AI6123: REPORT For Project 3

Names: Joseph UWINEZA
Matric No: G2303477F
Program: MSAI

NTU, May 2024

PROJECT 3

This project consists of two parts. 1) Part 1 is to analyze financial data. The data are from the daily historical Apple stock prices(open, high, low, close and adjusted prices) from February 1, 2002 to January 31, 2017 extracted from the Yahoo Finance website. The data has logged the prices of the Apple stock everyday and comprises of the open, close, low, high and the adjusted close prices of the stock for the span of 15 years.The goal of the project is to discover an interesting trend in the apple stock prices over the past 15 years (3775 attributes) and to design and develop the best model for forecasting.

2) Part 2 is to answer the questions in the pdf document. It is due on 2 May, 2024. Please submit it via this folder.

Abstract

This project analyzes the daily historical stock prices of Apple Inc[4]. From February 1, 2002, to January 31, 2017, sourced from Yahoo Finance, encompassing open, high, low, close, and adjusted close prices. The primary objective is to uncover significant trends in Apple's stock prices over the 15-year period and to develop a robust model for forecasting future stock movements. Traditional linear structural models, commonly used in time series analysis, often fall short in explaining the nuanced dynamics of financial data, such as the conditional variance crucial for financial modeling. Consequently, this research focuses on modeling the conditional variance structure, which is integral to understanding the risk associated with financial assets and is a fundamental aspect of the mathematical theory of asset pricing and Value at Risk (VaR) calculations.

1 Introduction

This project aims to develop a sophisticated model for forecasting Apple Inc.'s stock prices by focusing on the conditional variance of financial data, which traditional linear structural models often fail to address adequately. Our analysis begins with a thorough examination of the historical stock prices of Apple Inc [4]., sourced from Yahoo Finance for the period from February 1, 2002, to January 31, 2017. This dataset includes various metrics such as open, high, low, close, and adjusted close prices. For the purposes of this project, I have chosen to focus specifically on the Adjusted Closing Price due to its relevance in reflecting the true value of the stock, adjusted for dividends and splits. The initial visual representation of this data, crucial for identifying underlying trends and volatilities, is depicted in Figure 1. Our approach aims to uncover significant patterns and develop a robust forecasting model that incorporates the nuanced dynamics of Apple's stock prices over the analyzed 15-year period.

Figure 1: Shows the plot of original data, it is clear that it is not stationary with mean and variance of 10.78, and 101.79 respectively, with 33.25 maximum and 0.23 minimum.

The initial examination of the data plot reveals a clear increasing trend paired with a pattern of volatility, a characteristic often referred to as 'Volatility Clustering,' commonly observed in financial data curves. Moreover, this increasing trend displays a non-linear nature, with a sharp escalation in the later stages. Regarding seasonal patterns, the data appears to cycle every 12 months. To delve deeper, we will first transform the data into a monthly format, then apply seasonal decomposition using the stl() function to uncover more intricate details. The results of this process are depicted in Figure 2.

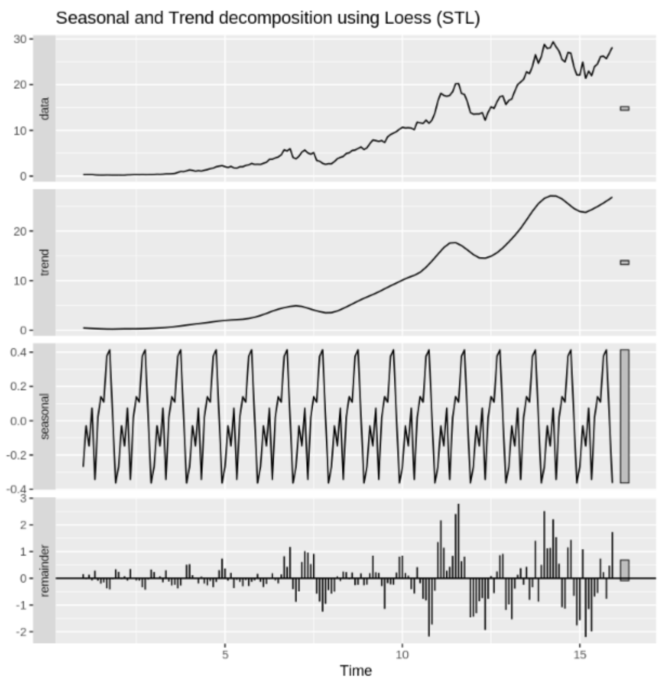


Figure 2: Seasonal Decomposition on Original Data.

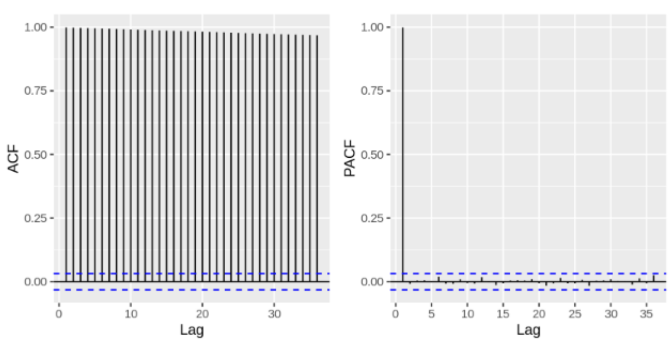


Figure 3: Shows ACF (left) and PACF (Right) of the original data

From the data plot and the remainder plot shown in Figure 3, a significant increase in seasonality variance is evident. Consequently, it would be prudent to apply the Box-Cox transformation to address this variance. Additionally, the results from the Augmented Dickey-Fuller Test, with a p-value of 0.4114, indicate that the data is non-stationary.

