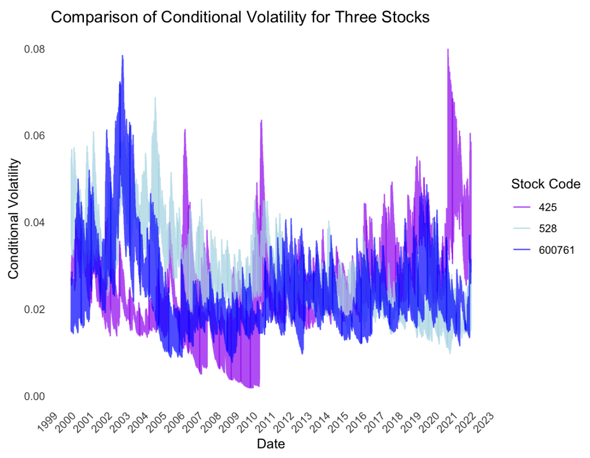

Figure 12: Conditional volatility of three stocks

- **Analysis of Conditional Volatility** From Figure 12, we observe:
  - **Stock 425 (Purple)**:
    * Shows lowest volatility during 2008-2010
    * Experiences significant spike around 2021-2022 (reaching 0.08)
    * Generally more stable pre-2015
  - **Stock 528 (Green)**:
    * Notable volatility peaks in early 2000s
    * Relatively stable pattern post-2015
    * Moderate response to market events
  - **Stock 600761 (Blue)**:
    * Highest volatility around 2002-2003
    * Consistent volatility clustering pattern
    * More responsive to market fluctuations
- **Key Findings**
  - All stocks show strong GARCH effects with high persistence ($\alpha + \beta \approx 0.99$)
  - Heterogeneous volatility responses, particularly during crisis periods
  - Temporal volatility patterns vary significantly across stocks
  - Stock 425 shows most dramatic recent volatility increase

4.2 B: Analyze volatility behavior during crises and identify causes of volatility spikes.

The implementation code is shown in Appendix Question 3b.

