Analysis of Financial Time Series

Contents

1 Introduction

We saw in earlier sections that one type of financial data is financial time series. This data sounded interesting to us and so we looked at the sequential stock prices of Apple (an American multinational corporation and technology company) over a period of time. This forms a financial time series. Financial time series exhibit distinct stylistic features such as volatility, which necessitate different methods to analysis of non-financial time series. Understanding the features of time series and techniques for their analysis is crucial to areas such as quant finance, and can help make informed investment decisions and assess market dynamics.

2 Requirements

Some packages are used to facilitate tasks. These can be installed if needed by running the code chunk below.

```
# List of required packages
required_packages <- c("quantmod", "ggplot2", "gridExtra", "zoo")

# Function to check if packages are installed
new_packages <- required_packages[!(required_packages %in%
↪  installed.packages()[,"Package"])]
```

```r
# Install missing packages
if(length(new_packages)) install.packages(new_packages)

# Load required packages
lapply(required_packages, library, character.only = TRUE)
```

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: TTR

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## [[1]]
##  [1] "quantmod"  "TTR"       "xts"       "zoo"       "stats"     "graphics"
##  [7] "grDevices" "utils"     "datasets"  "methods"   "base"
##
## [[2]]
##  [1] "ggplot2"   "quantmod"  "TTR"       "xts"       "zoo"       "stats"
##  [7] "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
##
## [[3]]
##  [1] "gridExtra" "ggplot2"   "quantmod"  "TTR"       "xts"       "zoo"
##  [7] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [13] "base"
##
## [[4]]
##  [1] "gridExtra" "ggplot2"   "quantmod"  "TTR"       "xts"       "zoo"
##  [7] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [13] "base"
```

3 Loading data

We fetch the stock price data of Apple from the Yahoo Finance API (application interface). We chose this data as it can easily be accessed using the Yahoo Finance API and is an example of a financial time series. We do not go into the technicalities of APIs, as the following very simple `R` code is sufficient to import the data. The interested reader may consult [5] for a detailed guide.

```r
# Get historical stock data for Apple (AAPL) in xts format
library(quantmod) # Get data from the Yahoo Finance API
getSymbols("AAPL", src = "yahoo", from = "2020-01-01", to = "2024-01-01")
```

```
## [1] "AAPL"
```

```r
str(AAPL)
```

```
## An xts object on 2020-01-02 / 2023-12-29 containing:
##   Data:    double [1006, 6]
##   Columns: AAPL.Open, AAPL.High, AAPL.Low, AAPL.Close, AAPL.Volume ... with 1 more column
##   Index:   Date [1006] (TZ: "UTC")
##   xts Attributes:
##     $ src    : chr "yahoo"
##     $ updated: POSIXct[1:1], format: "2024-09-30 14:32:31"
```

We see that the data comes in the form of an `xts` object, meaning "eXtensible Time Series". This is also an `R` package designed for handling and analyzing time series data. The interested reader may run `vignette('xts')` for more information. An `xts` object is a type of data structure that extends the so-called `zoo` class, allowing users to work with time series data more effectively. Due to time constraints, we do not explore the prerequisite `zoo` class or the `xts` object further. We direct the reader to a tutorial such as [1] for more information.

We turn to exploratory data analysis in order to understand the data.

4 Exploratory Data Analysis

We begin with EDA. WE first look at the variables present in the data. The variables present in the data are shown in the table below.

Variable Name	Description
Date	The date of the stock price observation.
Open	The stock price at market open.
High	The highest stock price during the trading day.
Low	The lowest stock price during the trading day.
Close	The stock price at market close.
Volume	The total number of shares traded during the day.
Adjusted	The adjusted closing price, accounting for splits and dividends.

We now show some quantitative aspects of the dataset.