2 Data preprocessing

2.1 Transformation

The Box-Cox transformation[2], with lambda set to zero for a logarithmic effect, is used to stabilize variance and reduce skewness in financial time series data, making multiplicative relationships additive. First-order differencing further stabilizes the data by eliminating trends. The transformed data is scaled by 100 to represent percentage changes, shown in Figure 4, aiding in the interpretation of proportional variations. The stationarity of the data is confirmed by an ADF test p-value of 0.01, validating the transformation's effectiveness for subsequent analysis.

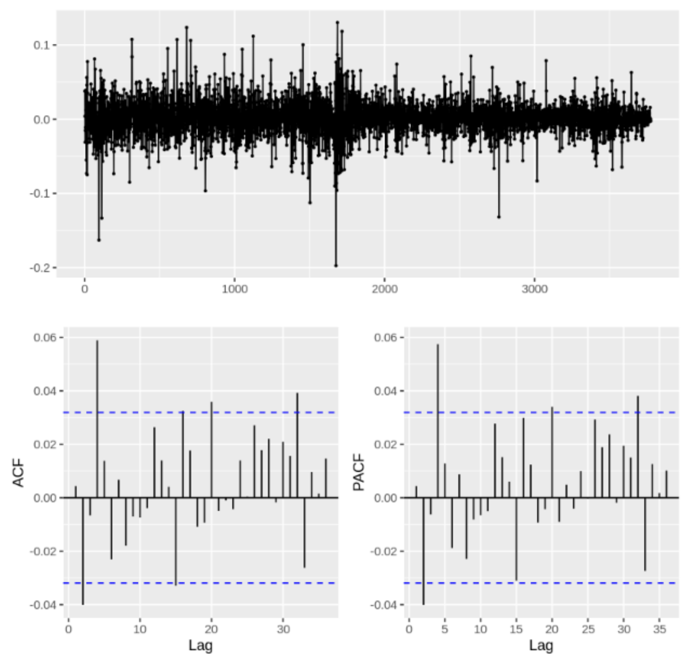


Figure 4: Log return - Data after Transformers.

Given the observed volatility clustering in the log-return data, it is necessary to further examine the ACF/-PACF plots for both squared log-return data, and absolute, as displayed in Figures 5 and 6. These plots reveal that the returns are not independently and identically distributed. Additionally, the QQ-plot, shown in Figure 7, is used to analyze the shape of the distribution of AAPL returns, indicating a heavy-tailed distribution that skews to the left. This observation is further supported by kurtosis and skewness tests, yielding values of 5.4356 and -0.1900, respectively, confirming the heavy-tailed and left-skewed characteristics of the returns data.