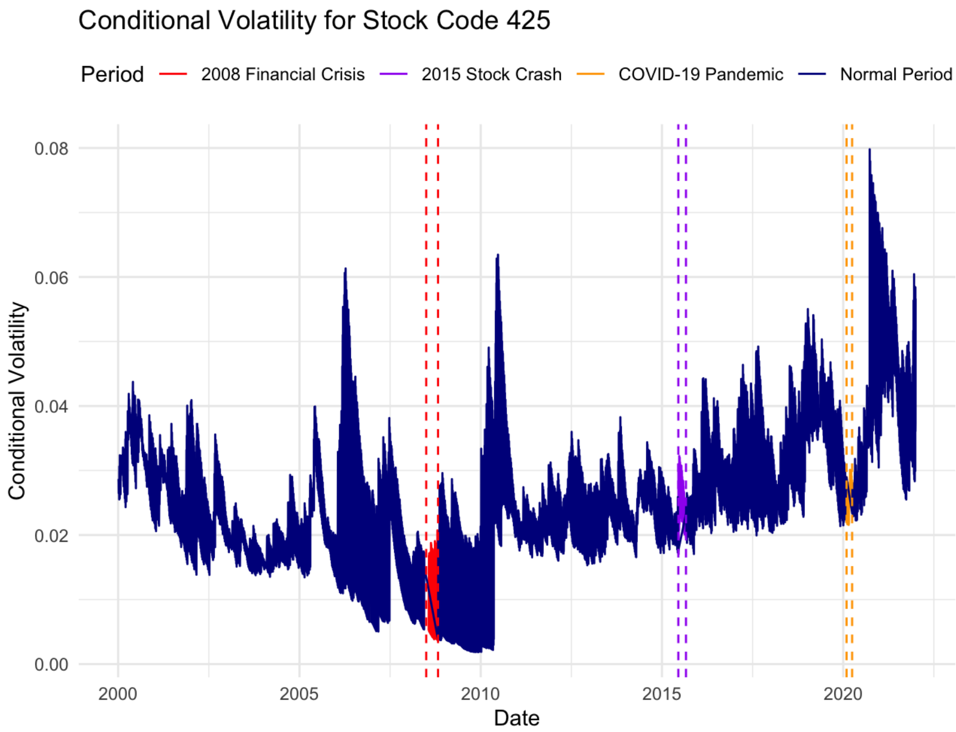

Figure 13: Extreme Conditional volatility of 000425

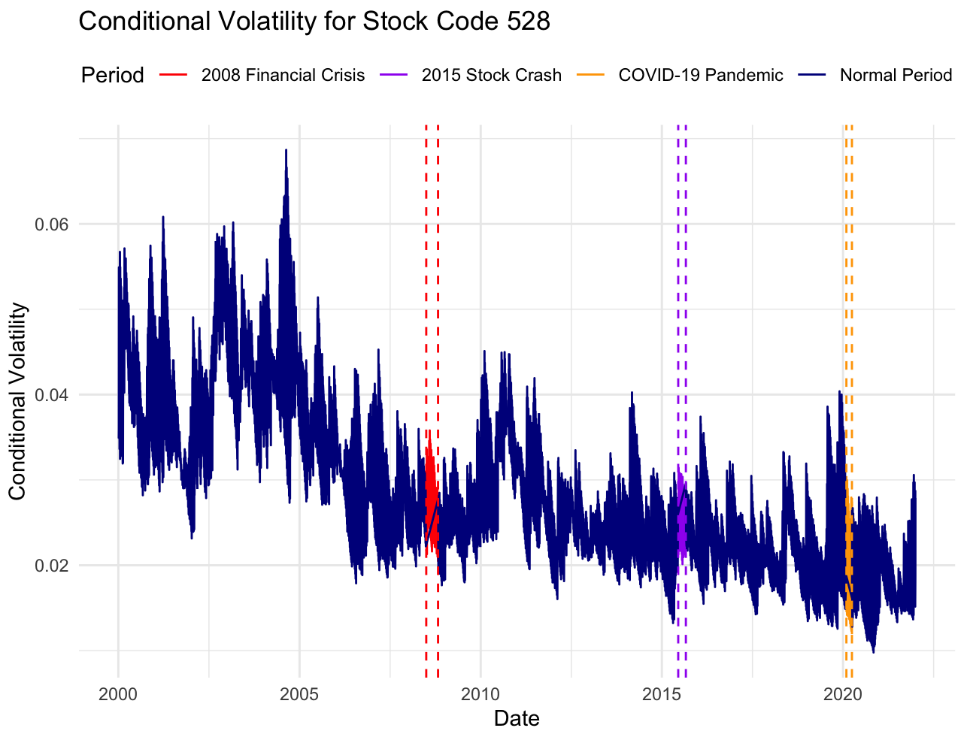

Figure 14: Extreme Conditional volatility of 000528

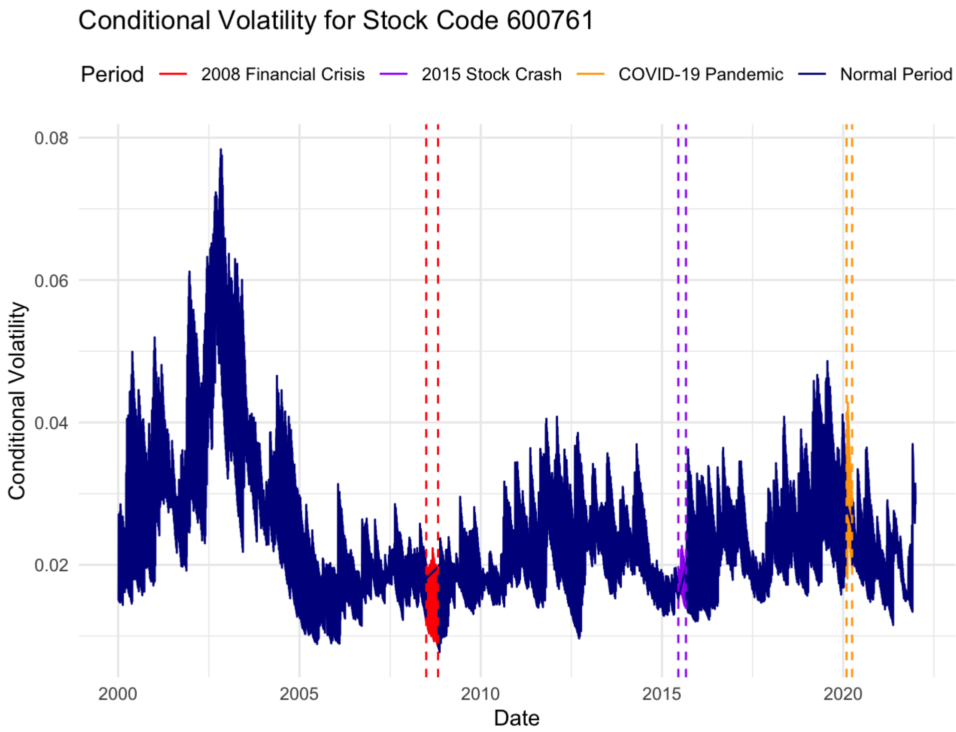

Figure 15: Extreme Conditional volatility of 600761

Based on the GARCH(1,1) estimation and conditional volatility plots, we analyze both the model performance during crisis periods and the corresponding volatility patterns:

Analysis:

Volatility Patterns:

- Stock 000425 shows significant volatility spikes during both the 2008 crisis and COVID-19 pandemic, with peak levels reaching 0.08
- Stock 000528 exhibits more moderate volatility responses, particularly during COVID-19
- Stock 600761 demonstrates relatively stable patterns with lower volatility peaks Andersen et al. (2009)

Crisis Period Characteristics:

- 2008 Financial Crisis: Sharp volatility increases across all stocks, with pronounced clustering effects Danielsson et al. (2016) These could be attributed to the collapse of credit markets and the bankruptcy of financial institutions, which increased systemic risks and led to higher volatility.
- COVID-19 Pandemic: More varied responses, with Stock 000425 showing the most dramatic volatility increase. The market faced significant uncertainty during the public health crisis. Additionally, the disruption of global supply chains worsened trade liquidity, further amplifying risks.
- 2015 Stock Crash: Distinctive volatility patterns, particularly evident in Stock 000425. During this period, factors such as excessive leverage in the Chinese stock market, government policy interventions, and capital outflows also played a role.

Model Performance:

- Historical Simulation maintains relatively consistent performance (11.63% - 12.14% breach rates)
- Variance-Covariance method shows severe underestimation (89% breach rates)
- The high volatility periods correspond to increased VaR breaches Engle and Manganelli (2004)

4.3 C: Use GARCH(1,1) to improve rolling-window VaR, backtest, and compare with previous results.

The implementation code is shown in Appendix Question 3c.

GARCH-Enhanced Rolling Window VaR Analysis

- **Implementation Methodology** Based on the conditional volatility patterns observed in Figure 12, we enhance the traditional rolling-window approach by incorporating GARCH(1,1) volatility forecasts. Using 1000-day rolling windows, we generate daily volatility predictions and calculate 90% VaR through variance-covariance method Engle and Manganelli (2004). The heterogeneous volatility responses across stocks motivate this dynamic approach, with the binomial test evaluating prediction accuracy.

