MM955 Financial Econometrics

University of Strathclyde

Table of Contents

Chapter 1 – Introduction

This report presents an analysis of the S&P 500 index, one of the most widely followed equity indices that tracks the performance of approximately 500 large companies listed on U.S. stock exchanges. The S&P 500 is commonly used as a benchmark due to the wide market breath of large-cap companies included in the index. (Beers, 2023)

The dataset comprises daily closing prices of the S&P 500 from January 2020 to March 2025, encompassing 1,302 observations. This period is particularly interesting as it captures several significant market phases: the COVID-19 pandemic crash in early 2020, the subsequent recovery, the market volatility of 2022-2023, and the more recent market performance. These diverse market conditions provide a rich environment for examining various financial phenomena, including volatility clustering and asymmetric responses to market shocks.

The objective of this analysis is to investigate the statistical properties of the S&P 500 returns and develop appropriate models for both the returns themselves and their volatility. Specifically, this report addresses five key areas:

1. Examination of the stationarity properties of the time series and appropriate transformations
2. Analysis of the return distribution and estimation of Value at Risk (VaR) at different confidence levels
3. Development of a suitable time series model for the return dynamics
4. Construction of volatility models to capture time-varying risk characteristics
5. Evaluation of the implications of these findings for financial risk management

The methods employed include standard statistical tests for stationarity, normality, and autocorrelation, as well as more advanced econometric techniques such as ARMA modelling for returns and GARCH-type models for volatility.

Through this analysis, we aim to gain insights into the risk-return characteristics of the S&P 500 and demonstrate how sophisticated statistical models can enhance our understanding of financial market behaviour and improve risk assessment capabilities.

Chapter 2 – Data and Stationary Analysis

2.1 Data Description

The analysis utilises daily closing prices of the S&P 500 index from January 2, 2020, to March 7, 2025, comprising 1,302 observations obtained from (Yahoo Finance , 2025). Figure 1 presents the time series plot of the S&P 500 closing prices over this period.

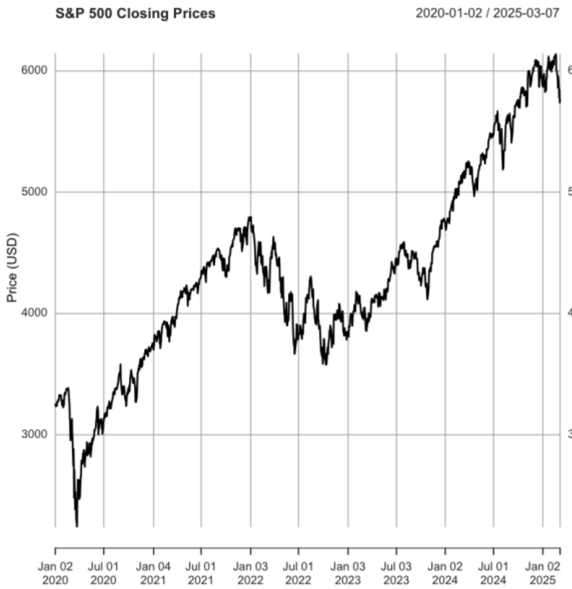


Figure 1 - S&P500 Closing Prices

The summary statistics in Table 1 provide an overview of the price data, revealing a minimum value of 2,237 and a maximum of 6,144, with a mean of 4,318. These statistics highlight the substantial growth in the index over the five-year period, with the mean value situated closer to the third quartile (4,697) than the median (4,241), suggesting a positively skewed distribution over time.

2.2 Stationarity Tests on Price Series

A fundamental requirement for time series modelling is stationarity, which implies that the statistical properties of the series do not change over time. The Augmented Dickey-Fuller (ADF) test was conducted to assess whether the S&P 500 price series is stationary. The null hypothesis of the ADF test is that the series contains a unit root (non-stationary).

The ADF test on the price series yielded a test statistic of -1.9744 with a p-value of 0.5891, failing to reject the null hypothesis at any conventional significance level. This confirms the visual impression from Figure 1 that the price series is non-stationary. The KPSS test, which has a null hypothesis of stationarity, further supported this conclusion with a test statistic of 11.513 and a p-value of 0.01, strongly rejecting the stationarity hypothesis.

2.3 Log-Returns Transformation

To address the non-stationarity issue, the price series was transformed into logarithmic returns, calculated as:

$$r_t = log(P_t/P_{t-1}) = log(P_t) - log(P_{t-1})$$

where P_t represents the closing price at time t. This transformation is standard practice in financial time series analysis as it approximates the percentage change while making the series more amenable to statistical modelling. Figure 2 displays the time series of daily log returns.

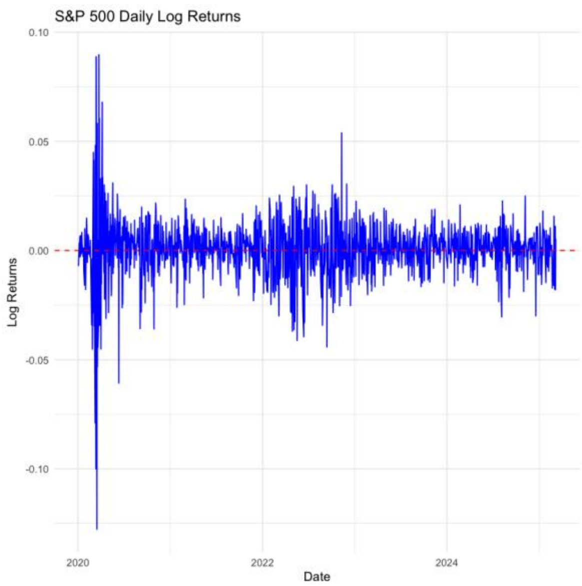


Figure 2 - S&P 500 Daily Log Returns

2.4 Stationarity Analysis of Log-Returns

The stationarity of the log-returns series was assessed using both the ADF and KPSS tests. The ADF test resulted in a test statistic of -10.215 with a p-value of 0.01, strongly rejecting the null hypothesis of non-stationarity. The KPSS test yielded a test statistic of 0.054449 with a p-value of 0.1, failing to reject the null hypothesis of stationarity. These results collectively confirm that the log-returns transformation successfully converted the non-stationary price series into a stationary series.

Table 2 presents the summary statistics of the log-returns series. The mean daily log return is 0.0004394 (approximately 0.04% per day), with a standard deviation of 0.01338. The minimum and maximum values of -0.1277 and 0.0897, respectively, indicate the presence of significant extreme returns during the analysis period.

The stationarity of the log-returns series is a crucial finding as it allows for the application of standard time series modelling techniques in the subsequent analysis. The next section will explore the distributional properties of these log-returns to better understand the risk characteristics of the S&P 500 index.

Chapter 3 – Distribution of Returns

3.1 Summary Statistics and Distributional Properties

The log-returns of the S&P 500 index exhibit several notable statistical properties. As shown in Table 3, the mean daily log-return is 0.0004394 (approximately 0.04% per day), with a standard deviation of 0.01338. The distribution demonstrates negative skewness of -0.8136, indicating a longer left tail with more extreme negative returns than would be expected in a normal distribution. Furthermore, the kurtosis value of 17.1966 is substantially higher than the normal distribution's value of 3, confirming the presence of fat tails and a more peaked distribution.

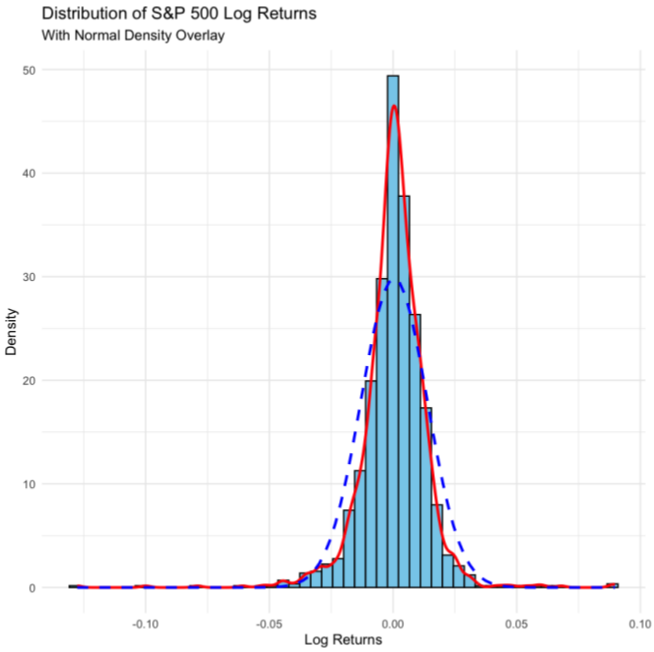


Figure 3 - Distribution of S&P 500 Log Returns with Normal Density Overlay

Figure 3 presents the distribution of log-returns with a normal density overlay, visually confirming the departure from normality. The observed distribution is more peaked around the centre and has heavier tails than the normal distribution, consistent with the high kurtosis value.

3.2 Tests for Normality

To formally test the normality of the log-returns distribution, both the Shapiro-Wilk and Jarque-Bera tests were conducted. The Shapiro-Wilk test resulted in a W-statistic of 0.87423 with a p-value < 2.2e-16, strongly rejecting the null hypothesis of normality. Similarly, the Jarque-Bera test yielded a chi-squared statistic of 11,069 with a p-value < 2.2e-16, providing additional evidence against normality.

The Q-Q plot in Figure 4 further illustrates the deviation from normality, with points deviating substantially from the reference line at both tails of the distribution.

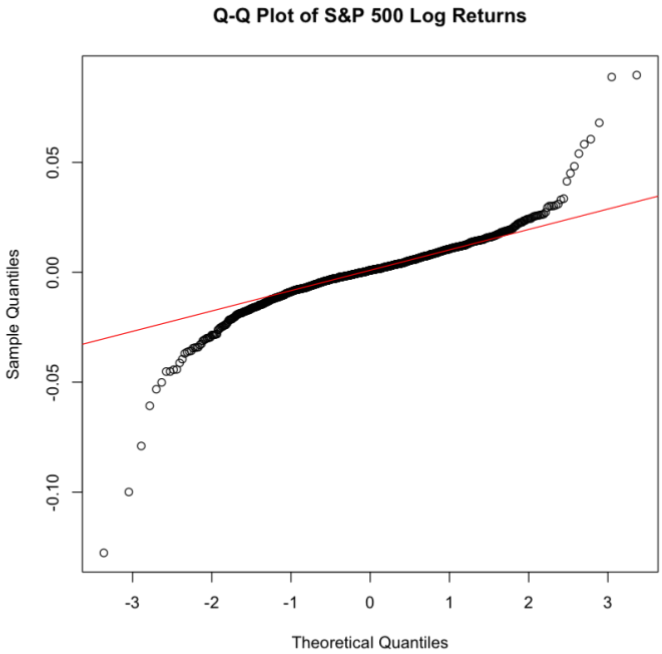


Figure 4 - Q-Q Plot of S&P 500 Log Returns

3.3 Value at Risk Analysis

Value at Risk (VaR) is a widely used risk measure that quantifies the potential loss in value of a portfolio over a defined period for a given confidence interval. Given the non-normal characteristics of the S&P 500 returns, VaR was estimated using three different approaches:

1. Historical VaR: Based on empirical quantiles of the observed return distribution
2. Parametric VaR (normal): Assuming returns follow a normal distribution
3. Parametric VaR (t-distribution): Using a Student's t-distribution to account for fat tails

Table 4 presents the VaR estimates at different confidence levels (90%, 95%, 99%, and 99.5%).