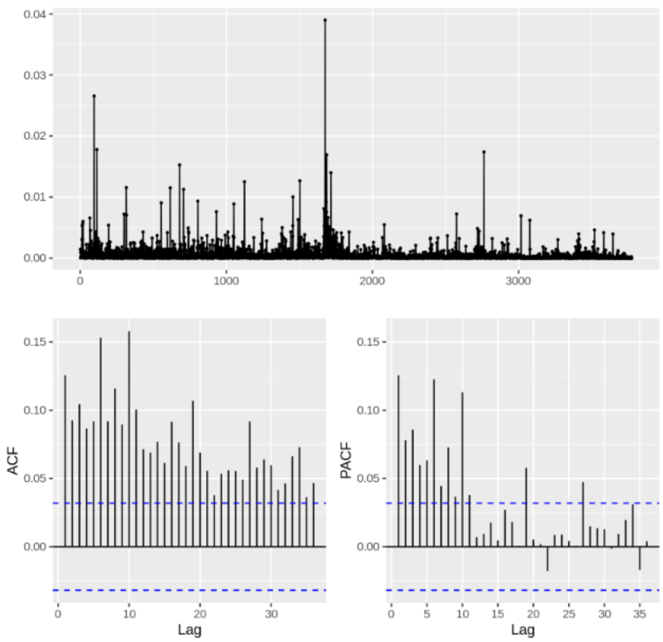


Figure 5: Squared return data

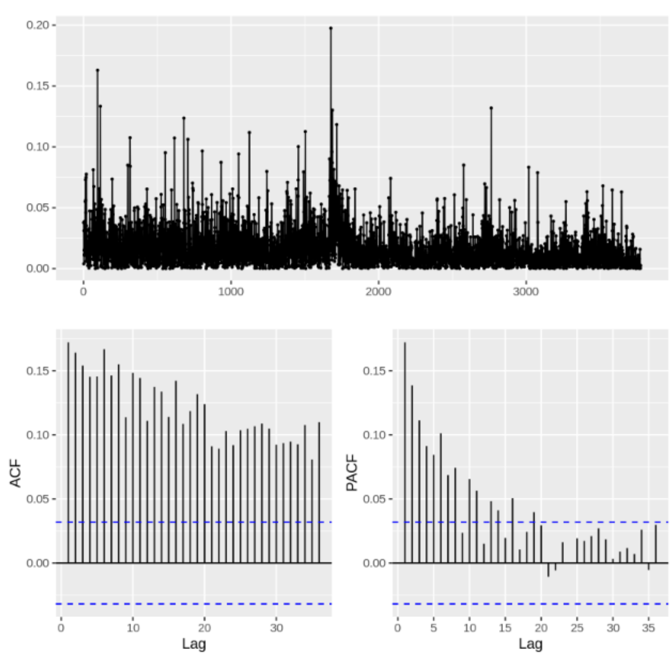


Figure 6: Absolute Log-return data

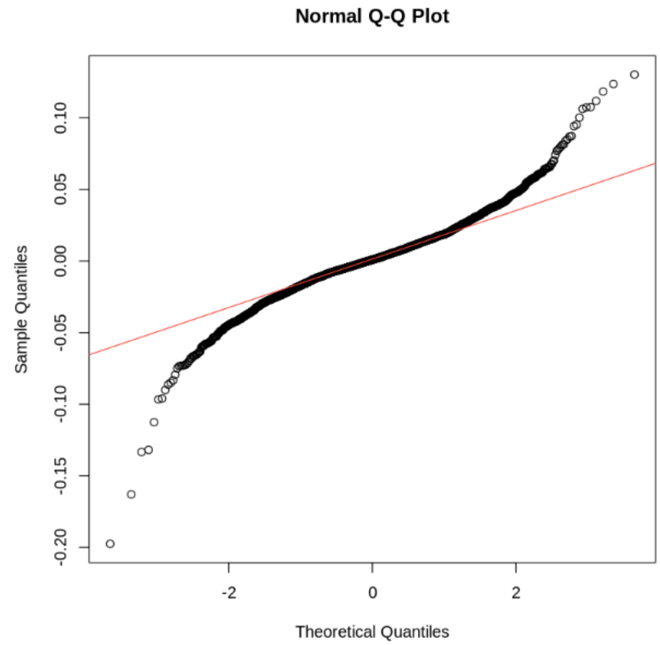


Figure 7: QQ-Plot of Log-return Data.

In conclusion, the log-return data exhibit no serial correlation and are characterized by a heavy-tailed and left-skewed distribution. Given these patterns, the ARCH-GARCH model framework is well-suited for modeling and analyzing such time series data

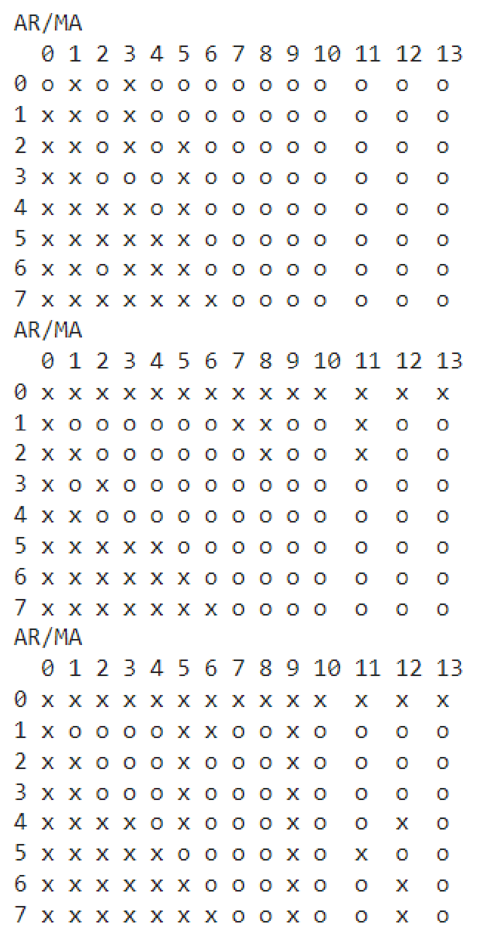
2.2 Data Splitting

Before initiating the model selection process, the data is partitioned into training and validation sets. Given the presence of volatility clustering within the time series, forecasting beyond a month is deemed unnecessary, leading to the establishment of 30 days for the validation set, rather than adopting a specific ratio for division. Following this partition, the training set comprises 3,745 data points, and the validation set contains 30 data points, which will solely be used for evaluating the predictive accuracy of the model, sum up to 3775.

3 Model Fitting

The Extended Autocorrelation Function (EACF) of daily returns indicates a parameter setting of (4,0), while the EACF for absolute returns points towards parameters of (1,1), (2,2), or (3,3). Furthermore, the EACF for squared returns recommends a parameter of (1,1). After synthesizing these insights, we have selected (1,1) as the most suitable parameter configuration for the data, which corresponds to the GARCH(1,1) model.