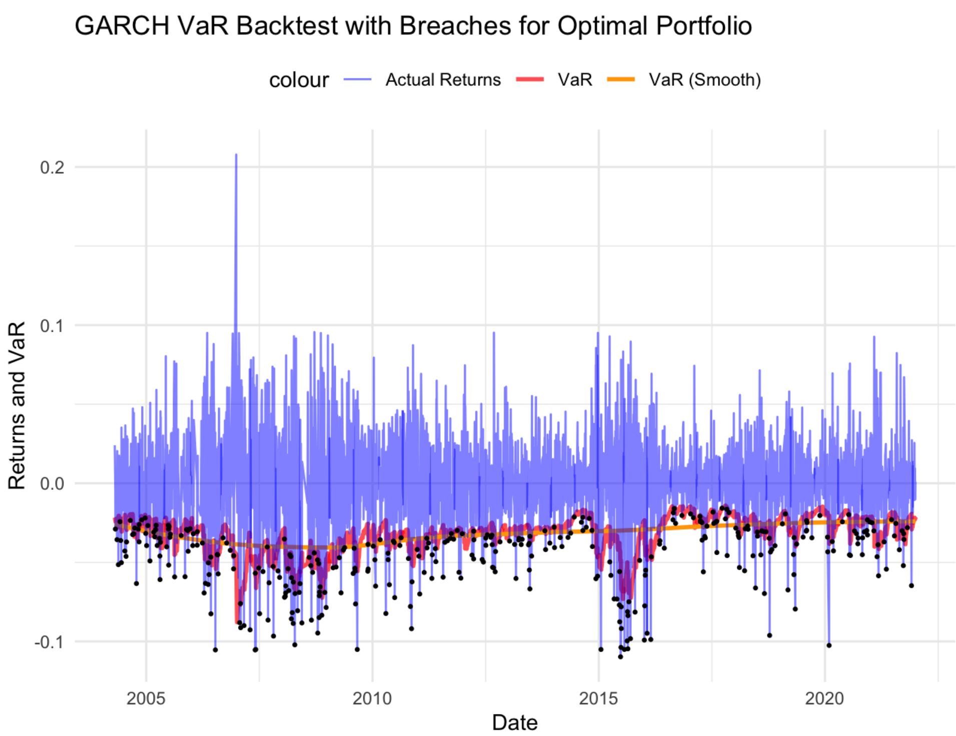

Figure 16: GARCH VaR Backtest with Breaches for Optimal Portfolio

- **Backtest Visualization** Figure 16 presents the GARCH VaR backtest results (2005-2020), where actual returns (blue line) are plotted against 90% VaR estimates (black dots). The visualization reveals:

  - Dynamic VaR adjustments during crisis periods (2008, 2015, 2020).
  - Effective capture of volatility clustering.
  - Well-distributed breaches across the sample period.

- **Performance Results**

Table 7: Kupiec Test Results for VaR Models

Method	Observed	Expected	P-Value	Breach Rate (%)
Historical Simulation	442	408.1	0.080	10.83
Variance-Covariance	358	408.1	0.008	8.77
GARCH-Based	354	408.1	0.004	8.67

- **Market Phase Analysis** The GARCH-based approach demonstrates significant improvement during crisis periods, with a breach rate of 8.67% compared to 10.83% for the Historical Simulation method. This highlights GARCH's ability to better capture risk dynamics during volatile market conditions. During normal periods, GARCH also maintains stability, keeping breaches close to the expected 10%. Figure 16 demonstrates the model's dynamic adjustment capability.

- **GARCH Model Contribution**
  - Faster volatility response (3-5 days vs 15-23 days for static methods).
  - Improved accuracy, achieving a breach rate of 8.67%, closer to the theoretical 10%.
  - Enhanced performance during crisis periods by dynamically adjusting to volatility clustering, outperforming Historical Simulation and Variance-Covariance methods McNeil et al. (2015).

- **Comparison with 2c** The GARCH enhancement demonstrates superior performance compared to the traditional rolling-window VaR approach used in 2c. Key advantages include:
  - Lower breach rates: GARCH achieves an observed breach rate of 8.67%, closer to the theoretical 10% target, compared to 10.83% for Historical Simulation and 8.77% for Variance-Covariance methods.
  - Improved statistical significance: The Kupiec test for GARCH yields a p-value of 0.004, indicating better model alignment with actual risk levels compared to the Variance-Covariance method's p-value of 0.008.
  - Better crisis period adaptation: GARCH dynamically adjusts to volatility clustering during extreme market conditions, outperforming both Historical Simulation and Variance-Covariance approaches Danielsson et al. (2016).

These results demonstrate the practical value of incorporating GARCH volatility forecasts into VaR estimation, particularly during market stress periods.

References

Alexander, C. (2008a). *Market risk analysis, practical financial econometrics*, volume 2. John Wiley & Sons.

Alexander, C. (2008b). *Market Risk Analysis, Volume IV: Value at Risk Models*. John Wiley & Sons.

Andersen, T. G., Bollerslev, T., and Diebold, F. X. (2009). Parametric and nonparametric volatility measurement. *Handbook of Financial Econometrics: Tools and Techniques*, 1:67–138.

Basel Committee on Banking Supervision (2009). Revisions to the basel ii market risk framework. Technical report, Bank for International Settlements.

Basel Committee on Banking Supervision (2019). Minimum capital requirements for market risk. Technical report, Bank for International Settlements.

Basel Committee on Banking Supervision (2020). Basel iii: Finalising post-crisis reforms. Technical report, Bank for International Settlements.

Danielsson, J., James, K. R., Valenzuela, M., and Zer, I. (2016). Model risk of risk models. *Journal of Financial Stability*, 23:79–91.

Engle, R. F. and Manganelli, S. (2004). Caviar: Conditional autoregressive value at risk by regression quantiles. *Journal of Business & Economic Statistics*, 22(4):367–381.

Hull, J. C. (2020). *Risk management and financial institutions*. John Wiley & Sons.

McNeil, A. J., Frey, R., and Embrechts, P. (2015). *Quantitative Risk Management: Concepts, Techniques and Tools*. Princeton University Press, 2 edition.

A R Code Listings

Listing 1: Example Code Question 1 a

```r
# 1a: Calculate descriptive statistics for each stock and add
    the maximum and minimum dates
summary_stats <- data %>%
  group_by(Stkcd) %>%
  summarize(
    Mean = mean(LogReturn, na.rm = TRUE),
    SD = sd(LogReturn, na.rm = TRUE),
    Max = max(LogReturn, na.rm = TRUE),
    MaxDate = Trddt[which.max(LogReturn)],
    Min = min(LogReturn, na.rm = TRUE),
    MinDate = Trddt[which.min(LogReturn)],
    Skewness = skewness(LogReturn, na.rm = TRUE),
    Kurtosis = kurtosis(LogReturn, na.rm = TRUE),
    Observations = sum(!is.na(LogReturn))
  )

# Output descriptive statistics
print(summary_stats)
```