The results reveal several important patterns. At lower confidence levels (90% and 95%), the three methods produce relatively similar VaR estimates. However, at higher confidence levels (99% and 99.5%), the t-distribution VaR produces significantly higher estimates than the normal VaR, reflecting the fat-tailed nature of the return distribution. The historical VaR falls between these two parametric estimates at higher confidence levels, suggesting that the empirical distribution has fatter tails than the normal distribution but not as extreme as predicted by the fitted t-distribution.

3.4 Expected Shortfall Analysis

Expected Shortfall (ES), also known as Conditional VaR (CVaR), measures the expected loss given that the loss exceeds the VaR threshold. This provides a more comprehensive assessment of tail risk. (TIO Staff, 2024) Table 5 presents the ES estimates at different confidence levels.

The ES values are substantially higher than the corresponding VaR estimates, particularly at high confidence levels. For instance, at the 99% confidence level, while the historical VaR is 3.63%, the ES is 5.74%, indicating that when losses exceed the VaR threshold, they tend to be considerably larger than the VaR itself. This finding underscores the importance of considering measures beyond VaR for comprehensive risk management.

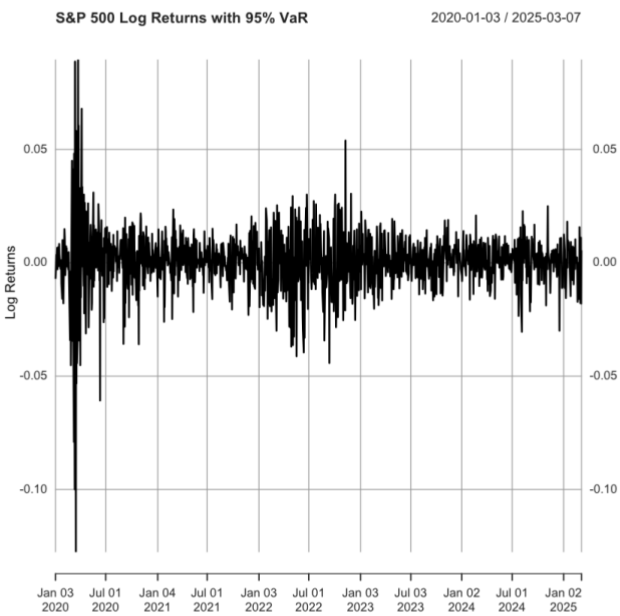


Figure 5 - S&P 500 Log Returns with 95% VaR

Figure 5 illustrates the daily log-returns with the 95% VaR threshold, visually demonstrating how often and by how much the returns exceed the VaR level. The graph highlights the clustering of volatility, with periods of high volatility characterized by more frequent and larger VaR exceedances.

These findings on the distributional properties and risk measures provide valuable insights for risk management and portfolio optimisation strategies, emphasising the need to account for non-normality and fat tails when assessing financial risk.

Chapter 4 – Time Series Modelling

4.1 Autocorrelation Analysis

Before fitting a time series model, the autocorrelation structure of the log-returns was examined. The autocorrelation function (ACF) and partial autocorrelation function (PACF) revealed significant autocorrelations at several lags, indicating potential serial dependence in the return series. The Ljung-Box test confirmed this finding with test statistics of 244.42 and 296.12 at lags 10 and 20, respectively, both with p-values < 2.2e-16, strongly rejecting the null hypothesis of no autocorrelation.

4.2 ARMA Model Selection

Based on the autocorrelation analysis, several ARMA(p,q) models were fitted to the log-returns series and compared using information criteria. Table 6 presents the AIC and BIC values for the candidate models.

The ARMA(1,2) model emerged as the best fit with the lowest AIC value of -7582.887, followed closely by ARMA(2,1) with an AIC of -7580.789. While the BIC criterion, which penalizes model complexity more heavily, slightly favoured the more parsimonious ARMA(1,1) model, the ARMA(1,2) specification was selected as the final model based on its superior fit according to AIC.

4.3 Model Diagnostics

Diagnostic checks were performed on the residuals of the ARMA(1,2) model to assess whether it adequately captured the serial dependence in the returns. Despite the model's good fit, the Ljung-Box test on the residuals yielded p-values of 5.506e-13 and 4.478e-13 at lags 10 and 20, respectively, indicating that significant autocorrelation remained in the residuals.

Further analysis of the squared residuals revealed strong evidence of conditional heteroskedasticity (time-varying volatility). The Box-Ljung test on squared residuals produced a test statistic of 1252.7 (p-value < 2.2e-16), and the ARCH-LM test yielded a test statistic of 411.17 (p-value $\approx$ 0), both decisively rejecting the null hypothesis of no ARCH effects.

These findings suggest that while the ARMA(1,2) model captures some of the serial dependence in the return series, it fails to account for the heteroskedasticity in the data. This is a common characteristic of financial time series and motivates the development of volatility models in the next section.

The residuals from the ARMA model also exhibited significant departures from normality, with the Shapiro-Wilk test yielding a W-statistic of 0.899 (p-value < 2.2e-16) and the Jarque-Bera test producing a chi-squared statistic of 7277.3 (p-value < 2.2e-16). This non-normality in the residuals further supports the need for more sophisticated modelling approaches that can accommodate both the serial dependence and the time-varying volatility in the return series.

In summary, while the ARMA(1,2) model provides insights into the mean dynamics of the S&P 500 returns, it is insufficient for capturing the complex volatility patterns observed in the data. The next section will address this limitation by developing models specifically designed to account for the heteroskedasticity in the return series.

Chapter 5 – Volatility Modelling

5.1 Volatility Model Selection

Based on the strong ARCH effects in the ARMA residuals, three volatility models were estimated: GARCH(1,1)-normal, GARCH(1,1)-t, and APARCH(1,1)-t. The APARCH(1,1)-t model provided the best fit with the lowest AIC value (-6.351), indicating the presence of asymmetric volatility responses to positive and negative shocks.

5.2 Model Interpretation

1. The APARCH(1,1)-t model revealed:
2. A small but significant positive daily return ($\mu = 0.00066$)
3. High volatility persistence ($\alpha+\beta = 0.9872$)
4. Strong asymmetry effect ($\gamma = 1.0000$), confirming that negative returns increase volatility more than positive returns
5. A shape parameter of 7.03, supporting the use of a fat-tailed distribution

Diagnostic tests showed the APARCH model successfully captured both the serial correlation and volatility dynamics, with Ljung-Box tests on standardized residuals yielding p-values above conventional significance levels.

5.3 Volatility Forecasting and Risk Assessment

The 5-day volatility forecast showed a gradual decrease from 0.01567 to 0.01489, suggesting a relatively stable near-term market environment. Incorporating these time-varying volatility estimates, the model produced volatility-adjusted VaR estimates of 3.29% (95% confidence) and 5.20% (99% confidence).

5.4 Implications of Findings

The analysis yields several important implications for financial risk management:

Time-varying risk: Market risk varies over time, necessitating dynamic risk management approaches

1. Fat-tailed distribution: Higher probability of extreme events requires appropriate distributional assumptions

2. Leverage effect: Market downturns increase volatility more than equivalent upturns, affecting option pricing and hedging strategies
3. Volatility persistence: Once elevated, volatility tends to remain high for extended periods
4. Model sophistication: Simple models may significantly underestimate tail risks, particularly during market stress

These findings emphasise the importance of incorporating time-varying volatility, fat-tailed distributions, and asymmetric market responses in risk management frameworks to accurately assess financial risks.

Chapter 6 – Conclusion and Implications

This study analysed S&P 500 returns from January 2020 to March 2025, examining stationarity, distributional properties, and volatility dynamics. The findings reveal significant non-normality in returns, with substantial negative skewness (-0.814) and excess kurtosis (17.20) indicating fat tails and higher probability of extreme events.

While an ARMA(1,2) model captured serial dependence in returns, the APARCH(1,1) model with t-distributed innovations provided superior fit for the volatility structure, identifying both high volatility persistence ($\alpha+\beta = 0.987$) and significant asymmetric responses to market shocks (leverage effect). The substantial difference between traditional VaR estimates and volatility-adjusted measures (99% one-day VaR of 3.07% vs. 5.20%) highlights the danger of assumptions based on normal distributions.

The practical implications for financial practitioners are clear: risk management frameworks should incorporate time-varying volatility and fat-tailed distributions to avoid underestimating tail risks, especially during market stress. The leverage effect suggests emphasizing downside protection during market downturns, while high volatility persistence indicates the need for sustained risk management during turbulent periods rather than assuming quick mean reversion.

This analysis demonstrates that sophisticated statistical techniques are not merely academic exercises but essential tools for accurate risk assessment in financial markets, where simpler approaches may lead to significant risk underestimation with potentially severe consequences.

References

Beers, B. (2023, 8 30). *Why Do Investors Use the S&P 500 as a Benchmark?* Retrieved from Investopedia: https://www.investopedia.com/ask/answers/041315/what-are-pros-and-cons-using-sp-500-benchmark.asp#:~:text=The%20key%20advantage%20of%20using,in%20so%20many%20different%20sectors.

MLAHayashi, F. (2000). *Econometrics.* Princeton: Princeton University Press.