```r
head(AAPL)
```

```
##            AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
## 2020-01-02   74.0600   75.1500  73.7975    75.0875   135480400       75.0875
## 2020-01-03   74.2875   75.1450  74.1250    74.3575   146322800       74.3575
## 2020-01-06   73.4475   74.9900  73.1875    74.9500   118387200       74.9500
## 2020-01-07   74.9600   75.2250  74.3700    74.5975   108872000       74.5975
## 2020-01-08   74.2900   76.1100  74.2900    75.7975   132079200       75.7975
## 2020-01-09   76.8100   77.6075  76.5500    77.4075   170108400       77.4075
```

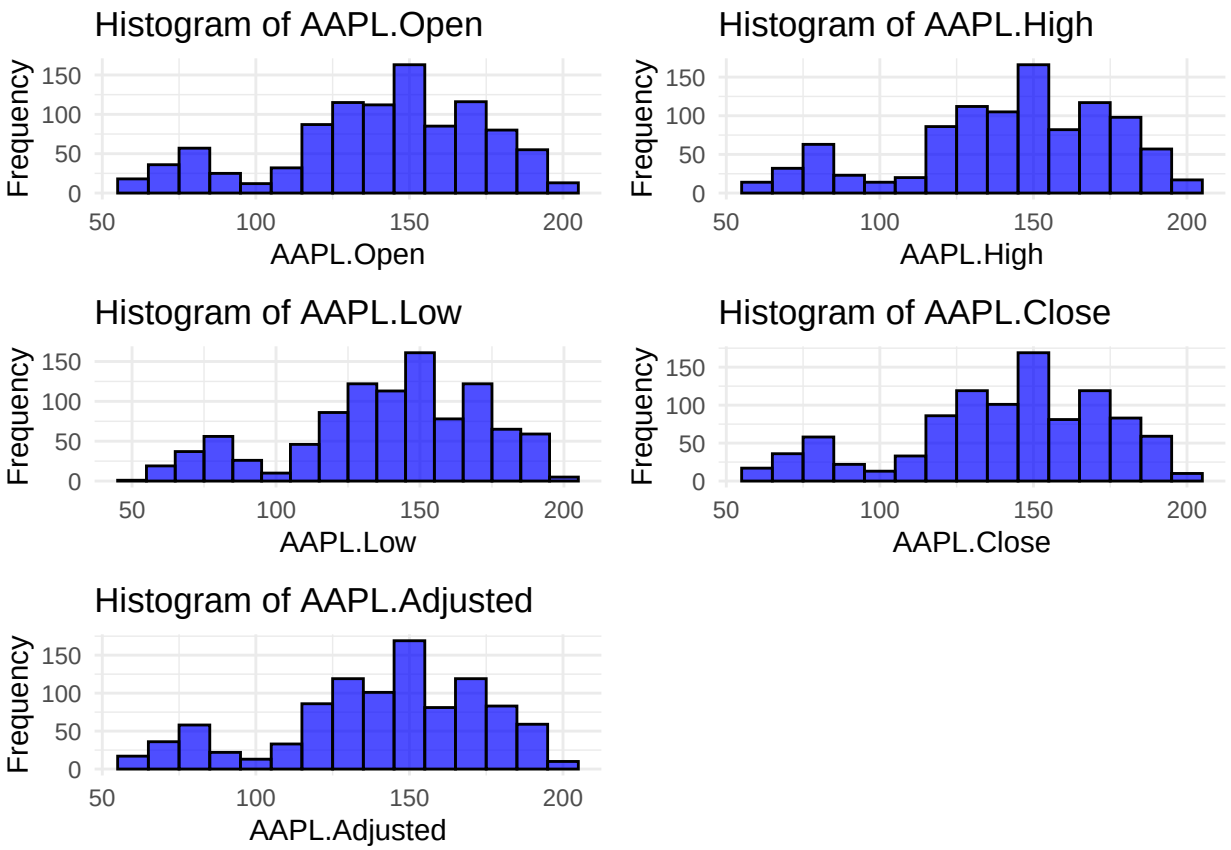
```r
summary(AAPL)
```