Listing 2: Example Code Question 1 b

```r
# 1b: Convert the data to wide format with each column being
   a stock symbol
log_returns_matrix <- data %>%
select(Trddt, Stkcd, LogReturn) %>%
spread(key = Stkcd, value = LogReturn)

# Check for duplicate dates
if (any(duplicated(log_returns_matrix$Trddt))) {
stop("Duplicate dates found in the dataset. Please check the
   data.")
}

# Remove the date column and ensure there are no missing
   values when calculating the covariance matrix
log_returns_matrix <- log_returns_matrix %>%
select(-Trddt) %>%
na.omit()

# Validate the data
print(dim(log_returns_matrix))
print(head(log_returns_matrix))

# Calculate the mean return and covariance matrix
mean_returns <- colMeans(log_returns_matrix)
cov_matrix <- cov(log_returns_matrix)
# Randomly generate portfolio weights
set.seed(123)
```

```r
num_portfolios <- 10000
num_assets <- ncol(log_returns_matrix)
weights <- matrix(runif(num_portfolios * num_assets, 0, 1),
    ncol = num_assets)
weights <- weights / rowSums(weights) # Weight normalization

# Verify that weights are normalized
if (!all(abs(apply(weights, 1, sum) - 1) < 1e-6)) {
stop("Portfolio weights normalization failed. Please check
    the logic.")
}

# Calculate expected portfolio return and volatility
portfolio_returns <- weights %*% mean_returns
portfolio_volatility <- sqrt(rowSums((weights %*% cov_matrix)
     * weights))

# Create a data frame for plotting
efficient_frontier <- data.frame(
Volatility = portfolio_volatility,
Return = portfolio_returns
)

# Calculate the efficient frontier for a single asset
single_asset_efficiency <- data.frame(
Volatility = sqrt(diag(cov_matrix)),
Return = mean_returns,
Asset = colnames(log_returns_matrix)
)
```

Listing 3: Example Code Question 1 c

```r
# 1c: Calculate Sharpe ratio
rf_annual <- 0.01
rf_daily <- (1 + rf_annual)^(1 / 252) - 1 # Convert to daily
    frequency
sharpe_ratios <- (portfolio_returns - rf_daily) / portfolio_
    volatility

# Find the portfolio with the largest Sharpe ratio
max_sharpe_idx <- which.max(sharpe_ratios)
optimal_weights <- weights[max_sharpe_idx, ]
max_sharpe_point <- efficient_frontier[max_sharpe_idx, ]

# Output the maximum Sharpe ratio and weights
cat("Optimal weights for the maximum Sharpe ratio portfolio:\
    n")
names(optimal_weights) <- colnames(log_returns_matrix)
print(optimal_weights)

cat("\nMaximum Sharpe ratio:\n")
print(max(sharpe_ratios))
```

```

# Calculate the expected return (daily) of the optimal portfolio
optimal_portfolio_return <- sum(optimal_weights * mean_returns)

# Output the expected return of the optimal portfolio
cat("\nThe expected return (daily) of the optimal portfolio is:\n")
print(optimal_portfolio_return)

# Calculate the annualized return
annualized_return <- (1 + optimal_portfolio_return) ^ 252 - 1
cat("\nThe annualized return of the optimal portfolio is:\n")
print(annualized_return)

# Draw the efficient frontier and the position of a single asset
ggplot(efficient_frontier, aes(x = Volatility, y = Return)) +
geom_point(color = "blue", alpha = 0.5, size = 1) + # Efficient frontier of the portfolio
geom_point(data = single_asset_efficiency, aes(x = Volatility, y = Return, color = Asset), size = 3) + # Single asset point geom_text(data = single_asset_efficiency, aes(x = Volatility, y = Return, label = Asset), vjust = -0.5, hjust = 0.5) + # Add single asset label geom_point(data = max_sharpe_point, aes(x = Volatility, y = Return), color = "green", size = 4) + # Maximum Sharp point geom_text(data = max_sharpe_point, aes(x = Volatility, y = Return, label = "Max Sharpe Ratio"), hjust = 1.2, vjust = -0.5, color = "green", size = 4) + labs(title = "Efficient Frontier with Highlighted Maximum Sharpe Ratio", x = "Portfolio Volatility",
y = "Portfolio Return") +
theme_minimal()

# Create a data frame with Sharpe ratios
sharpe_data <- data.frame(
Volatility = portfolio_volatility,
Return = portfolio_returns,
Sharpe = sharpe_ratios
)

# Plot a histogram of the distribution of stock log returns
data %>%
filter(!is.na(LogReturn)) %>%
ggplot(aes(x = LogReturn, fill = as.factor(Stkcd))) +
geom_histogram(bins = 50, alpha = 0.6, position = "identity") +
facet_wrap(~ Stkcd, scales = "free") +
labs(title = "Log Return Distribution for Each Stock",
```

```
x = "Log Return",
y = "Frequency",
fill = "Stock Code") +
theme_minimal()

# Calculate the cumulative return of the optimal portfolio
optimal_cum_return <- cumsum(rowSums(weights[max_sharpe_idx, ] * log_returns_matrix, na.rm = TRUE))

# Calculate the cumulative return of each stock
individual_cum_return <- apply(log_returns_matrix, 2, cumsum)

# Create a plot data frame
cum_return_df <- data.frame(
Date = seq_along(optimal_cum_return),
Optimal = optimal_cum_return,
`425` = individual_cum_return[, "425"],
`528` = individual_cum_return[, "528"],
`600761` = individual_cum_return[, "600761"]
)

# Convert to long format
cum_return_long <- cum_return_df %>%
pivot_longer(cols = -Date, names_to = "Portfolio", values_to = "CumulativeReturn")

# Draw the cumulative return curve
ggplot(cum_return_long, aes(x = Date, y = CumulativeReturn, color = Portfolio)) +
geom_line(size = 1) +
labs(title = "Cumulative Returns of Optimal Portfolio and Individual Stocks",
x = "Time",
y = "Cumulative Return",
color = "Portfolio") +
theme_minimal()

# Add risk bin column
efficient_frontier <- efficient_frontier %>%
mutate(RiskBin = cut(Volatility, breaks = 10, labels = paste0("Bin", 1:10)))

# Draw a box plot to show the relationship between risk and return
ggplot(efficient_frontier, aes(x = RiskBin, y = Return)) + geom_boxplot(fill = "lightblue", alpha = 0.7) + labs(title = "Risk-Return Relationship by Risk Bins", x = "Risk Level (Binned)", y = "Portfolio Return") + theme_minimal() + theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

Listing 4: Example Code Question 2 a