TIO Staff. (2024, July 8). *Expected shortfall: Explained | TIOmarkets* . Retrieved from Tiomarkets: https://tiomarkets.com/article/expected-shortfall-guide

Yahoo Finance . (2025, March 9). *API Documentation*. Retrieved from yahoofinance: https://python-yahoofinance.readthedocs.io/en/latest/api.html

Appendices

Table 1 - Summary Statistics of S&P 500

```
     Index               GSPC.Close
Min.   :2020-01-02   Min.   :2237
1st Qu.:2021-04-19   1st Qu.:3829
Median :2022-08-02   Median :4241
Mean   :2022-08-02   Mean   :4318
3rd Qu.:2023-11-15   3rd Qu.:4697
Max.   :2025-03-07   Max.   :6144
```

Table 2 - Summary Statistics

```
     Index               LogReturns
Min.   :2020-01-03   Min.   :-0.1276522
1st Qu.:2021-04-20   1st Qu.:-0.0053142
Median :2022-08-03   Median : 0.0008874
Mean   :2022-08-03   Mean   : 0.0004394
3rd Qu.:2023-11-16   3rd Qu.: 0.0071967
Max.   :2025-03-07   Max.   : 0.0896832
```

Table 3 - Stationary Tests

Test	Variable	Test Statistic	p-value	Result
ADF	S&P 500 Prices	-1.9744	0.5891	Non-stationary
KPSS	S&P 500 Prices	11.513	0.01	Non-stationary
ADF	Log Returns	-10.215	0.01	Stationary
KPSS	Log Returns	0.054449	0.1	Stationary

Table 4 - Value at Risk Estimates at Different Confidence Levels

Confidence Level	Historical VaR (%)	Normal VaR (%)	T-Distribution VaR (%)
90%	1.31	1.67	1.93
95%	1.86	2.16	2.65
99%	3.63	3.07	4.46
99.5%	4.52	3.40	5.35

Table 5 - Expected Shortfall Estimates

Confidence Level	Expected Shortfall (%)
90%	2.41
95%	3.24
99%	5.74
99.5%	7.37

Table 6 - ARMA Model Comparison

Model	AIC	BIC
ARIMA(1, 0, 2)	-7582.887	-7557.033
ARIMA(2, 0 ,1)	-7580.789	-7554.934
ARIMA(1, 0, 1)	-7577.470	-7556.786
ARIMA(1, 0, 0)	-7571.048	-7555.535
ARIMA(0, 0, 1)	-7563.109	-7547.597
ARIMA(0, 0, 0)	-7530.379	-7520.038

Table 7 - Volatility Model Comparison

Model	Log Likelihood	AIC	BIC
GARCH(1,1)-Normal	-4098.027	-6.293663	-6.277764
GARCH(1,1)-t	-4123.332	-6.331025	-6.311153
APARCH(1,1)-t	-4138.179	-6.350775	-6.322953

Table 8 - APARCH(1,1) -t Model Parameter Estimates

Parameter	Estimate	Std. Error	t-value	Pr(>|t|)	Significance
mu	0.0006629	0.0002350	2.821	0.00479	**
omega	0.0004828	0.0001195	4.040	5.35e-05	***
alpha1	0.1068491	0.0146535	7.292	3.06e-13	***
gamma1	1.0000000	0.0167989	59.528	< 2e-16	***
beta1	0.8804478	0.0169608	51.911	< 2e-16	***
delta	0.9707855	0.1736181	5.591	2.25e-08	***
shape	7.0268322	1.2097559	5.808	6.30e-09	***

Table 9 - APARCH (1,1)-t Model Parameter Estimates

Date	Mean	Volatility
2025-03-08	0.0006629204	0.01567243
2025-03-09	0.0006629204	0.01546482
2025-03-10	0.0006629204	0.01526562
2025-03-11	0.0006629204	0.01507450
2025-03-12	0.0006629204	0.01489111

Table 10 - Diagnostic Tests for APARCH Model

Test	Statistic	p-Value
Jarque-Bera Test (R)	39177.16	0.0000000
Shapiro-Wilk Test (R)	0.8839096	0.0000000
Ljung-Box Test R Q(10)	14.51418	0.1508034
Ljung-Box Test R Q(15)	16.86813	0.3268084
Ljung-Box Test R Q(20)	25.77746	0.1732881
Ljung-Box Test R^2 Q(10)	0.6383792	0.9999788
Ljung-Box Test R^2 Q(15)	1.418943	0.9999971
Ljung-Box Test R^2 Q(20)	1.76511	1.0000000
LM Arch Test R TR^2	6.368389	0.8963854

Part B - Project on Financial Data Modelling

Project Initialisation

In [31]:
```
# Install required packages if not already installed
if (!require("xts")) install.packages("xts")
if (!require("ggplot2")) install.packages("ggplot2")
if (!require("tseries")) install.packages("tseries")
if (!require("moments")) install.packages("moments")
if (!require("urca")) install.packages("urca")
if (!require("stats")) install.packages("stats")
if (!require("fGarch")) install.packages("fGarch")
library(xts)
library(ggplot2)
library(tseries)
library(urca)
library(stats)
library(fGarch)
```

In [32]:
```
# Set the time period for data (approximately 4-5 years for ~1000 observati
start_date <- as.Date("2020-01-01")
end_date <- Sys.Date()  # Current date

# Download S&P 500 data
# The symbol ^GSPC is used for S&P 500 on Yahoo Finance
getSymbols("^GSPC", from = start_date, to = end_date, src = "yahoo")

# Basic information about the data
head(GSPC)  # View first few rows of data
tail(GSPC)  # View last few rows
dim(GSPC)   # Check dimensions (number of observations)

# Create a basic plot of the S&P 500 closing prices
plot(GSPC$GSPC.Close, main = "S&P 500 Closing Prices",
     xlab = "Date", ylab = "Price (USD)")

# Save the data to a CSV file if needed
write.csv(GSPC, file = "SP500_data.csv")

# Check the structure of the data
str(GSPC)

# Calculate summary statistics
summary(GSPC$GSPC.Close)
```

'GSPC'

```
           GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume GSPC.Adjuste
d
2020-01-02   3244.67   3258.14  3235.53    3257.85  3459930000       3257.8
5
2020-01-03   3226.36   3246.15  3222.34    3234.85  3484700000       3234.8
5
2020-01-06   3217.55   3246.84  3214.64    3246.28  3702460000       3246.2
8
2020-01-07   3241.86   3244.91  3232.43    3237.18  3435910000       3237.1
8
2020-01-08   3238.59   3267.07  3236.67    3253.05  3726840000       3253.0
5
2020-01-09   3266.03   3275.58  3263.67    3274.70  3641230000       3274.7
0
           GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume GSPC.Adjuste
d
2025-02-28   5856.74   5959.40  5837.66    5954.50  6441140000       5954.5
0
2025-03-03   5968.33   5986.09  5810.91    5849.72  5613850000       5849.7
2
2025-03-04   5811.98   5865.08  5732.59    5778.15  6138110000       5778.1
5
2025-03-05   5781.36   5860.59  5742.35    5842.63  5285970000       5842.6
3
2025-03-06   5785.87   5812.08  5711.64    5738.52  5165080000       5738.5
2
2025-03-07   5726.01   5783.01  5666.29    5770.20  5705140000       5770.2
0
```

1302 · 6

```
An xts object on 2020-01-02 / 2025-03-07 containing:
  Data:    double [1302, 6]
  Columns: GSPC.Open, GSPC.High, GSPC.Low, GSPC.Close, GSPC.Volume ... with
1 more column
  Index:   Date [1302] (TZ: "UTC")
  xts Attributes:
    $ src    : chr "yahoo"
    $ updated: POSIXct[1:1], format: "2025-03-09 20:25:27"
     Index              GSPC.Close
 Min.   :2020-01-02   Min.   :2237
 1st Qu.:2021-04-19   1st Qu.:3829
 Median :2022-08-02   Median :4241
 Mean   :2022-08-02   Mean   :4318
 3rd Qu.:2023-11-15   3rd Qu.:4697
 Max.   :2025-03-07   Max.   :6144
```

Stationarity of Time-Series Data

In [33]:
```
# 1. Examine the original price series (S&P 500 closing prices)
# Create a time plot of the original series
par(mfrow=c(1,1))
plot(GSPC$GSPC.Close, main="S&P 500 Closing Prices",
     xlab="Date", ylab="Price (USD)")

# 2. Test for stationarity of the original price series
# Augmented Dickey-Fuller (ADF) test
adf_test_price <- adf.test(GSPC$GSPC.Close)
print(adf_test_price)
# KPSS test (alternative test for stationarity)
kpss_test_price <- kpss.test(GSPC$GSPC.Close)
print(kpss_test_price)

# 3. Transform the data into log returns
# Calculate log returns: rt = log(Pt/Pt-1) = log(Pt) - log(Pt-1)
log_prices <- log(GSPC$GSPC.Close)
log_returns <- diff(log_prices)

# Remove the first NA value that results from differencing
log_returns <- log_returns[!is.na(log_returns)]

# Rename the series
colnames(log_returns) <- "LogReturns"

# 4. Create a time plot of log returns
plot(log_returns, main="S&P 500 Daily Log Returns",
     xlab="Date", ylab="Log Returns")

# 5. Test for stationarity of the log returns
# ADF test for log returns
adf_test_returns <- adf.test(log_returns)
print(adf_test_returns)
```