Next, we proceed with a diagnostic evaluation of the GARCH(1,1) model to assess its characteristics. The analysis of standardized residuals is displayed in Figure 9, while the corresponding QQ-plot is illustrated in Figure 10. Additionally, Figure 11 presents the outcomes of the Generalized Portmanteau Test for the squared residuals of the GARCH(1,1) model, and Figure 12 depicts the ACF/PACF plots of these squared residuals.



```
AR/MA
  0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 o x o x o o o o o o o  o  o  o
1 x x o x o o o o o o o  o  o  o
2 x x o x o x o o o o o  o  o  o
3 x x o o o x o o o o o  o  o  o
4 x x x x o x o o o o o  o  o  o
5 x x x x x x o o o o o  o  o  o
6 x x o x x x o o o o o  o  o  o
7 x x x x x x x o o o o  o  o  o
AR/MA
  0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 x x x x x x x x x x x  x  x  x
1 x o o o o o o x x o o  x  o  o
2 x x o o o o o o x o o  x  o  o
3 x o x o o o o o o o o  o  o  o
4 x x o o o o o o o o o  o  o  o
5 x x x x x o o o o o o  o  o  o
6 x x x x x x o o o o o  o  o  o
7 x x x x x x x o o o o  o  o  o
AR/MA
  0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 x x x x x x x x x x x  x  x  x
1 x o o o o x x o o x o  o  o  o
2 x x o o o x o o o x o  o  o  o
3 x x o o o x o o o x o  o  o  o
4 x x x x o x o o o x o  o  x  o
5 x x x x x o o o o x o  x  o  o
6 x x x x x x o o o x o  o  x  o
7 x x x x x x x o o x o  o  x  o
```

Figure 8: EACF[1]

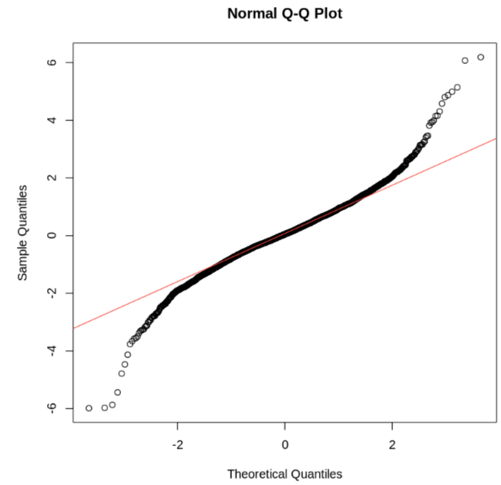


Figure 10: QQ-Plot of GARCH(1,1)

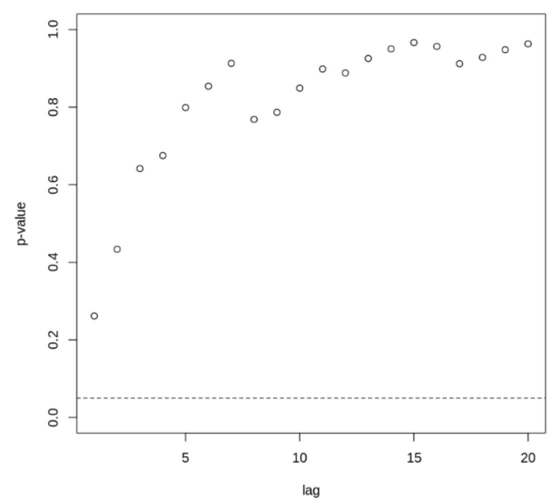


Figure 11: Generalized Portmanteau Test

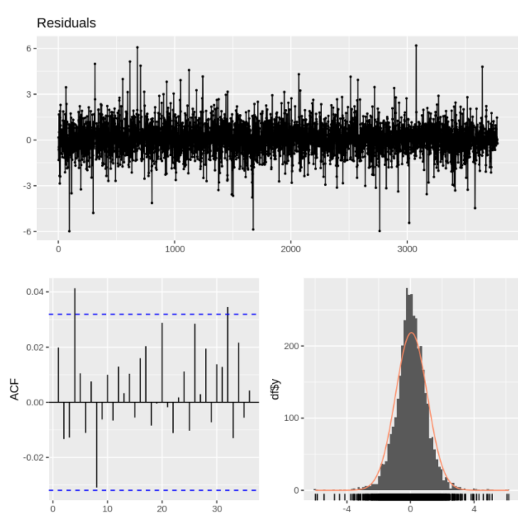


Figure 9: Diagnostic Check of Residuals of GARCH(1,1).

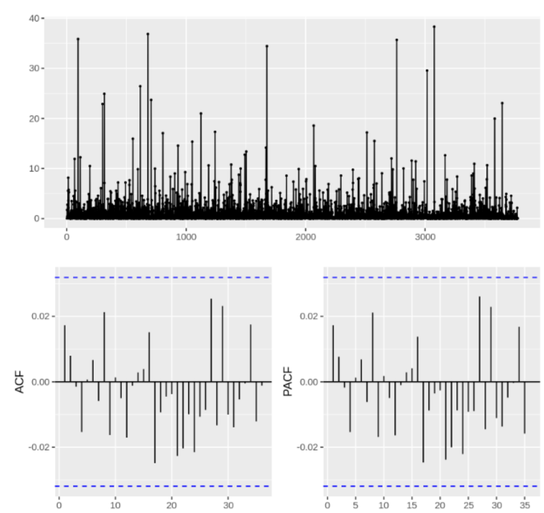


Figure 12: ACF/PACF plots of the squared residuals.