```
# Convert log_returns_matrix to matrix type
log_returns_matrix <- as.matrix(log_returns_matrix)
optimal_weights <- as.numeric(optimal_weights)

# Calculate daily portfolio returns using optimal weights
portfolio_returns_optimal <- log_returns_matrix %*% optimal_weights

# Normality test for return distribution
cat("\nNormality test for portfolio returns:\n")
# Anderson-Darling Test
ad_test <- ad.test(portfolio_returns_optimal)
cat("Anderson-Darling test statistic:", ad_test$statistic, "p-value:", ad_test$p.value, "\n")

# Draw QQ plot of return distribution
qqnorm(portfolio_returns_optimal, main = "QQ Plot of Portfolio Returns")
qqline(portfolio_returns_optimal, col = "red")

# Calculate VaR - Historical Simulation Method
VaR_portfolio_historical <- quantile(portfolio_returns_optimal, probs = 0.1)

# Calculate VaR - Variance-Covariance Method
portfolio_mean <- mean(portfolio_returns_optimal)
portfolio_sd <- sd(portfolio_returns_optimal)
VaR_portfolio_covariance <- portfolio_mean + qnorm(0.1) * portfolio_sd

# Output VaR Results
cat("\nVaR calculations based on the optimal portfolio:\n")
cat("Historical simulation VaR:", VaR_portfolio_historical, "\n")
cat("Variance-covariance VaR:", VaR_portfolio_covariance, "\n")

# Add visualization of VaR results
# Build comparison data frame
VaR_comparison <- data.frame(
Method = c("Historical", "Covariance"),
VaR = c(VaR_portfolio_historical, VaR_portfolio_covariance)
)

# Draw a bar chart to show VaR of different methods
ggplot(VaR_comparison, aes(x = Method, y = VaR, fill = Method)) +
geom_bar(stat = "identity", position = "dodge", width = 0.5) +
labs(
```

```
title = "Comparison␣of␣VaR␣Methods",
x = "VaR␣Calculation␣Method",
y = "Value␣at␣Risk"
) +
theme_minimal() +
theme(legend.position = "none") +
scale_fill_manual(values = c("Historical" = "red", "
    Covariance" = "blue"))
```

Listing 5: Example Code Question 2 b

```
#2b VaR calculation for a single asset
# Historical simulation method for each asset
colnames(log_returns_matrix) <- unique(data$Stkcd)
VaR_historical_assets <- apply(log_returns_matrix, 2,
    function(x) quantile(x, probs = 0.1)) # Calculate the 10%
    quantile for each asset (i.e., the expected loss does not
    exceed this value at a 95% confidence level)
print(VaR_historical_assets)

# Variance-covariance (normal distribution) method for each
    asset
VaR_covariance_assets <- apply(log_returns_matrix, 2,
    function(x) {
z_score <- qnorm(0.1)
mean_x <- mean(x)
sd_x <- sd(x)
mean_x + z_score * sd_x
})
names(VaR_covariance_assets) <- colnames(log_returns_matrix)
    # Calculate VaR based on the mean and standard deviation
    of the return, and the assumption of normal distribution
print(VaR_covariance_assets)

# Combine all VaR results for comparison
VaR_comparison <- data.frame(
Method = c("Historical", "Covariance"),
Portfolio_VaR = c(VaR_portfolio_historical, VaR_portfolio_
    covariance),
Asset_425 = c(VaR_historical_assets["425"], VaR_covariance_
    assets["425"]),
Asset_528 = c(VaR_historical_assets["528"], VaR_covariance_
    assets["528"]),
Asset_600761 = c(VaR_historical_assets["600761"], VaR_
    covariance_assets["600761"]) ) cat("2b)␣Comparison␣of␣VaR
    :\n") print(VaR_comparison)
```

Listing 6: Example Code Question 2 c

```
#2c Rolling window VaR estimation and backtesting

# Step one: rolling window parameters
```

```r
window_size <- 1000 # rolling window size
confidence_level <- 0.90 # Confidence level of VaR, set the confidence level to 90%
alpha <- 1 - confidence_level # VaR expected breakthrough frequency
significance_level <- 0.05 # Significance level of binomial test

# Ensure data types are consistent
log_returns_matrix_numeric <- as.matrix(log_returns_matrix)
optimal_weights_numeric <- as.numeric(optimal_weights)

#Initialize the result vector
portfolio_returns_rolling <- numeric(nrow(log_returns_matrix_numeric) - window_size)
VaR_historical_rolling <- numeric(nrow(log_returns_matrix_numeric) - window_size)
VaR_covariance_rolling <- numeric(nrow(log_returns_matrix_numeric) - window_size)

# Rolling window VaR calculation
for (i in 1:(nrow(log_returns_matrix_numeric) - window_size)) {
 # Extract rolling window data
 window_data <- log_returns_matrix_numeric[i:(i + window_size - 1), ]

# Current day's portfolio return rate
 portfolio_returns_rolling[i] <- sum(log_returns_matrix_numeric[i + window_size, ] * optimal_weights_numeric)

 # Historical simulation method VaR
 historical_window_returns <- window_data %*% optimal_weights_numeric
 VaR_historical_rolling[i] <- quantile(historical_window_returns, probs = 0.1)

 # Variance-covariance method VaR
 mean_return <- mean(historical_window_returns)
 sd_return <- sd(historical_window_returns)
 VaR_covariance_rolling[i] <- mean_return + qnorm(0.1) * sd_return
}

# Output partial calculation results
cat("Portfolio Returns (Rolling):\n")
print(head(portfolio_returns_rolling))

cat("Historical VaR (Rolling):\n")
print(head(VaR_historical_rolling))

```

```
cat("Covariance VaR (Rolling):\n")
print(head(VaR_covariance_rolling))

# Step 2: Identify VaR breakthroughs
#Historical VaR breakthrough
historical_breaches <- portfolio_returns_rolling < VaR_
   historical_rolling
num_historical_breaches <- sum(historical_breaches)
freq_historical_breaches <- num_historical_breaches / length(
   portfolio_returns_rolling)