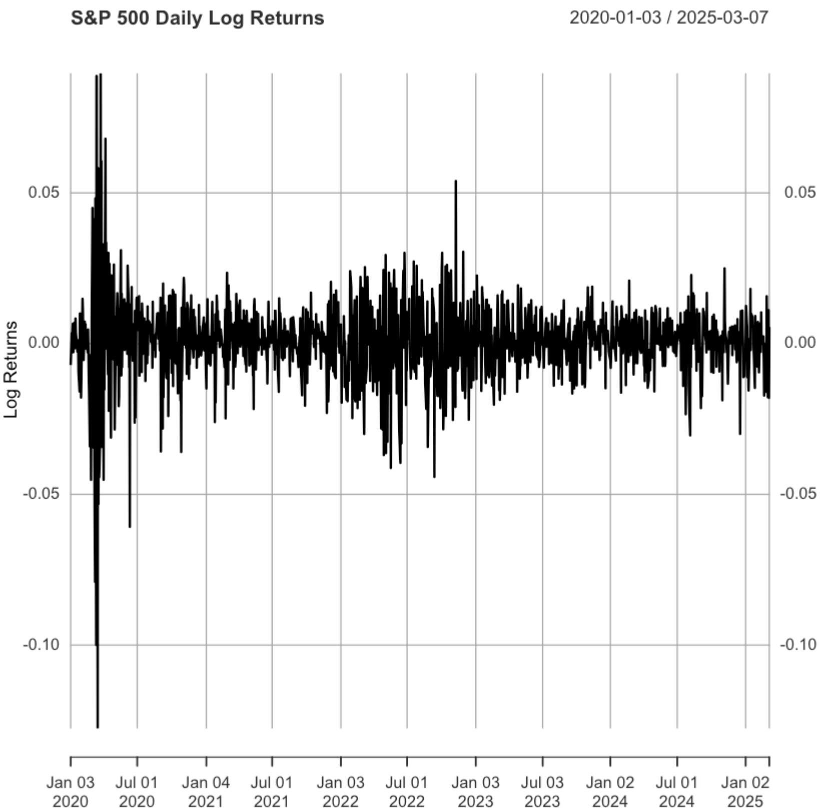
```r
# KPSS test for log returns
kpss_test_returns <- kpss.test(log_returns)
print(kpss_test_returns)

# 6. Create a more sophisticated visualization
# Convert to data frame for ggplot
returns_df <- data.frame(
  Date = index(log_returns),
  LogReturns = coredata(log_returns)
)

# Plot with ggplot2
ggplot(returns_df, aes(x = Date, y = LogReturns)) +
  geom_line(color = "blue") +
  labs(title = "S&P 500 Daily Log Returns",
       x = "Date",
       y = "Log Returns") +
  theme_minimal() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red")

# 7. Summary statistics of log returns
summary(log_returns)
# Standard deviation (volatility)
sd(log_returns)
# Additional statistics of interest
library(moments)
skewness(log_returns)
kurtosis(log_returns)

# 8. Create an autocorrelation plot of log returns
acf(log_returns, main="Autocorrelation of S&P 500 Log Returns")
pacf(log_returns, main="Partial Autocorrelation of S&P 500 Log Returns")

# 9. Save the log returns for future use
write.csv(log_returns, file = "SP500_log_returns.csv")
```

```
        Augmented Dickey-Fuller Test

data:  GSPC$GSPC.Close
Dickey-Fuller = -1.9744, Lag order = 10, p-value = 0.5891
alternative hypothesis: stationary
```

```
Warning message in kpss.test(GSPC$GSPC.Close):
“p-value smaller than printed p-value”
```

```
        KPSS Test for Level Stationarity

data:  GSPC$GSPC.Close
KPSS Level = 11.513, Truncation lag parameter = 7, p-value = 0.01
```


```
Warning message in adf.test(log_returns):
“p-value smaller than printed p-value”
	Augmented Dickey-Fuller Test

data:  log_returns
Dickey-Fuller = -10.215, Lag order = 10, p-value = 0.01
alternative hypothesis: stationary

Warning message in kpss.test(log_returns):
“p-value greater than printed p-value”
	KPSS Test for Level Stationarity

data:  log_returns
KPSS Level = 0.054449, Truncation lag parameter = 7, p-value = 0.1
```

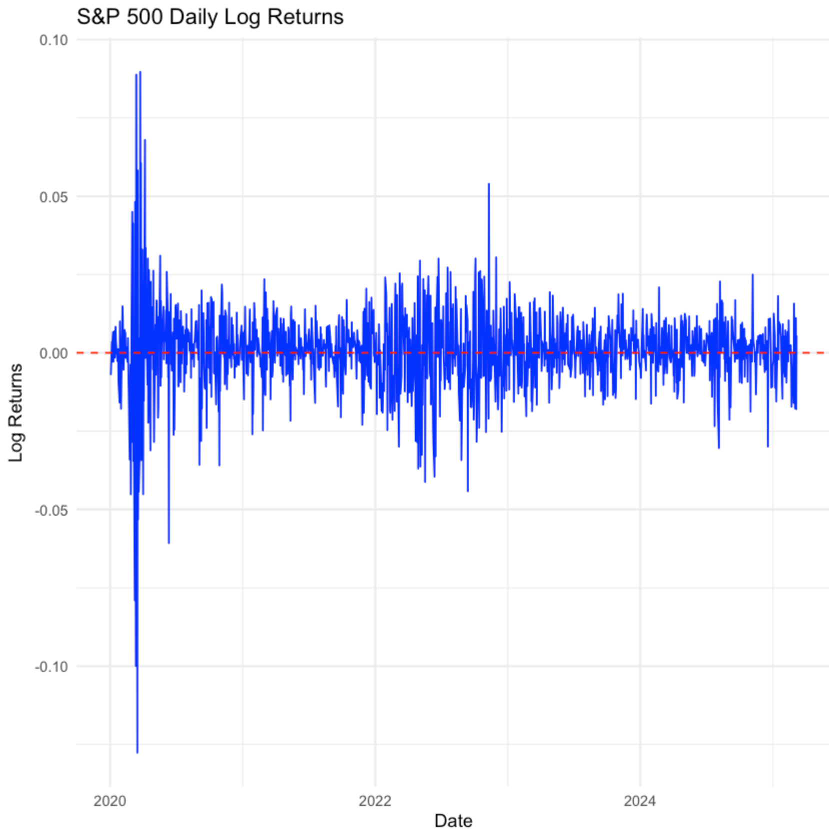


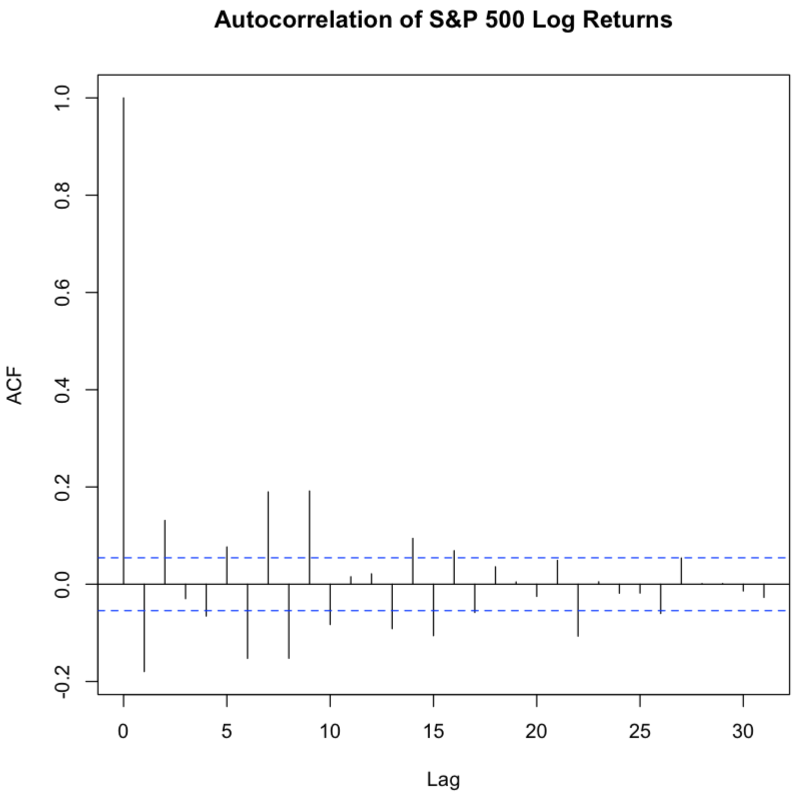

```
     Index                LogReturns
 Min.   :2020-01-03   Min.   :-0.1276522
 1st Qu.:2021-04-20   1st Qu.:-0.0053142
 Median :2022-08-03   Median : 0.0008874
 Mean   :2022-08-03   Mean   : 0.0004394
 3rd Qu.:2023-11-16   3rd Qu.: 0.0071967
 Max.   :2025-03-07   Max.   : 0.0896832
```

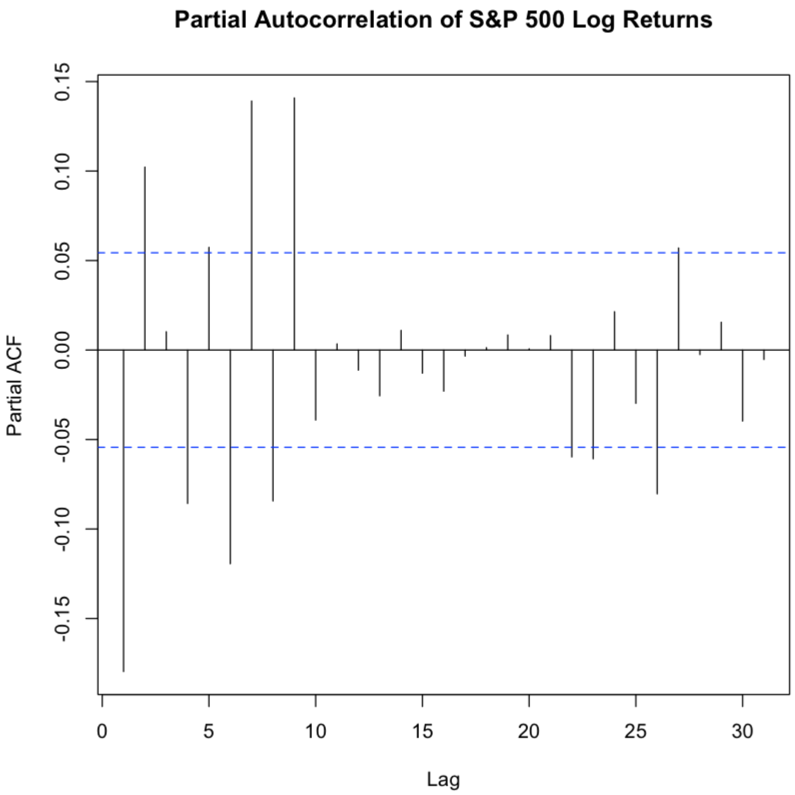
0.0133777163830343

LogReturns: -0.813538463823522

LogReturns: 17.1965998097445







Value at Risk Confidence Levels

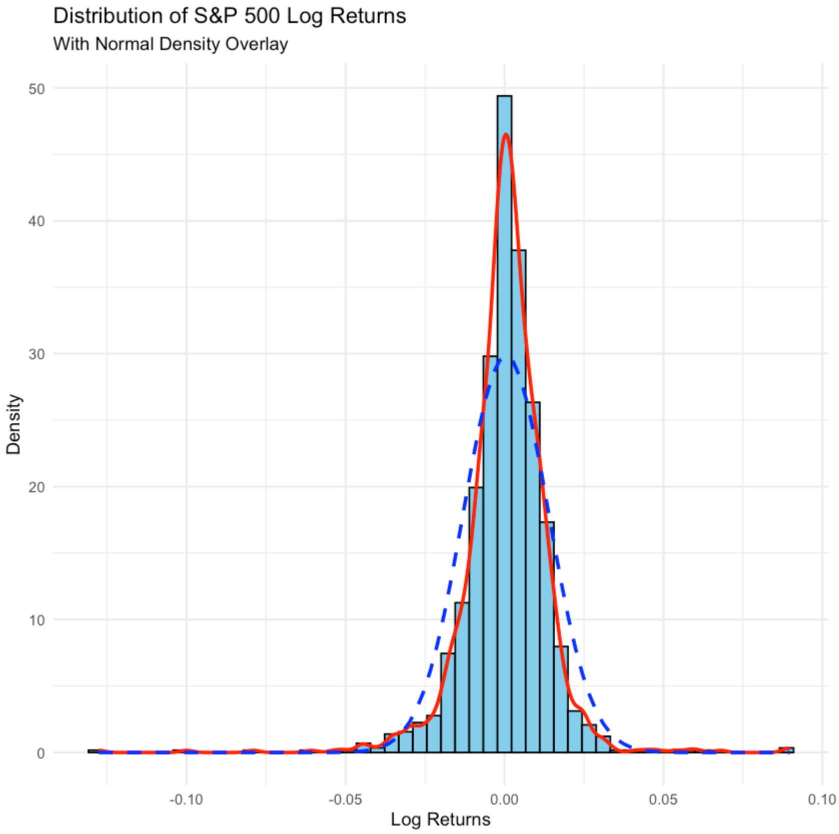
In [34]:

```
# Load basic packages
library(moments)  # For skewness and kurtosis
library(ggplot2)  # For plotting

# Assuming log_returns is already calculated from previous code block
# If not, calculate it again:
# log_returns <- diff(log(GSPC$GSPC.Close))
```