The p-values depicted in Figure 17 exceed 0.05, indicating that the squared residuals do not exhibit correlation over time, thereby suggesting the potential independence of the standardized residuals. Subsequently, we employ the rugarch package in R to determine the most suitable GARCH(1,1) model, considering various data distributions and specific sub-types of GARCH models.

To identify the optimal distribution setting, we conduct a series of sGARCH(1,1) models, each configured with different data distribution assumptions, to evaluate their likelihood (with higher values indicating better fit) and information criteria (specifically, the Akaike Information Criterion, where lower values are preferable). The outcomes of these analyses are presented in Table 1. The data in the table 1 reveals that the (Skew-) T-Distribution outperforms the other distributions. Consequently, we have chosen the T-distribution as the optimal distribution setting.

Distribution	Likelihood	Akaike
Normal	9296.009	-4.9229
Skew Normal	9296.242	-4.9225
Generalized Error	9449.263	-5.0036
T-Distribution	**9472.314**	**-5.0158**
Skew Generalized Error	9450.581	-5.0038
Normal Inverse Gaussian	9468.302	-5.0131
Skew T-Distribution	**9472.777**	**-5.0155**
Johnson's SU	9471.499	-5.0148
Generalized Hyperbolic	9472.504	-5.0148

Table 1: Likelihood and Akaike Value of Different Distribution.

To identify the most suitable sub-model within the fGARCH framework (or other GARCH models), we conduct a series of tests, akin to how we selected the distribution. These tests help us determine the best sub-model, with the findings presented in Table 2.

No	Sub-Model / Model	Likelihood	Akaike
1	fGARCH	9472.314	-5.0158
2	fGARCH - TGARCH	9494.400	-5.0270
3	fGARCH - AVGARCH	9494.357	-5.0264
4	fGARCH - NGARCH	9482.028	-5.0204
5	fGARCH - NAGARCH	9487.081	-5.0231
6	fGARCH - APARCH	9494.421	-5.0264
7	fGARCH - GJRGARCH	9482.289	-5.0206
8	fGARCH - ALLGARCH	9495.035	-5.0262
9	**eGARCH**	**9496.097**	**-5.0279**
10	gjrGARCH	9482.289	-5.0206
11	apARCH	9494.421	-5.0264
12	iGARCH	9471.993	-5.0162
13	csGARCH	9486.041	-5.0220

Table 2: Likelihood and Akaike Value of Different Distribution.

The outcomes presented in the table 2 above demonstrate that the eGARCH(1,1) model is the most effective for fitting the time series data returns. As a result, we have selected eGARCH(1,1) as our definitive model for forecasting Apple's stock prices. Furthermore, all the models mentioned have successfully passed both diagnostic and Ljung-Box tests.

4 Forecasting

Time series forecasting involves the collection of historical data, preparing it for algorithms to consume, and then predicting the future values based on patterns learned from the historical data[3].

I will only consider the data in the training set and not the full dataset. After re-implementing the model fitting procedure, we were able to determine the training data's likelihood and Akaike score, which came back with scores of 9385.9 and -5.0093, respectively.

As shown in Figure 16, we can simply construct forecasting graphs using the N-Roll arrangement. The yellow-shaded region in these graphs denotes the 95% upper and lower boundaries. The 30-day forecast series with unconditional 1-Sigma and forecast unconditional sigma plots are shown in the left parts.

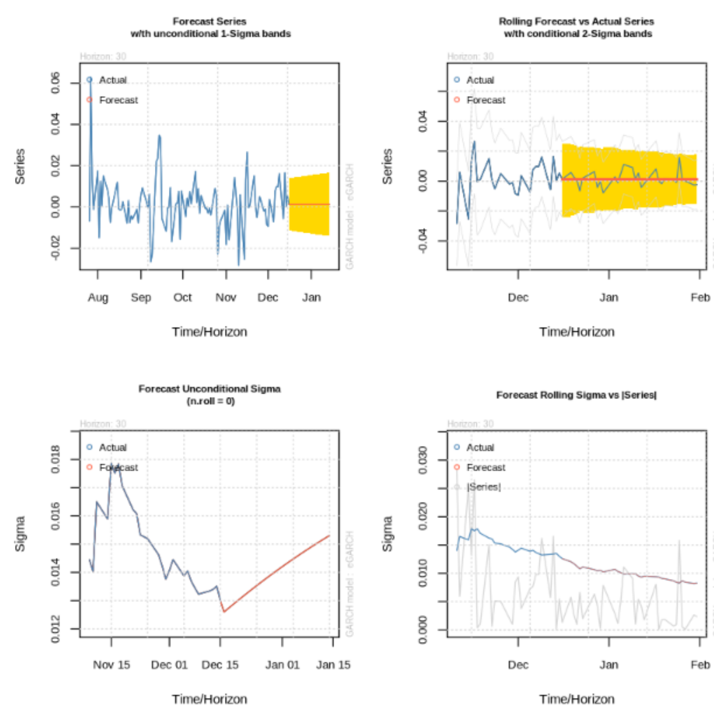


Figure 13: shows Forecast graphs

The upper left section of Figure 16 illustrates the actual versus predicted values in a time series, with a shaded region signifying the 1-sigma confidence bounds, suggesting a 68% probability of future data points falling within this range. The upper right section also compares actual data with predictions, but with 2-sigma confidence intervals, implying a 95% confidence level and providing a broader prediction range as the model incorporates new information.