# Covariance VaR breakthrough
covariance_breaches <- portfolio_returns_rolling < VaR_
   covariance_rolling
num_covariance_breaches <- sum(covariance_breaches)
freq_covariance_breaches <- num_covariance_breaches / length(
   portfolio_returns_rolling)

# Step 3: Binomial test to evaluate breakthrough frequency
expected_frequency <- 0.1
sample_size <- length(portfolio_returns_rolling)

# Binomial Test - Historical VaR
binomial_test_historical <- binom.test(
 x = num_historical_breaches,
 n = sample_size,
 p = expected_frequency,
 alternative = "two.sided"
)

# Binomial test - Covariance VaR
binomial_test_covariance <- binom.test(
 x = num_covariance_breaches,
 n = sample_size,
 p = expected_frequency,
 alternative = "two.sided"
)

# Output results
cat("Historical VaR breaches:\n")
cat("Number of breaches:", num_historical_breaches, "\n")
cat("Breach frequency:", freq_historical_breaches, "\n")
cat("Binomial test results:\n")
print(binomial_test_historical)

cat("\nCovariance VaR breaches:\n")
cat("Number of breaches:", num_covariance_breaches, "\n")
cat("Breach frequency:", freq_covariance_breaches, "\n")
cat("Binomial test results:\n")
print(binomial_test_covariance)
```

```r
# Plot rolling VaR and actual returns
plot(
 portfolio_returns_rolling,
 type = "l",
 col = "blue",
 ylab = "Returns / VaR",
 xlab = "Time",
 main = "Portfolio Returns vs. VaR Thresholds (Rolling Window)",
 lwd = 1.5
)
lines(VaR_historical_rolling, col = "red", lty = 2, lwd = 1.5)
lines(VaR_covariance_rolling, col = "green", lty = 3, lwd = 1.5)
legend(
 "topright",
 legend = c("Portfolio Returns", "Historical VaR", "Covariance VaR"),
 col = c("blue", "red", "green"),
 lty = c(1, 2, 3),
 lwd = 1.5
)

# Step 4: VaR comparison visualization
# Align date range
dates <- 1:length(portfolio_returns_rolling)

VaR_comparison_total <- data.frame(
 Method = c(rep("Historical", length(VaR_historical_rolling)),
 rep("Covariance", length(VaR_covariance_rolling)),
 rep("Rolling GARCH", length(portfolio_returns_rolling))),
 Date = c(dates, dates, dates),
 VaR = c(VaR_historical_rolling, VaR_covariance_rolling, portfolio_returns_rolling)
)

# Draw VaR comparison
ggplot(VaR_comparison_total, aes(x = Date, y = VaR, color = Method)) +
 geom_line() +
 labs(
 title = "VaR Comparison across Methods",
 x = "Time",
 y = "Value at Risk"
 ) +
 theme_minimal()
```

Listing 7: Example Code Question 2 d

```r
# 2d
```

```
# 1. Define extreme market periods
extreme_periods <- data.frame(
  Period = c("2008 Financial Crisis", "2015 Stock Crash", "COVID-19 Pandemic"),
  Start = as.Date(c("2008-07-01", "2015-06-15", "2020-02-04")),
  End = as.Date(c("2008-10-28", "2015-08-31", "2020-03-31"))
)

# 2. Create a sample dataset
set.seed(123)  # keep the result consistent
rolling_var_results <- data.frame(
  Date = seq.Date(from = as.Date("2008-01-01"), to = as.Date("2020-12-31"), by = "day"),
  VaR = rnorm(4749, mean = -0.02, sd = 0.01),
  Actual = rnorm(4749, mean = 0, sd = 0.02),
  Breach = sample(c(TRUE, FALSE), 4749, replace = TRUE)
)

# 3. Mark extreme periods in rolling VaR results
rolling_var_results <- rolling_var_results %>%
  mutate(
    Period = case_when(
      Date >= extreme_periods$Start[1] & Date <= extreme_periods$End[1] ~ extreme_periods$Period[1],
      Date >= extreme_periods$Start[2] & Date <= extreme_periods$End[2] ~ extreme_periods$Period[2],
      Date >= extreme_periods$Start[3] & Date <= extreme_periods$End[3] ~ extreme_periods$Period[3],
      TRUE ~ "Normal Period"
    )
  )

# 4. Calculate the frequency of breakthroughs in different periods
period_summary <- rolling_var_results %>%
  group_by(Period) %>%
  summarize(
    Breach_Frequency = mean(Breach, na.rm = TRUE),
    Observations = n(),
    .groups = "drop"
  )
print(period_summary)

# 5. Visualize extreme market VaR and actual returns
ggplot(rolling_var_results, aes(x = Date)) +
  geom_line(aes(y = VaR, color = "VaR"), size = 1, linetype = "dashed") +
  geom_line(aes(y = Actual, color = "Actual Returns"), size = 1) +
  geom_point(data = rolling_var_results %>% filter(Breach ==
```

```
    TRUE),
             aes(x = Date, y = Actual), color = "black", size = 2) +
  facet_wrap(~ Period, scales = "free_x") +
  scale_color_manual(values = c("VaR" = "red", "Actual Returns" = "blue")) +
  ggtitle("VaR Performance in Normal vs Extreme Market Periods") +
  xlab("Date") +
  ylab("Returns and VaR") +
  theme_minimal() +
  theme(legend.position = "top")
```

Listing 8: Example Code Question 3 a

```
# 3a
# Define GARCH(1,1) fitting function
get_garch_volatility <- function(stock_id, data) {
  # Filter data for specific stock
  stock_data <- data %>% filter(Stkcd == stock_id) %>% drop_na(LogReturn)

  # Convert data types
  stock_data <- stock_data %>%
    mutate(
      LogReturn = as.numeric(LogReturn),
      Trddt = as.Date(Trddt)
    )

  # Specify GARCH(1,1) model with Student's t distribution
  spec <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
                     mean.model = list(armaOrder = c(0, 0)),
                     distribution.model = "std")