```
##      Index              AAPL.Open        AAPL.High         AAPL.Low
##  Min.   :2020-01-02   Min.   : 57.02   Min.   : 57.12   Min.   : 53.15
##  1st Qu.:2020-12-30   1st Qu.:123.68   1st Qu.:125.03   1st Qu.:122.16
##  Median :2021-12-29   Median :145.54   Median :147.26   Median :144.12
##  Mean   :2021-12-30   Mean   :140.68   Mean   :142.32   Mean   :139.14
##  3rd Qu.:2022-12-28   3rd Qu.:166.30   3rd Qu.:168.15   3rd Qu.:164.81
##  Max.   :2023-12-29   Max.   :198.02   Max.   :199.62   Max.   :197.00
##    AAPL.Close      AAPL.Volume        AAPL.Adjusted
##  Min.   : 56.09   Min.   : 24048300   Min.   : 56.09
##  1st Qu.:123.59   1st Qu.: 64076750   1st Qu.:123.59
##  Median :145.86   Median : 84675400   Median :145.86
##  Mean   :140.81   Mean   : 98952111   Mean   :140.81
##  3rd Qu.:166.22   3rd Qu.:115506875   3rd Qu.:166.22
##  Max.   :198.11   Max.   :426510000   Max.   :198.11
```

We can see that the data was collected from times January 1 2020 to December 29 2023. The volume seems to be quite positively skewed with a mean of 98.95 million, a maximum of 426.51 million and a minimum of 24.0483 million. The opening, high, low and closing and adjusted closing prices seem to be similar in their means and quantiles. To explore this further, we looked at the shape of the histograms of the prices.

```r
# Load necessary libraries
library(ggplot2)    # To make plots
library(gridExtra) # To arrange multiple plots in a grid

# Function to plot histograms
plot_histogram <- function(data, column_name) {
  suppressWarnings(
    ggplot(data, aes_string(x = column_name)) +
      geom_histogram(binwidth = 10, fill = "blue", color = "black", alpha = 0.7) +
      labs(title = paste("Histogram of", column_name), x = column_name, y = "Frequency")
      ↪  +
      theme_minimal()
  )
}

# We suppress warnings here. The warning gotten was:
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

# List of columns to plot
columns_to_plot <- c("AAPL.Open", "AAPL.High", "AAPL.Low", "AAPL.Close", "AAPL.Adjusted")

# Initialize a list to store the plots
plot_list <- list()

# Generate histograms for each column and store them in the list
```

```r
for (col in columns_to_plot) {
  plot_list[[col]] <- plot_histogram(AAPL, col)
}