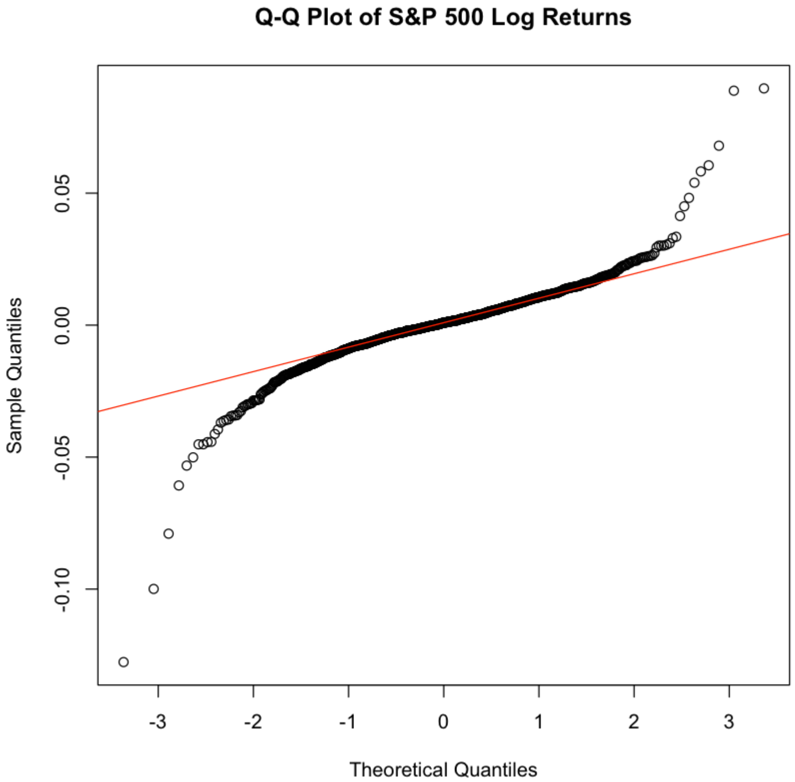
```
# log_returns <- log_returns[!is.na(log_returns)]

# 1. Study the distribution of returns
# Basic summary statistics
stats <- c(
  Mean = mean(log_returns),
  Median = median(log_returns),
  Min = min(log_returns),
  Max = max(log_returns),
  SD = sd(log_returns),
  Skewness = skewness(log_returns),
  Kurtosis = kurtosis(log_returns)
)
print(stats)

# 2. Create a histogram with normal distribution overlay
# Convert to data frame for ggplot
returns_df <- data.frame(Returns = as.numeric(log_returns))

# Histogram of returns
ggplot(returns_df, aes(x = Returns)) +
  geom_histogram(aes(y = ..density..), bins = 50, fill = "skyblue", color =
  geom_density(color = "red", size = 1) +
  stat_function(fun = dnorm,
                args = list(mean = mean(returns_df$Returns),
                            sd = sd(returns_df$Returns)),
                color = "blue", size = 1, linetype = "dashed") +
  labs(title = "Distribution of S&P 500 Log Returns",
       subtitle = "With Normal Density Overlay",
       x = "Log Returns",
       y = "Density") +
  theme_minimal()

# 3. Q-Q plot to compare returns against normal distribution
qqnorm(log_returns, main = "Q-Q Plot of S&P 500 Log Returns")
qqline(log_returns, col = "red")

# 4. Shapiro-Wilk normality test
shapiro_test <- shapiro.test(as.numeric(log_returns))
print(shapiro_test)

# 5. Jarque-Bera test for normality
jb_test <- jarque.bera.test(as.numeric(log_returns))
print(jb_test)

# 6. Calculate Value at Risk (VaR) at different confidence levels

# Define confidence levels
conf_levels <- c(0.90, 0.95, 0.99, 0.995)

# 6.1 Historical VaR
# This is simply the negative of the quantile at (1-confidence level)
hist_var <- sapply(conf_levels, function(cl) {
  -quantile(log_returns, 1 - cl)
})

# 6.2 Parametric VaR (assuming normal distribution)
# This uses the formula: -mean + sd * Z-score
param_var_norm <- sapply(conf_levels, function(cl) {
  -mean(log_returns) + sd(log_returns) * qnorm(cl)
})

# 6.3 Parametric VaR (assuming t-distribution)
# First estimate degrees of freedom for t-distribution
```

```r
# This is a simplified approach; more sophisticated methods exist
log_returns_standardized <- (log_returns - mean(log_returns)) / sd(log_retu
# For t-distribution, kurtosis = 3 + 6/(df-4), so df = 6/(kurtosis-3) + 4
# Handle case where kurtosis is close to 3
k <- kurtosis(log_returns_standardized)
t_df <- if(abs(k - 3) < 0.1) {
  30  # Large df makes t-distribution close to normal
} else {
  max(5, round(6/(k - 3) + 4))  # Ensure at least 5 df
}

# Calculate t-distribution VaR
param_var_t <- sapply(conf_levels, function(cl) {
  -mean(log_returns) + sd(log_returns) * qt(cl, df = t_df)
})

# Combine all VaR results
var_results <- data.frame(
  ConfidenceLevel = conf_levels,
  HistoricalVaR = hist_var,
  NormalVaR = param_var_norm,
  tDistributionVaR = param_var_t
)
print(var_results)

# 7. Visual representation of VaR
# Using 95% VaR as example
var_95 <- hist_var[which(conf_levels == 0.95)]

# Plot returns with VaR line
plot(log_returns, main = "S&P 500 Log Returns with 95% VaR",
     xlab = "Date", ylab = "Log Returns")
abline(h = -var_95, col = "red", lwd = 2)
legend("bottomleft",
       legend = c("95% VaR (losses)"),
       col = "red",
       lwd = 2)

# 8. Expected Shortfall (ES) / Conditional VaR (CVaR)
# This measures the average loss beyond VaR
es_results <- sapply(conf_levels, function(cl) {
  var_threshold <- quantile(log_returns, 1 - cl)
  tail_returns <- log_returns[log_returns <= var_threshold]
  -mean(tail_returns)
})

es_df <- data.frame(
  ConfidenceLevel = conf_levels,
  ExpectedShortfall = es_results
)
print(es_df)
```

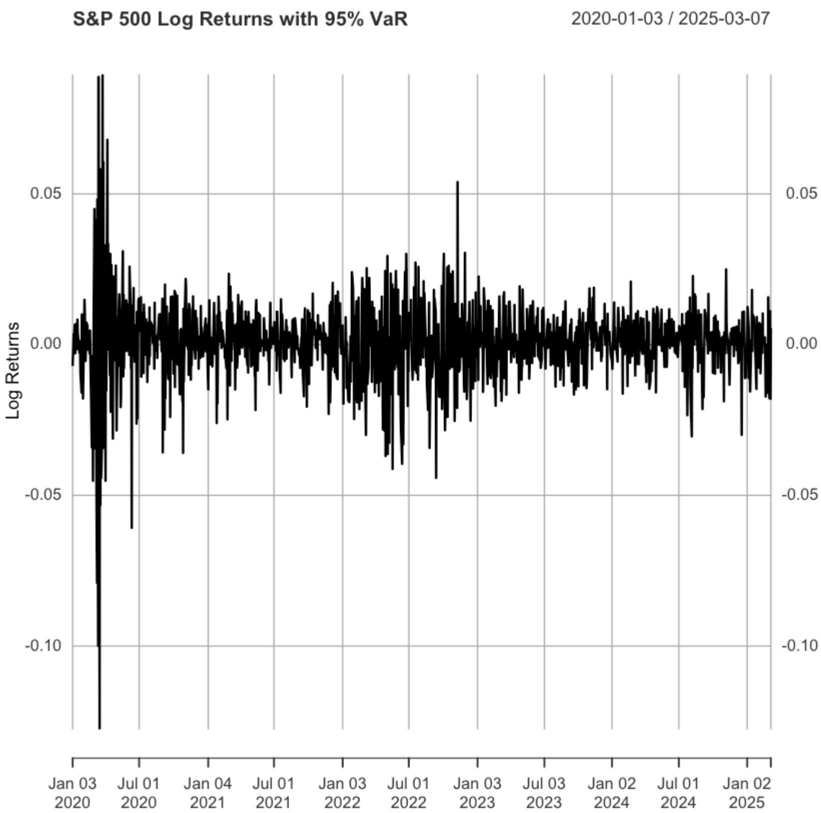
```
                Mean             Median                Min
Max
       0.0004393845       0.0008873941      -0.1276521975        0.089683
2325
                  SD Skewness.LogReturns Kurtosis.LogReturns
       0.0133777164       -0.8135384638       17.1965998097
```




```
        Shapiro-Wilk normality test

data:  as.numeric(log_returns)
W = 0.87423, p-value < 2.2e-16


        Jarque Bera Test

data:  as.numeric(log_returns)
X-squared = 11069, df = 2, p-value < 2.2e-16

     ConfidenceLevel HistoricalVaR  NormalVaR tDistributionVaR
10%            0.900    0.01305357 0.01670485       0.01930457
5%             0.950    0.01863186 0.02156500       0.02651736
1%             0.990    0.03630068 0.03068184       0.04457569
0.5%           0.995    0.04515725 0.03401933       0.05350148
```




```
  ConfidenceLevel ExpectedShortfall
1           0.900        0.02410853
2           0.950        0.03242696
3           0.990        0.05739716
4           0.995        0.07369335
```