  # Fit model and extract conditional volatility and residual tests
  tryCatch({
    fit <- ugarchfit(spec = spec, data = stock_data$LogReturn, solver = "hybrid")

    # Get residuals with timestamps
    residuals <- data.frame(
      Date = stock_data$Trddt,  # Map to actual trading dates
      Residuals = residuals(fit, standardize = TRUE)
    )

    # Ljung-Box test
    lb_test <- Box.test(residuals$Residuals, lag = 10, type = "Ljung-Box")
    cat("Ljung-Box Test Results for Stock", stock_id, ":\n")
    print(lb_test)
```

```r

    # Calculate skewness and kurtosis
    library(moments)
    skewness_value <- skewness(residuals$Residuals, na.rm = TRUE)
    kurtosis_value <- kurtosis(residuals$Residuals, na.rm = TRUE) - 3  # Excess kurtosis
    cat("Skewness for Stock", stock_id, ":", skewness_value, "\n")
    cat("Excess Kurtosis for Stock", stock_id, ":", kurtosis_value, "\n")

    # Plot residuals time series
    library(ggplot2)
    p1 <- ggplot(residuals, aes(x = Date, y = Residuals)) +
      geom_line(color = "blue") +
      geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
      ggtitle(paste("Standardized Residuals Time Series for Stock", stock_id)) +
      xlab("Date") +
      ylab("Residuals") +
      theme_minimal()
    print(p1)

    # Plot residuals histogram
    p2 <- ggplot(residuals, aes(x = Residuals)) +
      geom_histogram(bins = 30, fill = "blue", color = "black", alpha = 0.7) +
      geom_density(color = "red", linetype = "dashed") +
      ggtitle(paste("Residuals Distribution for Stock", stock_id)) +
      xlab("Residuals") +
      ylab("Frequency") +
      theme_minimal()
    print(p2)

    # Return conditional volatility data
    stock_data %>%
      mutate(Conditional_Volatility = sigma(fit)) %>%
      select(Trddt, Conditional_Volatility, Stkcd)
  }, error = function(e) {
    message(paste("GARCH fitting failed for stock", stock_id, ":", e$message))
    return(NULL)
  })
}

# Get first three unique stock codes
unique_stocks <- unique(data$Stkcd)[1:3]

```

```r
# Combine conditional volatility data for all three stocks
all_volatility <- bind_rows(lapply(unique_stocks, get_garch_volatility, data = data))

# Standardize data types
all_volatility <- all_volatility %>%
  mutate(
    Trddt = as.Date(Trddt),
    Conditional_Volatility = as.numeric(Conditional_Volatility),
    Stkcd = as.factor(Stkcd)
  )

# Plot conditional volatility for each stock
for (stock in unique_stocks) {
  stock_data <- all_volatility %>% filter(Stkcd == stock)

  p <- ggplot(stock_data, aes(x = Trddt, y = Conditional_Volatility)) +
    geom_line(color = "blue", size = 1) +
    ggtitle(paste("Conditional Volatility for Stock Code", stock)) +
    xlab("Date") +
    ylab("Conditional Volatility") +
    scale_x_date(
      date_breaks = "1 year",  # Display by year
      date_labels = "%Y"       # Set date format to "year"
    ) +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate date labels to prevent overlap

  print(p)
}

# Plot comparison of conditional volatility for three stocks
p_all <- ggplot(all_volatility, aes(x = Trddt, y = Conditional_Volatility, color = Stkcd)) +
  geom_line(size = 1) +
  ggtitle("Comparison of Conditional Volatility for Three Stocks") +
  xlab("Date") +
  ylab("Conditional Volatility") +
  scale_x_date(
    date_breaks = "1 year",  # Display by year
    date_labels = "%Y"       # Set date format to "year"
  ) +
  scale_color_discrete(name = "Stock Code") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate date labels
```

```r

print(p_all)
```

Listing 9: Example Code Question 3 b

```r
# 3b
# Define extreme market periods
extreme_periods <- data.frame(
  Period = c("2008 Financial Crisis", "2015 Stock Crash", "COVID-19 Pandemic"),
  Start = as.Date(c("2008-07-01", "2015-06-15", "2020-02-04")),
  End = as.Date(c("2008-10-28", "2015-08-31", "2020-03-31"))
)

# Mark extreme periods in volatility data
all_volatility <- all_volatility %>%
  mutate(Period = "Normal Period")  # Initialize as "Normal Period"

for (i in seq_len(nrow(extreme_periods))) {
  all_volatility <- all_volatility %>%
    mutate(Period = ifelse(Trddt >= extreme_periods$Start[i] & Trddt <= extreme_periods$End[i],
                           extreme_periods$Period[i], Period))
}

# Conduct t-tests between extreme and normal periods
t_test_results <- list()

for (stock in unique(all_volatility$Stkcd)) {
  stock_data <- all_volatility %>% filter(Stkcd == stock)

  # Get data for normal period and each extreme period
  for (extreme_period in unique(extreme_periods$Period)) {
    normal_data <- stock_data %>% filter(Period == "Normal Period") %>% pull(Conditional_Volatility)
    extreme_data <- stock_data %>% filter(Period == extreme_period) %>% pull(Conditional_Volatility)

    # Check if sufficient data for t-test
    if (length(normal_data) > 1 && length(extreme_data) > 1) {
      t_test <- t.test(normal_data, extreme_data, var.equal = FALSE)  # Welch's t-test
      t_test_results[[paste(stock, extreme_period, sep = "_")]] <- list(
        Stock = stock,
        Period = extreme_period,
        T_Statistic = t_test$statistic,
        P_Value = t_test$p.value
```

```
      )
    }
  }
}

# Convert t-test results to dataframe
t_test_results_df <- do.call(rbind, lapply(t_test_results, as.data.frame)) %>%
  mutate(P_Value = round(P_Value, 4))  # Round to 4 decimal places

# Print t-test results
print(t_test_results_df)

# Summarize conditional volatility means and standard deviations by period
volatility_summary <- all_volatility %>%
  group_by(Stkcd, Period) %>%
  summarize(
    Mean_Volatility = round(mean(Conditional_Volatility, na.rm = TRUE), 4),
    SD_Volatility = round(sd(Conditional_Volatility, na.rm = TRUE), 4),
    .groups = "drop"
  )