The lower left graph displays the projected long-term average volatility, independent of past sigma values, with a red line mapping out the expected volatility over time. Conversely, the lower right graph presents a dynamic view of volatility forecasting, showing actual volatility against predictions that adjust with incoming data, illustrating the model's responsiveness to recent volatility trends. These visualizations collectively evaluate a volatility model's capability to forecast financial time series data, such as stock returns, highlighting its precision and reliability in prediction.

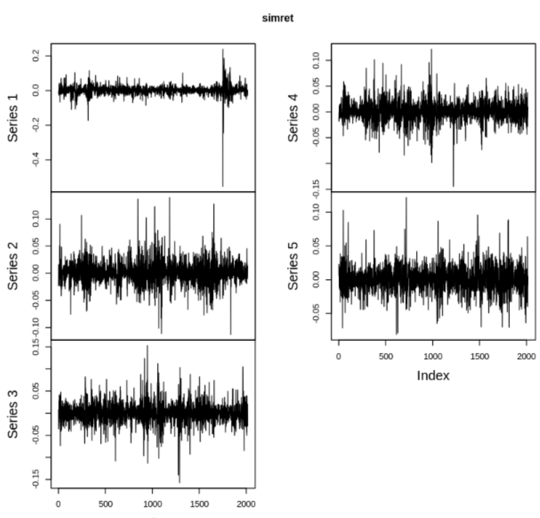


Figure 14: Depicts the projected returns as forecasted by the GARCH model's process, showcasing five distinct simulated trajectories of financial returns. These trajectories oscillate around a central value, exhibiting differing intensities of volatility. Such simulations are indicative of possible future patterns in returns, valuable for evaluating risks and predicting financial trends. The observed trends imply the presence of volatility clustering, a typical feature in financial datasets.

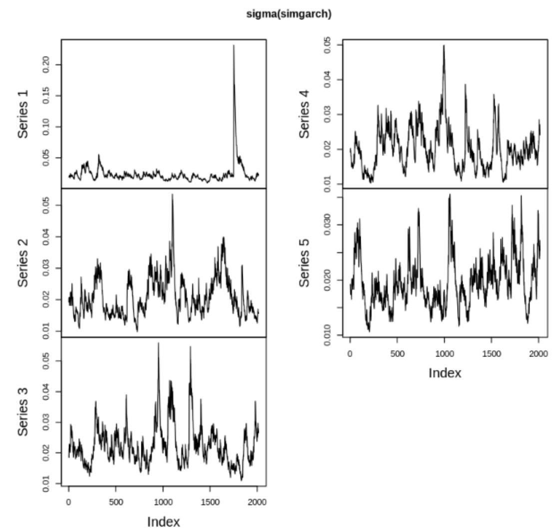


Figure 15: Symbolizes the predicted variability, or volatility, of the projected returns. Displays a set of five distinct simulated sequences that trace the progression of volatility through time as foreseen by a GARCH model. Each sequence corresponds to the conditional standard deviation—indicating the level of volatility—of financial returns across various simulated settings.

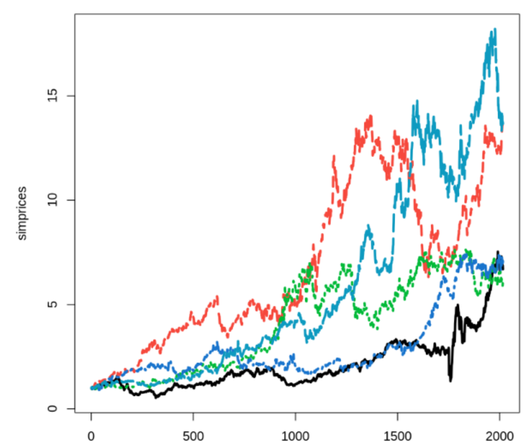


Figure 16: Simulated Stock Price Trajectories: A graphical representation of multiple potential future paths for stock prices generated by a GARCH model, highlighting the divergence and variability in outcomes over time.

PART 2.

1. Is X_t stationary?
Define X_t as

$$X_t = \begin{cases} Y_t & \text{if } t \text{ is even,} \\ Y_t + 1 & \text{if } t \text{ is odd} \end{cases}$$

where Y_t is a stationary time series.

To ascertain if Xt is stationary, we must observe its behavior across time. When t is even, X_t is equal to Y_t, which is a stationary series. This implies that for even t:

The mean of X_t stays constant: $E[X_t] = E[Y_t]$. The variance of X_t remains constant: $\text{Var}(X_t) = \text{Var}(Y_t)$. The autocorrelation pattern of X_t mirrors that of Y_t

Similarly, when t is odd, X_t is equivalent to Y_{t+1}. Since Y_t is stationary, for odd t: The mean of X_t remains consistent: $E[X_t] = E[Y_{t+1}] = E[Y_t]$. The variance of X_t remains unchanged: $\text{Var}(X_t) = \text{Var}(Y_{t+1}) = \text{Var}(Y_t)$. The autocorrelation structure of X_t mirrors that of Y_{t+1}, which is also stationary

Because the statistical properties of X_t (mean, variance, autocorrelation) remain constant over time, irrespective of t being even or odd, it is concluded that X_t is a stationary time series. The alternating arrangement between Y_t and Y_{t+1} does not affect the stationarity of X_t, as long as Y_t remains stationary.