# Arrange and display the plots side by side
grid.arrange(grobs = plot_list, ncol = 2)  # Adjust ncol to the number of columns you
↪  want side by side
```



As expected from our earlier analysis of quantiles and means, we see that the shapes of the histograms are similar. We however uncover that the prices are bimodal. This was not seen in the earlier analysis.

We proceed with our exploration and use line plots to visualise the closing price and trading volume over time. We also use a 30-day moving average for the closing price to understand the trend. A moving average is used to smooth out the price data by creating a constantly updated average price over the last 30 days. This helps in identifying the underlying trend by filtering out short-term noise and fluctuations.

The moving average MA_t for day t is calculated by averaging the closing prices of the last 30 days. Mathematically, the 30-day moving average is given by:

$$\text{MA}_t = \frac{1}{30} \sum_{i=0}^{29} P_{t-i}$$

Where:

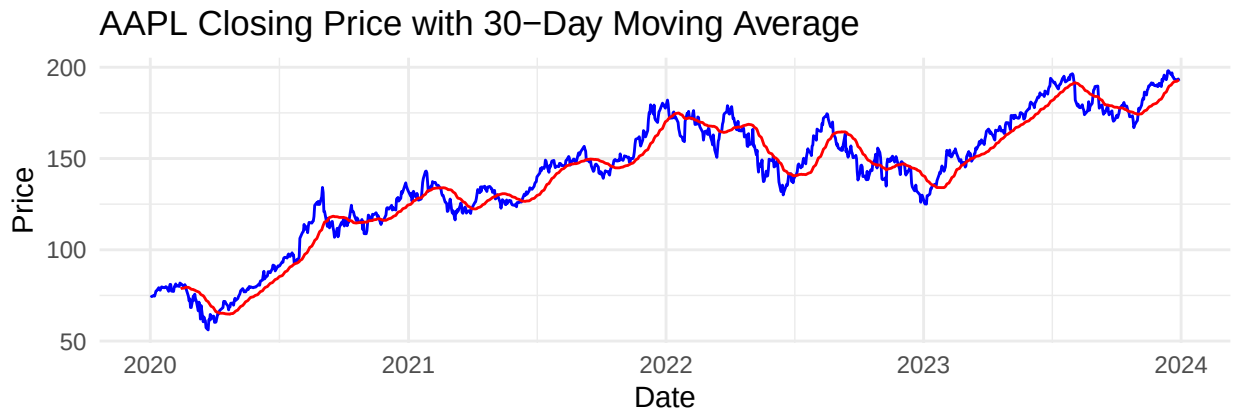
- MA_t is the moving average at day t.

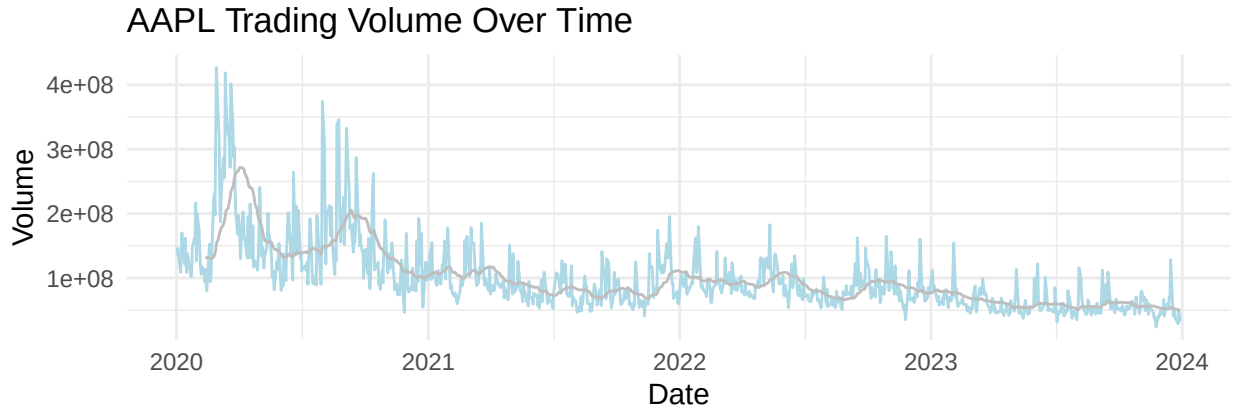
- P_{t-i} is the closing price on day $t - i$.

The moving average is recalculated each day, hence the name **moving** average. As new price data becomes available, the oldest data point is dropped, and the new price is added to the calculation, keeping the window fixed at 30 days.

```
library(zoo)          # For rollmean
library(gridExtra)    # For arranging plots

# Calculate 30-Day Moving Average for Closing Price
AAPL$MA30Price <- rollmean(AAPL$AAPL.Close, k = 30, fill = NA, align = "right")

# 1. Plot: Closing Price Over Time
p1 <- ggplot() +
  geom_line(aes(x = index(AAPL), y = coredata(AAPL$AAPL.Close)), color = "blue") +
  geom_line(aes(x = index(AAPL), y = coredata(AAPL$MA30Price)), color = "red") +
  labs(title = "AAPL Closing Price with 30-Day Moving Average",
       x = "Date",
       y = "Price") +
  theme_minimal()

# Calculate 30-Day Moving Average for Volume
AAPL$MA30Vol <- rollmean(AAPL$AAPL.Volume, k = 30, fill = NA, align = "right")

# 2. Plot: Volume Over Time
p2 <- ggplot() +
  geom_line(aes(x = index(AAPL), y = coredata(AAPL$AAPL.Volume)), color = "lightblue") +
  geom_line(aes(x = index(AAPL), y = coredata(AAPL$MA30Vol)), color = "grey") +
  labs(title = "AAPL Trading Volume Over Time",
       x = "Date",
       y = "Volume") +
  theme_minimal()

# Arrange plots in a grid
gridExtra::grid.arrange(p1, p2, ncol = 1)
```

```
## Warning: Removed 29 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 29 rows containing missing values or values outside the scale range
## (`geom_line()`).
```





We see warnings as follows:

```
Warning: [38;5;232mRemoved 29 rows containing missing values or values outside the scale range (geom_li
## Warning: Removed 29 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 29 rows containing missing values or values outside the scale range
## (`geom_line()`).
(from running the above code)
```

This is because the 30-day moving average will not have values for the first 29 observations, because there are not enough observations at that point to form the 30-day moving average. It is not a cause for concern, but the code was not modified to suppress this warning due to time constraints.

We can see that the closing price is quite volatile but with an overall increasing trend. The trading volume seems to be quite volatile as well, having high peaks in the time period between 2020 and 2021. We investigate the reason for these peaks. We found that this period coincided with a global pandemic, with a disease called Coronavirus Disease 2019 (COVID-19) affecting the world. On August 19 2020, Apple's market value exceeded 2 trillion US dollars (see a source here). The trading volume seems to taper off and stabilise after 2021.

5 Further Exploration

We follow the literature (see Section 6.1 of Engle [1]) to explore models for financial time series. The quantity that is often modelled is the log-returns. Given the prices $\{P_t\}$, we can compute the log-return X_t at time

t using the formula

$$X_t = \log\left(\frac{P_t}{P_{t-1}}\right).$$

Note that if we have N values for the price, we will have $N-1$ values for the log-returns.

We compute and plot the log-returns below. We observe that $\{X_t\}$ has a sample path resembling that of a sequence of uncorrelated random variables with possibly time-varying variance.

```
# Calculate Daily Log Returns directly on the xts object
AAPL$Log_Returns <- diff(log(AAPL$AAPL.Close))

# Plot: Daily Log Returns
p4 <- ggplot() +
  geom_line(aes(x = index(AAPL), y = coredata(AAPL$Log_Returns)), color = "green") +
  labs(title = "AAPL Daily Log Returns", x = "Date", y = "Log Return") +
  theme_minimal()

# Log Returns (X_t)
AAPL$Log_Returns <- diff(log(AAPL$AAPL.Close))
```

The so-called "stylised features" (see [2] for example of the usage of the term) of log-returns include:

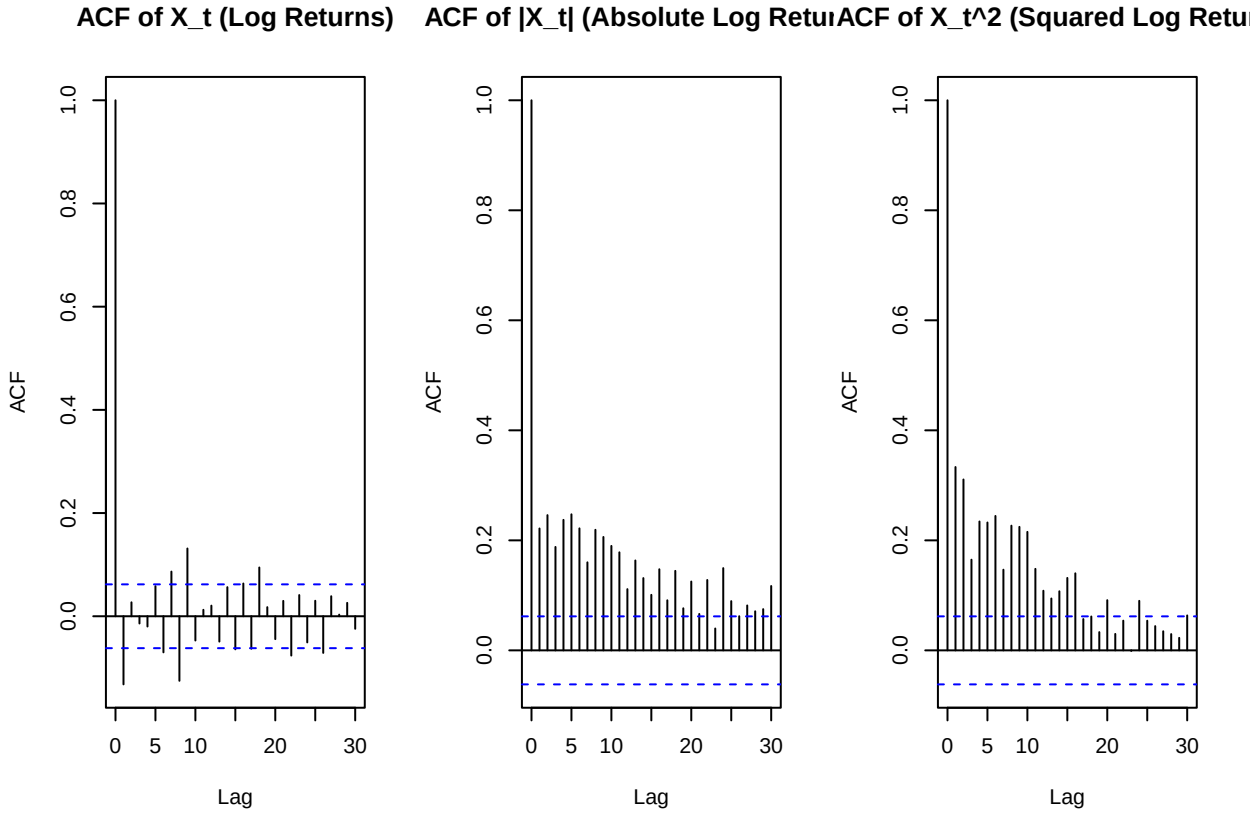
1. **Serial Dependence Without Correlation**: No significant autocorrelation is exhibited, but there may still be dependencies. To uncover this, we will consider the autocorrelation function (ACF) plots of $\{X_t\}$, $\{|X_t|\}$ and $\{X_t^2\}$.

Note: The autocorrelation ρ_k of a time series $\{X_t\}$ with n observations at lag k is given by:

$$\rho_k = \frac{\sum_{t=k+1}^{n}(X_t - \bar{X})(X_{t-k} - \bar{X})}{\sum_{t=1}^{n}(X_t - \bar{X})^2}.$$

This is investigated in the following plot. Notice that there is no persistent autocorrelation in $\{X_t\}$ while the autocorrelation are very persistent for the $\{|X_t|\}$ and $\{X_t^2\}$.