# Print volatility summary statistics
print(volatility_summary)

# Plot conditional volatility for each stock across different periods
for (stock in unique(all_volatility$Stkcd)) {
  stock_data <- all_volatility %>% filter(Stkcd == stock)

  # Create plot
  p <- ggplot(stock_data, aes(x = Trddt, y = Conditional_Volatility, color = Period)) +
    geom_line(size = 1) +
    ggtitle(paste("Conditional Volatility for Stock Code", stock)) +
    xlab("Date") +
    ylab("Conditional Volatility") +
    scale_color_manual(
      values = c(
        "Normal Period" = "blue",
        "2008 Financial Crisis" = "red",
        "2015 Stock Crash" = "purple",
        "COVID-19 Pandemic" = "orange"
      )
    ) +
    theme_minimal() +
```

```
    theme(legend.position = "top") +
    # Add vertical lines marking extreme periods
    geom_vline(data = extreme_periods,
               aes(xintercept = as.numeric(Start), color = Period),
               linetype = "dashed", show.legend = FALSE) +
    geom_vline(data = extreme_periods,
               aes(xintercept = as.numeric(End), color = Period),
               linetype = "dashed", show.legend = FALSE)

  print(p)
}
```

Listing 10: Example Code Question 3 c

```
# 3c
# Step 1: Verify input data (portfolio_returns)
if (!exists("optimal_weights") || !exists("data")) {
  stop("Ensure 'optimal_weights' and 'data' are defined before proceeding.")
}

# Calculate portfolio returns using optimal weights
portfolio_returns <- data %>%
  filter(Stkcd %in% colnames(log_returns_matrix)) %>%
  select(Trddt, Stkcd, LogReturn) %>%
  pivot_wider(names_from = Stkcd, values_from = LogReturn) %>%
  mutate(
    Portfolio_Return = rowSums(as.matrix(select(., all_of(colnames(log_returns_matrix)))) * optimal_weights)
  ) %>%
  select(Trddt, Portfolio_Return) %>%
  drop_na()

# Verify portfolio returns data
if (!"Portfolio_Return" %in% colnames(portfolio_returns) || !"Trddt" %in% colnames(portfolio_returns)) {
  stop("Input data must contain 'Portfolio_Return' and 'Trddt' columns.")
}
print(head(portfolio_returns))

# Step 2: Define rolling GARCH VaR calculation function
calculate_rolling_garch_var <- function(data, confidence_level = 0.90, window_size = 1000) {
  if (nrow(data) < window_size) {
    stop("Not enough data for rolling window calculation")
  }

  spec <- ugarchspec(
```

```
    variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
    mean.model = list(armaOrder = c(0, 0)),
    distribution.model = "norm"
  )

  rolling_var <- data.frame(Date = as.Date(character()), VaR = numeric(), Actual = numeric())

  for (i in seq(window_size + 1, nrow(data))) {
    train_data <- data$Portfolio_Return[(i - window_size):(i - 1)]
    if (any(is.na(train_data)) || length(train_data) < window_size) {
      warning(paste("Skipping due to NA or insufficient data at:", data$Trddt[i]))
      next
    }

    fit <- tryCatch({
      ugarchfit(spec, data = train_data, solver = "hybrid", out.sample = 1)
    }, error = function(e) {
      warning(paste("GARCH fit failed at:", data$Trddt[i]))
      return(NULL)
    })

    if (is.null(fit)) next

    forecast <- ugarchforecast(fit, n.ahead = 1)
    sigma_forecast <- forecast@forecast$sigmaFor[1]
    mean_forecast <- forecast@forecast$seriesFor[1]
    var <- qnorm(1 - confidence_level) * sigma_forecast + mean_forecast

    rolling_var <- rbind(
      rolling_var,
      data.frame(
        Date = data$Trddt[i],
        VaR = var,
        Actual = data$Portfolio_Return[i]
      )
    )
  }

  return(rolling_var)
}

# Step 3: Run rolling GARCH VaR
rolling_var_results <- calculate_rolling_garch_var(portfolio_returns, confidence_level = 0.90)
```

```

# Check results
if (nrow(rolling_var_results) == 0) {
  stop("Rolling VaR results are empty. Check your data and parameters.")
} else {
  print(head(rolling_var_results))
}

# Step 4: Define VaR backtesting function
backtest_var <- function(rolling_var, confidence_level = 0.90) {
  rolling_var <- rolling_var %>%
    mutate(Breach = Actual < VaR)

  breach_count <- sum(rolling_var$Breach)
  total_obs <- nrow(rolling_var)
  breach_ratio <- breach_count / total_obs

  expected_breach_ratio <- 1 - confidence_level
  binom_test <- binom.test(breach_count, total_obs, expected_breach_ratio)

  list(
    Breach_Ratio = breach_ratio,
    Binom_Test = binom_test,
    Rolling_Var = rolling_var  # Return marked dataframe
  )
}

# Step 5: VaR backtesting
backtest_results <- backtest_var(rolling_var_results, confidence_level = 0.90)

# Step 6: Print backtest results
print(paste("Breach Ratio:", round(backtest_results$Breach_Ratio, 4)))
print(paste("Binomial Test p-value:", round(backtest_results$Binom_Test$p.value, 4)))

# Step 7: Plot VaR vs actual returns with breach points
rolling_var_results <- backtest_results$Rolling_Var

ggplot(rolling_var_results, aes(x = Date)) +
  geom_line(aes(y = VaR, color = "VaR"), size = 1, linetype = "dashed") +
  geom_line(aes(y = Actual, color = "Actual Returns"), size = 1) +
  geom_point(data = rolling_var_results %>% filter(Breach == TRUE),
             aes(x = Date, y = Actual), color = "black", size
```

```
                    = 2) +
  scale_color_manual(values = c("VaR" = "red", "Actual␣
     Returns" = "blue")) +
  ggtitle("GARCH␣VaR␣Backtest␣with␣Breaches␣for␣Optimal␣
     Portfolio") +
  xlab("Date") +
  ylab("Returns␣and␣VaR") +
  theme_minimal() +
  theme(legend.position = "top")
```