2. Suggestion for a transformation so that X_t becomes stationary:
Define X_t as

$$X_t = (1 + 2t)S_t + Z_t,$$

where $S_t = S_{t-12}$. Here, X_t exhibits both a trend (due to $1+2t$) and seasonality (due to S_t, which repeats every 12 periods). To make X_t stationary, both the trend and the seasonality need to be addressed:

- **Detrending:** Subtract the trend component, $1+2t$, from X_t. This can be achieved by fitting a linear model to X_t and then removing the fitted line.

- **Deseasonalizing:** Since S_t is periodic with a 12-month cycle, you can either subtract the seasonal average for each period from X_t or use differencing by subtracting X_{t-12} from X_t.

After detrending and deseasonalizing, we may check if the transformed series is stationary using statistical tests such as the Augmented Dickey-Fuller (ADF) test.

3. Analysis of the International Airline Passengers time series:

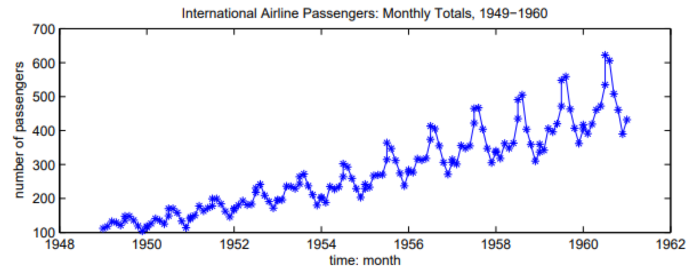


Figure 17: Time plot of International Airline Passengers

(a) **Is it stationary?**
The series is **not stationary**. The visual inspection often shows the non-constant mean and variance over time due to the apparent trend and possible seasonality. This would need to be confirmed with a formal statistical test like the Augmented Dickey-Fuller (ADF) test.

(b) **What kind of time series components do the data contain?**
The series likely contains **Trend, Seasonal, and Irregularity** components. The trend component suggests growth over time, and the seasonal component reflects fluctuations within a year and presence of irregularity which can not explained by both trend and seasonality.

(c) **Suggestion for a transformation to equalize the seasonal variation:**
A common transformation to stabilize the variance and equalize seasonal variation is the logarithmic transformation, especially effective when dealing with exponential growth or multiplicative seasonality. After transforming, seasonal differencing might be used to further stabilize the series:

$$\log(X_t) - \log(X_{t-12})$$

References

[1] Kung-Sik Chan. eacf compute the sample extended acf (esacf), 2020. Accessed: 2024-02-28.

[2] David Lane. Box-cox transformations, 2022. Accessed: 2024-02-26.

[3] Samuel L. Smith Soham De. Time series forecasting tutorial, 2022. Accessed: 2024-02-29.

[4] Yahoo. Apple inc. (aapl). In *NasdaqGS - Nasdaq Real Time Price USD Apple Inc. (AAPL)*. Yahoo. Accessed: 2024-04-26.