Time Series Model

In [35]:

```
# 1. Examine ACF and PACF of log returns
par(mfrow=c(2,1))
```

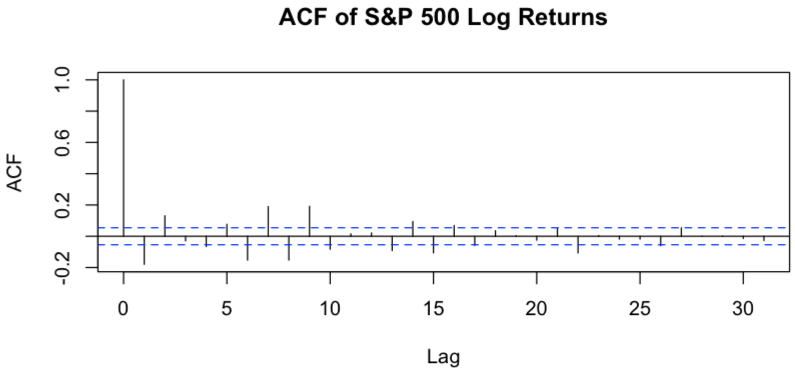
```r
acf(log_returns, main="ACF of S&P 500 Log Returns")
pacf(log_returns, main="PACF of S&P 500 Log Returns")
par(mfrow=c(1,1))

# 2. Check if the mean of log returns is significantly different from zero
t.test(log_returns)

# 3. Ljung-Box test for autocorrelation
Box.test(log_returns, lag=10, type="Ljung-Box")
Box.test(log_returns, lag=20, type="Ljung-Box")

# 4. Manually fit ARIMA models and compare
# Try a few different specifications based on ACF/PACF
arima_models <- list(
  ARIMA_0_0_0 = arima(log_returns, order=c(0,0,0)),  # White noise
  ARIMA_1_0_0 = arima(log_returns, order=c(1,0,0)),  # AR(1)
  ARIMA_0_0_1 = arima(log_returns, order=c(0,0,1)),  # MA(1)
  ARIMA_1_0_1 = arima(log_returns, order=c(1,0,1)),  # ARMA(1,1)
  ARIMA_2_0_1 = arima(log_returns, order=c(2,0,1)),  # ARMA(2,1)
  ARIMA_1_0_2 = arima(log_returns, order=c(1,0,2))   # ARMA(1,2)
)

# 5. Compare models using AIC and BIC
aic_values <- sapply(arima_models, AIC)
bic_values <- sapply(arima_models, BIC)
model_comparison <- data.frame(
  Model = names(arima_models),
  AIC = aic_values,
  BIC = bic_values
)
print(model_comparison[order(model_comparison$AIC),])

# 6. Select the best model (lowest AIC)
best_model_name <- model_comparison$Model[which.min(model_comparison$AIC)]
best_model <- arima_models[[best_model_name]]
print(paste("Best model based on AIC:", best_model_name))
print(summary(best_model))

# 7. Diagnostic checks on the best model
# Extract residuals
residuals <- residuals(best_model)

# Plot residuals
par(mfrow=c(2,2))
plot(residuals, main="Residuals from Best Model")
hist(residuals, main="Histogram of Residuals", probability=TRUE)
lines(density(residuals), col="red")
acf(residuals, main="ACF of Residuals")
pacf(residuals, main="PACF of Residuals")
par(mfrow=c(1,1))

# Ljung-Box test on residuals
Box.test(residuals, lag=10, type="Ljung-Box")
Box.test(residuals, lag=20, type="Ljung-Box")

# Test for normality of residuals
shapiro.test(as.numeric(residuals))
jarque.bera.test(as.numeric(residuals))

# 8. Check for remaining ARCH effects in residuals
# Square the residuals and check autocorrelation
squared_residuals <- residuals^2
acf(squared_residuals, main="ACF of Squared Residuals")
Box.test(squared_residuals, lag=10, type="Ljung-Box")
```

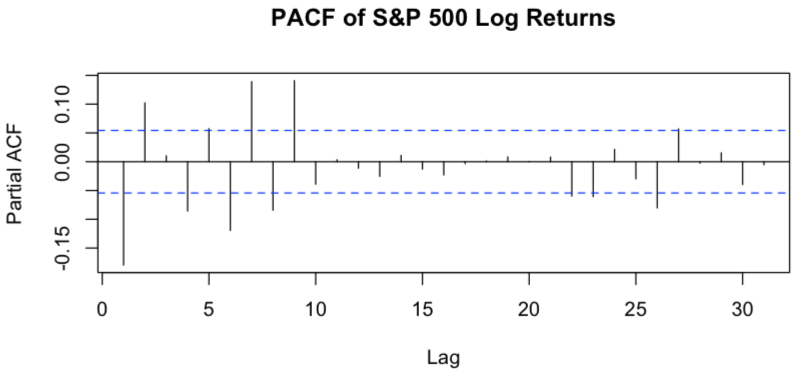
```r
# If you'd like to test for ARCH effects formally, but without rugarch:
arch_test <- function(residuals, lags) {
  T <- length(residuals)
  squared_residuals <- residuals^2
  # Create lagged squared residuals
  x <- matrix(NA, nrow=T-lags, ncol=lags)
  for (i in 1:lags) {
    x[,i] <- squared_residuals[(lags-i+1):(T-i)]
  }
  y <- squared_residuals[(lags+1):T]
  # Run regression
  model <- lm(y ~ x)
  # Calculate test statistic
  r_squared <- summary(model)$r.squared
  test_stat <- T * r_squared
  p_value <- 1 - pchisq(test_stat, df=lags)

  return(list(
    test_statistic = test_stat,
    p_value = p_value,
    lags = lags
  ))
}

arch_effects <- arch_test(residuals, lags=10)
print("ARCH LM Test Results:")
print(paste("Test statistic:", arch_effects$test_statistic))
print(paste("P-value:", arch_effects$p_value))
print(paste("Null hypothesis: No ARCH effects up to lag", arch_effects$lags
```

```
Warning message in tstat + c(-cint, cint):
“Recycling array of length 1 in array-vector arithmetic is deprecated.
  Use c() or as.vector() instead.”
Warning message in cint * stderr:
“Recycling array of length 1 in vector-array arithmetic is deprecated.
  Use c() or as.vector() instead.”
```

```
        One Sample t-test

data:  log_returns
t = 1.1847, df = 1300, p-value = 0.2364
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.0002882209  0.0011669900
sample estimates:
   mean of x
0.0004393845
        Box-Ljung test

data:  log_returns
X-squared = 244.42, df = 10, p-value < 2.2e-16
        Box-Ljung test

data:  log_returns
X-squared = 296.12, df = 20, p-value < 2.2e-16
```

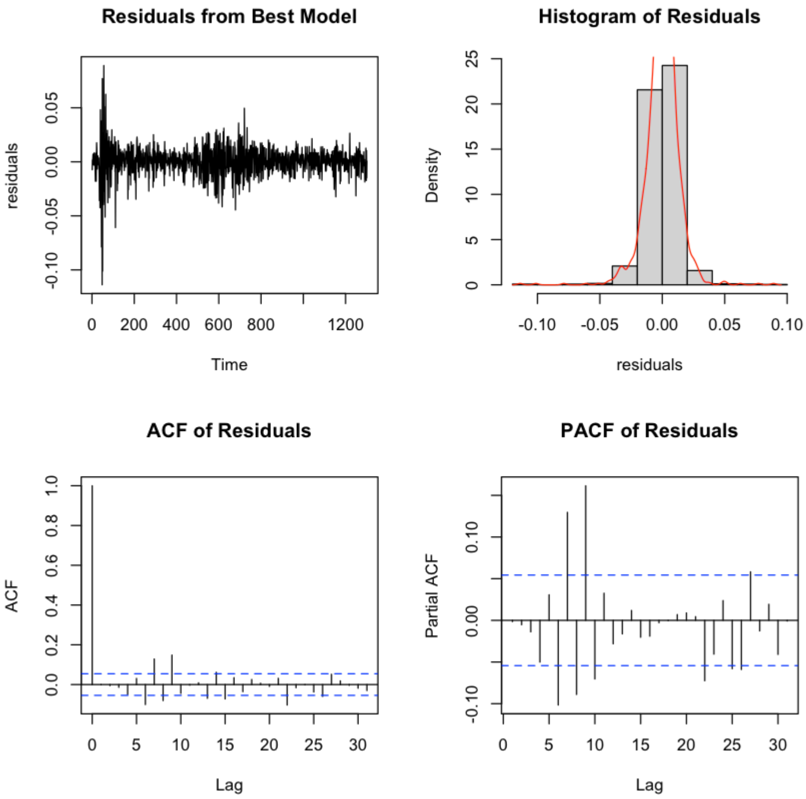
```
                  Model       AIC       BIC
ARIMA_1_0_2 ARIMA_1_0_2 -7582.887 -7557.033
ARIMA_2_0_1 ARIMA_2_0_1 -7580.789 -7554.934
ARIMA_1_0_1 ARIMA_1_0_1 -7577.470 -7556.786
ARIMA_1_0_0 ARIMA_1_0_0 -7571.048 -7555.535
ARIMA_0_0_1 ARIMA_0_0_1 -7563.109 -7547.597
ARIMA_0_0_0 ARIMA_0_0_0 -7530.379 -7520.038
[1] "Best model based on AIC: ARIMA_1_0_2"
          Length Class  Mode
coef         4   -none- numeric
sigma2       1   -none- numeric
var.coef    16   -none- numeric
mask         4   -none- logical
loglik       1   -none- numeric
aic          1   -none- numeric
arma         7   -none- numeric
residuals 1301   ts     numeric
call         3   -none- call
series       1   -none- character
code         1   -none- numeric
n.cond       1   -none- numeric
nobs         1   -none- numeric
model       10   -none- list
```