```
# --- 1. Serial Correlation ---

# Set up the plotting area to display 3 plots side by side
par(mfrow = c(1, 3))  # Set up the plot window to have 1 row and 3 columns

# 1. ACF of X_t (log returns)
acf(AAPL$Log_Returns, na.action = na.omit, main = "ACF of X_t (Log Returns)")

# 2. ACF of |X_t| (absolute log returns)
acf(abs(AAPL$Log_Returns), na.action = na.omit, main = "ACF of |X_t| (Absolute Log
↪  Returns)")

# 3. ACF of X_t^2 (squared log returns)
acf(AAPL$Log_Returns^2, na.action = na.omit, main = "ACF of X_t^2 (Squared Log Returns)")
```

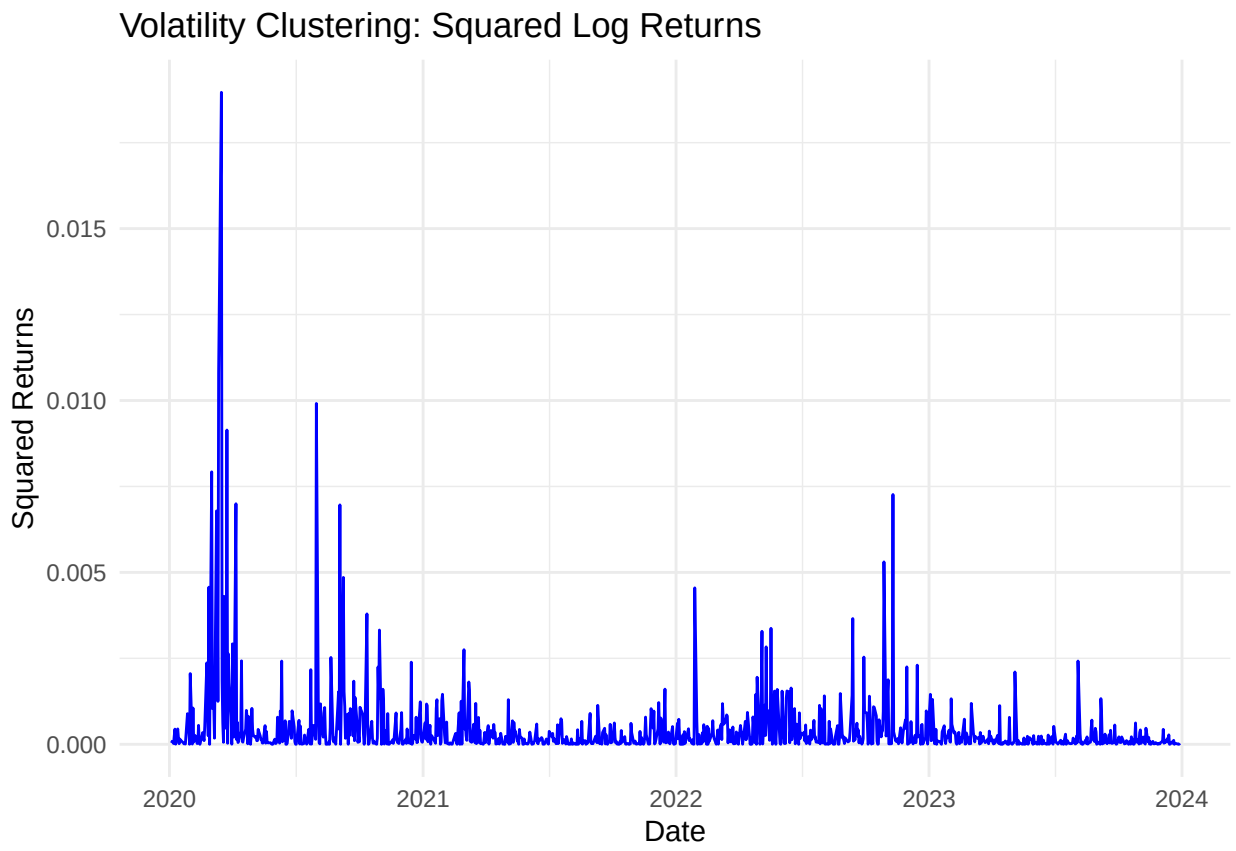


2. **Volatility Clustering**: High-volatility periods tend to be followed by high-volatility periods, and low-volatility periods tend to follow low-volatility periods. This is often visible in returns' time series. We will plot $\{X_t^2\}$ to see this.

This is investigated in the following plot. For the plot of $\{X_t^2\}$, we can see peaks indicating high volatility being somewhat clustered, showing the volatility clustering.

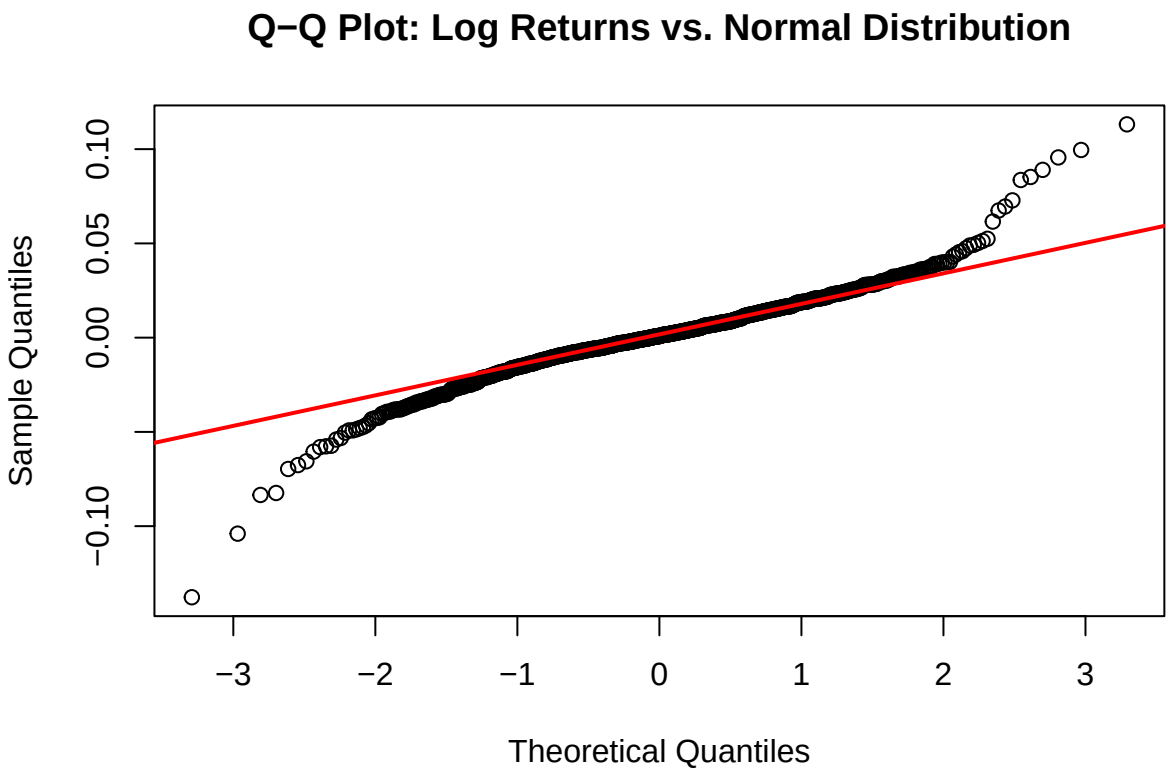
```
# --- 2. Volatility Clustering ---