A Appendix

Below are the lines of codes used to implement part 1 of the project.

```
# Install Necessary Packages
install.packages("TSA")
install.packages("astsa")
install.packages("zoo")
install.packages("xts")
install.packages("quantmod")
install.packages("fBasics")
install.packages("forecast")
install.packages("ggplot2")
install.packages("fGarch")
install.packages("rugarch")
install.packages("tseries")

# Load Libraries
library(TSA)
library(astsa)
library(zoo)
library(xts)
library(quantmod)
library(fBasics)
library(forecast)
library(ggplot2)
library(fGarch)
library(rugarch)
library(tseries)
library(zoo)

library(googledrive)
# Authenticate access to Google Drive
drive_auth(use_oob = TRUE)

file <- drive_get("AAPL.csv")
# Download the file to the Colab environment
drive_download(file, overwrite = TRUE)
# Read the downloaded CSV file into R
data <- read.csv("AAPL.csv")

# Set Global Variables
global.xlab <- 'Date'
global.ylab <- 'Adjusted Closing Price (USD)'
global.stockname <- 'AAPL'

# Load Data from Yahoo Finance
data <- getSymbols(global.stockname, from='2002-02-01', to='2017-02-01', src='yahoo', auto.assign = F)
data <- na.omit(data)
data_AC <- data[,4]

# Plotting Adjusted Closing Price
plot(data_AC,
     main="Adjusted Closing Price of AAPL (2002-2017)",
     xlab=global.xlab,
     ylab=global.ylab,
     col="blue")

# Basic Data Attributes
print(min(data_AC))
print(max(data_AC))
print(mean(data_AC))
print(var(data_AC))
# ACF/PACF Plots
ggtsdisplay(data_AC)

# Augmented Dickey-Fuller Test
adf.test(data_AC)
# Seasonal Decomposition
ts_AC <- ts(Ad(to.monthly(data)), frequency = 12)
fit.stl <- stl(ts_AC[,1], s.window = "period")
autoplot(fit.stl, main="Seasonal and Trend decomposition using Loess (STL)")

# DATA TRANSFORMATION
lambda = 0 # natural logarithm
percentage = 1
data_R <- diff(BoxCox(data_AC, lambda))*percentage # returns
data_R <- data_R[!is.na(data_R)]

# ACF and PACF
ggtsdisplay(data_R)
ggtsdisplay(abs(data_R))
ggtsdisplay(data_R^2)
adf.test(data_R)

# QQ Plot
qqnorm(data_R)
qqline(data_R, col = 2)
skewness(data_R)
kurtosis(data_R)

  # EACF
eacf(data_R)
eacf(abs(data_R))
eacf(data_R^2)

 # Splitting data
train_num <- (length(data_R) - 30)
data_train <- head(data_R, train_num) # 3745
data_test <- tail(data_R, round(length(data_R) - train_num)) # 30

  # GARCH model Fitting
Garch_11=garch(data_R, order=c(1,1))
summary(Garch_11)
AIC(Garch_11)

  # Distribution
# sGARCH(1,1), Normal Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0))),
  data = data_R)

# sGARCH(1,1), Skew Normal Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "snorm"),
  data = data_R)
# sGARCH(1,1), T-Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "std"),
  data = data_R)

 # sGARCH(1,1), Skew T-Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "sstd"),
  data = data_R)

   #sGARCH(1,1), Generalized Error Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
```

```
  distribution.model = "ged"),
  data = data_R)

  #sGARCH(1,1), Skew Generalized Error Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "sged"),
  data = data_R)

# sGARCH(1,1), Normal Inverse Gaussian Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "nig"),
  data = data_R)

  # sGARCH(1,1), Generalized Hyperbolic Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "ghyp"),
  data = data_R)

 # sGARCH(1,1), Johnson's S_U Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "jsu"),
  data = data_R)

  # Sub-MODEL
# fGARCH(1,1), GARCH, T-Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(model = "fGARCH",
                      submodel = "GARCH",
                      garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "std"),
  data = data_R)

 #fGARCH(1,1), TGARCH, T-Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(model = "fGARCH",
                      submodel = "TGARCH",
                      garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "std"),
  data = data_R)

 # fGARCH(1,1), AVGARCH, T-Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(model = "fGARCH",
                      submodel = "AVGARCH",
                      garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "std"),
  data = data_R)

  #fGARCH(1,1), NGARCH, T-Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(model = "fGARCH",
                      submodel = "NGARCH",
                      garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "std"),
  data = data_R)

  #fGARCH(1,1), NAGARCH, T-Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(model = "fGARCH",
                      submodel = "NAGARCH",
                      garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "std"),
  data = data_R)

  #fGARCH(1,1), APARCH, T-Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(model = "fGARCH",
                      submodel = "APARCH",
                      garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "std"),
  data = data_R)

 #fGARCH(1,1), GJRGARCH, T-Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(model = "fGARCH",
                      submodel = "GJRGARCH",
                      garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "std"),
  data = data_R)

 # fGARCH(1,1), ALLGARCH, T-Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(model = "fGARCH",
                      submodel = "ALLGARCH",
                      garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "std"),
  data = data_R)

  # eGARCH(1,1), T-Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(model = "eGARCH",
                      garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "std"),
  data = data_R)

  # gjrGARCH(1,1), T-Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(model = "gjrGARCH",
                      garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "std"),
  data = data_R)

 # apARCH(1,1), T-Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(model = "apARCH",
                      garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "std"),
  data = data_R)

 # iGARCH(1,1), T-Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(model = "iGARCH",
                      garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "std"),
  data = data_R)

#csGARCH(1,1), T-Distribution
ugarchfit(spec = ugarchspec(
  variance.model=list(model = "csGARCH",
                      garchOrder=c(1,1)),
  mean.model=list(armaOrder = c(0,0)),
  distribution.model = "std"),
  data = data_R)
```

```
# FORECAST
# N-roll
garchspec <- ugarchspec(mean.model=list(armaOrder=c(0,0)),
                        variance.model=list(model = "eGARCH", garchOrder=c(1,1)),
                        distribution.model = "std")
fta <- ugarchfit(garchspec, data_R, out.sample=length(data_test))
fwdCast = ugarchforecast(fta, n.ahead=length(data_test), n.roll=length(data_test))
plot(fwdCast, which="all")

 # SIMULATIONS
garchspec <- ugarchspec(mean.model=list(armaOrder=c(0,0)),
                        variance.model=list(model = "eGARCH", garchOrder=c(1,1),
                        variance.targeting = FALSE),
                        distribution.model = "std")
garchfit <- ugarchfit(data = data_train, spec = garchspec)
simgarchspec <- garchspec
setfixed(simgarchspec) <- as.list(coef(garchfit))
simgarch <- ugarchpath(spec = simgarchspec, m.sim = 5,
                       n.sim = 8 * 252, rseed = 123)
simret <- fitted(simgarch)
plot.zoo(simret)
plot.zoo(sigma(simgarch))
simprices <- exp(apply(simret, 2, "cumsum"))
matplot(simprices, type = "l", lwd = 3)
```

Source Code 1: Full codes implementation