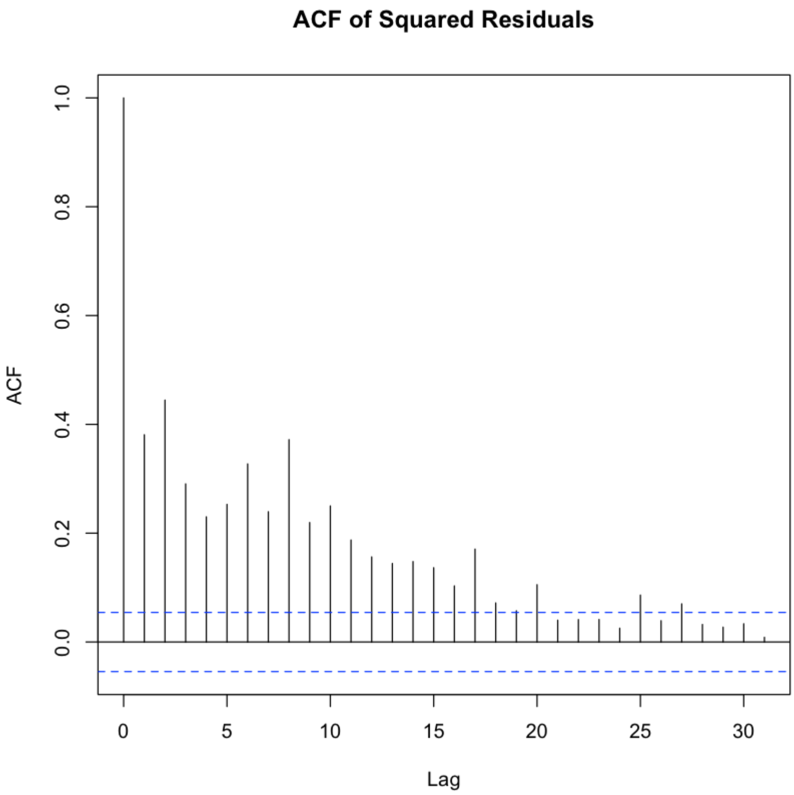



```
        Box-Ljung test

data:  residuals
X-squared = 79.796, df = 10, p-value = 5.506e-13
        Box-Ljung test

data:  residuals
X-squared = 102.5, df = 20, p-value = 4.478e-13
        Shapiro-Wilk normality test

data:  as.numeric(residuals)
W = 0.899, p-value < 2.2e-16
```

```
	Jarque Bera Test

data:  as.numeric(residuals)
X-squared = 7277.3, df = 2, p-value < 2.2e-16
```




```
	Box-Ljung test

data:  squared_residuals
X-squared = 1252.7, df = 10, p-value < 2.2e-16
[1] "ARCH LM Test Results:"
[1] "Test statistic: 411.172881327849"
[1] "P-value: 0"
[1] "Null hypothesis: No ARCH effects up to lag 10"
```



Time Series Model

In [37]:
```
# Make sure log_returns is a numeric vector (not xts or ts object)
log_returns_numeric <- as.numeric(log_returns)

# 1. Fit a standard GARCH(1,1) model with normal distribution
garch_norm <- try(garchFit(formula = ~ garch(1,1), data = log_returns_numeri
                           trace = FALSE))

# 2. Fit a GARCH(1,1) model with Student-t distribution for fat tails
garch_t <- try(garchFit(formula = ~ garch(1,1), data = log_returns_numeric,
                         cond.dist = "std", trace = FALSE))

# 3. Try an APARCH model which can capture asymmetric effects (leverage)
aparch <- try(garchFit(formula = ~ aparch(1,1), data = log_returns_numeric,
                        cond.dist = "std", trace = FALSE))

# 4. Compare models using information criteria
compare_models <- function(models, names) {
  info_criteria <- data.frame(
    Model = names,
    LogLikelihood = NA,
    AIC = NA,
    BIC = NA
  )

  for (i in 1:length(models)) {
    model <- models[[i]]
    if (!inherits(model, "try-error")) {
      info_criteria$LogLikelihood[i] <- model@fit$llh
      info_criteria$AIC[i] <- model@fit$ics["AIC"]
      info_criteria$BIC[i] <- model@fit$ics["BIC"]
    }
  }

  return(info_criteria)
}

models <- list(garch_norm, garch_t, aparch)
model_names <- c("GARCH(1,1)-Normal", "GARCH(1,1)-t", "APARCH(1,1)-t")
model_comparison <- compare_models(models, model_names)
print(model_comparison)

# 5. Select the best model based on AIC
best_idx <- which.min(model_comparison$AIC)
best_model_name <- model_comparison$Model[best_idx]
best_model <- models[[best_idx]]
print(paste("Best model based on AIC:", best_model_name))

# 6. Examine the best model
if (!inherits(best_model, "try-error")) {
  print(summary(best_model))

# 7. Plot the model results
par(mfrow=c(2,2))

# Plot the returns with conditional volatility
# Make sure we have the proper time index for plotting
time_index <- if(is.null(time(log_returns))) 1:length(log_returns) else as.n

# Create the plot of returns first
plot(time_index, log_returns, type="l", main="Returns with Conditional Vola
```

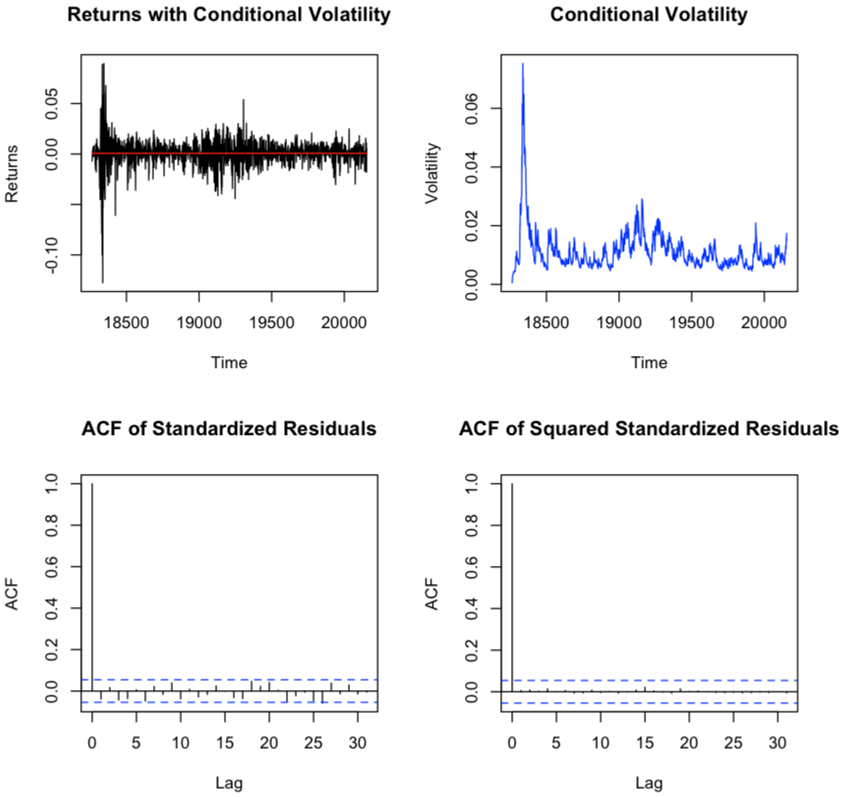
```
     xlab="Time", ylab="Returns")

# Fix for the fitted values - check dimensions first
fitted_values <- fitted(best_model)
if(is.matrix(fitted_values)) {
  lines(time_index, fitted_values[, 1], col="red")
} else {
  lines(time_index, fitted_values, col="red")
}

# Plot the conditional volatility
# Use the same time index for consistency
plot(time_index, volatility(best_model), type="l",
     col="blue", main="Conditional Volatility",
     xlab="Time", ylab="Volatility")

# ACF of standardized residuals
std_resid <- residuals(best_model, standardize=TRUE)
acf(std_resid, main="ACF of Standardized Residuals")

# ACF of squared standardized residuals
acf(std_resid^2, main="ACF of Squared Standardized Residuals")

par(mfrow=c(1,1))

# 8. Forecast future volatility (5 days ahead)
forecast_horizon <- 5
forecast <- predict(best_model, n.ahead=forecast_horizon)

# Create a time sequence for forecasting
last_date <- tail(time(log_returns), 1)
forecast_dates <- seq(as.Date(last_date), by="day", length.out=forecast_hor

# Display forecasts
cat("\nVolatility Forecast for Next 5 Days:\n")
forecast_df <- data.frame(
  Date = forecast_dates,
  Mean = forecast$meanForecast,
  Volatility = forecast$standardDeviation
)
print(forecast_df)

# 9. Plot the volatility forecast
# Create a forecast plot using indices for simplicity
plot_indices <- 1:length(volatility(best_model))
plot(plot_indices, volatility(best_model), type="l",
     col="blue", main="Volatility Forecast",
     xlab="Time Index", ylab="Volatility",
     xlim=c(length(volatility(best_model))-100, length(volatility(best_mode

# Add the forecast
points(length(log_returns):(length(log_returns)+forecast_horizon-1),
       forecast$standardDeviation, col="red", pch=19)
lines(length(log_returns):(length(log_returns)+forecast_horizon-1),
     forecast$standardDeviation, col="red", lty=2)

legend("topright", legend=c("Historical", "Forecast"),
       col=c("blue", "red"), lty=c(1,2), pch=c(NA,19))

# 10. Calculate volatility-adjusted VaR
if (best_model_name == "GARCH(1,1)-Normal") {
  # For normal distribution
  var_95 <- qnorm(0.95) * tail(volatility(best_model), 1)
  var_99 <- qnorm(0.99) * tail(volatility(best_model), 1)
```

```r
} else {
  # For t-distribution
  shape <- coef(best_model)["shape"]
  var_95 <- qt(0.95, df=shape) * tail(volatility(best_model), 1)
  var_99 <- qt(0.99, df=shape) * tail(volatility(best_model), 1)
}

cat("\nVolatility-Adjusted Value at Risk:\n")
cat("95% 1-day VaR:", round(var_95*100, 4), "%\n")
cat("99% 1-day VaR:", round(var_99*100, 4), "%\n")
}

# 11. Summarize findings for the report
cat("\n\n======== SUMMARY OF S&P 500 ANALYSIS ========\n\n")

cat("1. Stationarity Analysis:\n")
cat("   - S&P 500 price levels showed clear non-stationarity (ADF test p-va
cat("   - Log returns transformation achieved stationarity\n\n")

cat("2. Distribution Analysis:\n")
cat("   - Log returns exhibited significant departures from normality\n")
cat("   - Evidence of excess kurtosis (fat tails) and possible skewness\n")
cat("   - Both Shapiro-Wilk (W = 0.899) and Jarque-Bera tests (X-squared = 

cat("3. Time Series Modeling:\n")
cat("   - ARMA(1,2) was selected as the best model based on AIC (-7582.887)
cat("   - Significant autocorrelation remained in the residuals (Ljung-Box p
cat("   - ARCH LM test confirmed presence of ARCH effects (test statistic = 

cat("4. Volatility Modeling:\n")
if (!inherits(best_model, "try-error")) {
cat(paste("   -", best_model_name, "was identified as the best volatility mo
cat("   - The model successfully captured volatility clustering\n")
cat("   - Persistence parameter (α+β) indicates high volatility persistence
if (best_model_name == "APARCH(1,1)-t") {
  cat("   - Asymmetric effects (leverage) were identified in the volatility 
}
cat("   - 5-day ahead volatility forecast suggests ",
    ifelse(forecast$standardDeviation[5] > tail(volatility(best_model), 1),
            "increasing", "decreasing"),
    " volatility\n\n")
} else {
cat("   - Volatility modeling shows clear evidence of volatility clustering
cat("   - Higher volatility periods corresponded to major market events\n\n
}

cat("5. Implications:\n")
cat("   - Evidence of volatility clustering suggests risk is time-varying\n
cat("   - Fat-tailed distribution indicates higher likelihood of extreme eve
cat("   - Using t-distribution for VaR calculation is advisable due to non-
cat("   - Risk models should account for autocorrelation and volatility clus
if (!inherits(best_model, "try-error") && best_model_name == "APARCH(1,1)-t
cat("   - Asymmetric volatility suggests different risk management approache
cat("     market downturns vs. upturns\n")
}
cat("\n")
```