# Plot X_t^2 (squared returns)
volatility_plot <- ggplot() +
  geom_line(aes(x = index(AAPL), y = coredata(AAPL$Log_Returns^2)), color = "blue") +
  labs(title = "Volatility Clustering: Squared Log Returns", x = "Date", y = "Squared
  ↪  Returns") +
  theme_minimal()

# Print volatility plot
print(volatility_plot)
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```



3. **Heavy-Tailed Distribution**: Financial returns do not follow a normal distribution. They tend to have heavier tails, meaning extreme events (both positive and negative) occur more frequently than would be expected in a normal distribution. We will uncover this with a quantile-quantile plot. The QQ plot shows that the log returns are heavy tailed.

```
# --- 3. Heavy-Tailedness ---

# Assuming AAPL data is already loaded and log returns are calculated
AAPL$Log_Returns <- diff(log(AAPL$AAPL.Close))

# Q-Q plot of log returns against a normal distribution
qqnorm(AAPL$Log_Returns, main = "Q-Q Plot: Log Returns vs. Normal Distribution")
qqline(AAPL$Log_Returns, col = "red", lwd = 2)  # Add a Q-Q line to compare to a normal
↪  distribution
```



4. **Leverage Effect**: The market reacts differently to positive and negative news, and this is reflected in the time series. We do not produce a plot to investigate this as we found variability in the literature in treating such effects qualitatively. We direct the reader to sources such as [6] and [7] for further details

6 Modelling

Modelling this data uses the so-called ARCH models, or **auto-regressive conditional heteroskedastic models**. This is an advanced topic that would significantly lengthen this document. We refer the reader to Engle [1] for further details. Generalisations such as the GARCH, EGARCH, IGARCH are also used in practice (see [3] for details on GARCH). This would take us too far afield.

Indeed, such models are fitted by the methods of quasi-maximum likelihood. This involvess minimizing a loss function (the negative (quasi-)log-likelihood). In particular, we learned about the richness of this data in its stylistic features and its possible modelling avenues. This was of interest to group members, and will help inform our choice of data for Project 1.

7 Conclusion

We looked at Apple stock data in detail, exploring its structure and uncovering insights into the data. We looked at more advanced features of the data, by exploring the log-returns and looking at its stylised features. We briefly mentioned the methods used to model financial time series and cut the discussion short upon noting that it may be too complicated for analysis.

Note: It may be noted that time series models do appear under `Appropriate Content` in Assessment 1 guidance and so our goal was to look at this data type to assess its feasibility for our group to analyse data like this in Assessment 1.

8 References

1. Engle, Robert F. "Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation." Econometrica: Journal of the econometric society (1982): 987-1007.
2. Lecture Notes on Financial Time Series: Accessed 9:09 on 27/09/2024.
3. Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. Journal of Econometrics, 31:307–327.
4. Apple Hits a $2tn Market Capitalisation.
5. Tutorial on Yahoo Finance API.
6. Leverage effect: Source 1.
7. Leverage effect: Source 2
8. Leverage effect: Source 3.