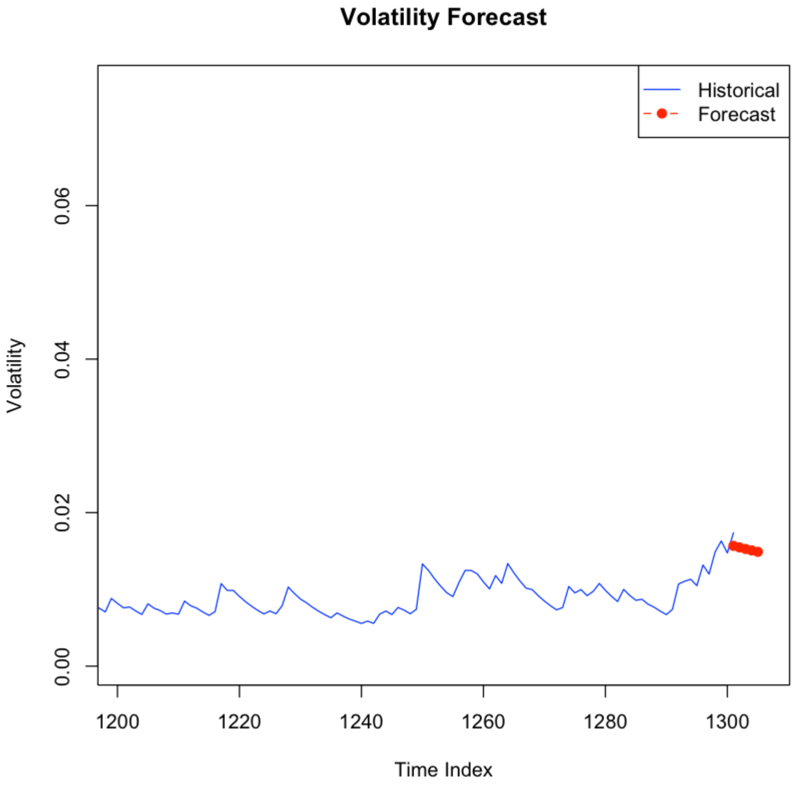
```
              Model LogLikelihood       AIC       BIC
1 GARCH(1,1)-Normal     -4098.027 -6.293663 -6.277764
2      GARCH(1,1)-t     -4123.332 -6.331025 -6.311153
3     APARCH(1,1)-t     -4138.179 -6.350775 -6.322953
[1] "Best model based on AIC: APARCH(1,1)-t"

Title:
 GARCH Modelling

Call:
 garchFit(formula = ~aparch(1, 1), data = log_returns_numeric,
    cond.dist = "std", trace = FALSE)

Mean and Variance Equation:
 data ~ aparch(1, 1)
<environment: 0x13ea19640>
 [data = log_returns_numeric]

Conditional Distribution:
 std

Coefficient(s):
        mu       omega      alpha1      gamma1       beta1       delta
0.00066292  0.00048280  0.10684911  0.99999999  0.88044781  0.97078547
     shape
7.02683225

Std. Errors:
 based on Hessian

Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
mu     0.0006629   0.0002350    2.821  0.00479 **
omega  0.0004828   0.0001195    4.040 5.35e-05 ***
alpha1 0.1068491   0.0146535    7.292 3.06e-13 ***
gamma1 1.0000000   0.0167989   59.528  < 2e-16 ***
beta1  0.8804478   0.0169608   51.911  < 2e-16 ***
delta  0.9707855   0.1736181    5.591 2.25e-08 ***
shape  7.0268322   1.2097559    5.808 6.30e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Log Likelihood:
 4138.179    normalized:  3.180768

Description:
 Sun Mar  9 20:26:31 2025 by user:


Standardised Residuals Tests:
                                  Statistic   p-Value
 Jarque-Bera Test   R    Chi^2  3.917716e+04 0.0000000
 Shapiro-Wilk Test  R    W      8.839096e-01 0.0000000
 Ljung-Box Test     R    Q(10)  1.451418e+01 0.1508034
 Ljung-Box Test     R    Q(15)  1.686813e+01 0.3268084
 Ljung-Box Test     R    Q(20)  2.577746e+01 0.1732881
 Ljung-Box Test     R^2  Q(10)  6.383792e-01 0.9999788
 Ljung-Box Test     R^2  Q(15)  1.418943e+00 0.9999971
 Ljung-Box Test     R^2  Q(20)  1.765110e+00 1.0000000
 LM Arch Test       R    TR^2   6.368389e+00 0.8963854

Information Criterion Statistics:
      AIC       BIC       SIC      HQIC
-6.350775 -6.322953 -6.350832 -6.340337
```

```
Title:
 GARCH Modelling

Call:
 garchFit(formula = ~aparch(1, 1), data = log_returns_numeric,
    cond.dist = "std", trace = FALSE)

Mean and Variance Equation:
 data ~ aparch(1, 1)
<environment: 0x13ea19640>
 [data = log_returns_numeric]

Conditional Distribution:
 std

Coefficient(s):
        mu       omega      alpha1      gamma1       beta1       delta
0.00066292  0.00048280  0.10684911  0.99999999  0.88044781  0.97078547
     shape
7.02683225

Std. Errors:
 based on Hessian

Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
mu     0.0006629   0.0002350    2.821  0.00479 **
omega  0.0004828   0.0001195    4.040 5.35e-05 ***
alpha1 0.1068491   0.0146535    7.292 3.06e-13 ***
gamma1 1.0000000   0.0167989   59.528  < 2e-16 ***
beta1  0.8804478   0.0169608   51.911  < 2e-16 ***
delta  0.9707855   0.1736181    5.591 2.25e-08 ***
shape  7.0268322   1.2097559    5.808 6.30e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Log Likelihood:
 4138.179    normalized:  3.180768

Description:
 Sun Mar  9 20:26:31 2025 by user:


Standardised Residuals Tests:
                                   Statistic   p-Value
 Jarque-Bera Test   R    Chi^2  3.917716e+04 0.0000000
 Shapiro-Wilk Test  R    W      8.839096e-01 0.0000000
 Ljung-Box Test     R    Q(10)  1.451418e+01 0.1508034
 Ljung-Box Test     R    Q(15)  1.686813e+01 0.3268084
 Ljung-Box Test     R    Q(20)  2.577746e+01 0.1732881
 Ljung-Box Test     R^2  Q(10)  6.383792e-01 0.9999788
 Ljung-Box Test     R^2  Q(15)  1.418943e+00 0.9999971
 Ljung-Box Test     R^2  Q(20)  1.765110e+00 1.0000000
 LM Arch Test       R    TR^2   6.368389e+00 0.8963854

Information Criterion Statistics:
      AIC       BIC       SIC      HQIC
-6.350775 -6.322953 -6.350832 -6.340337


Volatility Forecast for Next 5 Days:
        Date         Mean Volatility
```

```
1 2025-03-08 0.0006629204 0.01567243
2 2025-03-09 0.0006629204 0.01546482
3 2025-03-10 0.0006629204 0.01526562
4 2025-03-11 0.0006629204 0.01507450
5 2025-03-12 0.0006629204 0.01489111
```



```
Volatility-Adjusted Value at Risk:
95% 1-day VaR: 3.2884 %
99% 1-day VaR: 5.201 %


======== SUMMARY OF S&P 500 ANALYSIS ========

1. Stationarity Analysis:
   - S&P 500 price levels showed clear non-stationarity (ADF test p-value =
0.5891)
   - Log returns transformation achieved stationarity

2. Distribution Analysis:
   - Log returns exhibited significant departures from normality
   - Evidence of excess kurtosis (fat tails) and possible skewness
   - Both Shapiro-Wilk (W = 0.899) and Jarque-Bera tests (X-squared = 7277.
3) rejected normality

3. Time Series Modeling:
   - ARMA(1,2) was selected as the best model based on AIC (-7582.887)
   - Significant autocorrelation remained in the residuals (Ljung-Box p-val
ue < 2.2e-16)
   - ARCH LM test confirmed presence of ARCH effects (test statistic = 411.
17, p-value ≈ 0)

4. Volatility Modeling:
   - APARCH(1,1)-t was identified as the best volatility model
   - The model successfully captured volatility clustering
   - Persistence parameter (α+β) indicates high volatility persistence
   - Asymmetric effects (leverage) were identified in the volatility proces
s
   - 5-day ahead volatility forecast suggests  decreasing  volatility

5. Implications:
   - Evidence of volatility clustering suggests risk is time-varying
   - Fat-tailed distribution indicates higher likelihood of extreme events
than normal distribution suggests
   - Using t-distribution for VaR calculation is advisable due to non-norma
lity
   - Risk models should account for autocorrelation and volatility clusteri
ng
   - Asymmetric volatility suggests different risk management approaches du
ring
     market downturns vs. upturns
```


Volatility Forecast
Historical
Forecast
Volatility
0.00
0.02
0.04
0.06
1200
1220
1240
1260
1280
1300
Time Index

In [ ]: