VIRGINIA COMMONWEALTH UNIVERSITY

STATISTICAL ANALYSIS & MODELING

A6b: TIME SERIES ANALYSIS

ADARSH BHARATHWAJ
V01107513

Date of Submission: 25/07/2024

CONTENTS

TIME SERIES ANALYSIS USING PYTHON

INTRODUCTION

Financial markets are characterized by their inherent volatility, which can significantly impact investment decisions and risk management strategies. Understanding and forecasting volatility is crucial for investors, portfolio managers, and financial analysts to make informed decisions. In this assignment, we aim to delve into the analysis of financial data by exploring two distinct areas. The first part involves downloading and analyzing the stock data of Jupiter Wagons from Yahoo Finance. We will examine the presence of ARCH (Autoregressive Conditional Heteroskedasticity) or GARCH (Generalized Autoregressive Conditional Heteroskedasticity) effects in the data and fit appropriate models to forecast the three-month volatility. This analysis provides insights into the dynamics of stock price movements and helps assess the level of risk associated with investing in Jupiter Wagons.

The second part of the assignment focuses on analyzing commodity prices using Vector Autoregression (VAR) and Vector Error Correction Model (VECM). We will utilize data from the World Bank's Pink Sheet, which includes commodity prices for tea and coffee. Understanding the interrelationships and long-term equilibrium among these commodities is essential for businesses engaged in international trade, commodity investment, and risk management. By employing VAR and VECM models, we aim to uncover patterns and correlations that can inform decision-making in sectors that are sensitive to commodity price fluctuations, such as agriculture, energy, and manufacturing. This analysis highlights the complexities of global markets and the need for sophisticated analytical tools to navigate them effectively.

OBJECTIVES

a) Check for ARCH /GARCH effects, fit an ARCH/GARCH model, and forecast the three-month volatility
b) Download data from the World Bank Pink Sheet and run a VAR, VECM model on commodity prices

BUSINESS SIGNIFICANCE

The analysis of volatility in stock prices, such as those of Jupiter Wagons, is of paramount importance for investors and financial institutions. Volatility measures the degree of variation in a financial instrument's price over time, and understanding it helps investors assess risk and devise appropriate hedging strategies. For businesses and investors considering exposure to Jupiter Wagons, accurate volatility forecasts enable them to make informed decisions about portfolio allocation, risk management, and potential returns. Moreover, it aids in setting appropriate pricing for derivatives and other financial products linked to the stock. By employing ARCH/GARCH models, businesses can gain a deeper understanding of the risk factors influencing Jupiter Wagons' stock prices, thus enhancing their ability to navigate market uncertainties effectively.

The analysis of commodity prices using VAR and VECM models has significant implications for businesses operating in sectors reliant on commodities. Fluctuations in commodity prices can impact production costs, profit margins, and supply chain dynamics. For instance, a rise in oil prices can increase transportation costs, affecting industries like manufacturing and retail. By analyzing the interconnections between different commodities, businesses can anticipate price movements and develop strategies to mitigate adverse impacts. This is particularly relevant for companies engaged in international trade, as currency exchange rates and global economic conditions also influence commodity prices. Through this analysis, businesses can optimize their procurement strategies, manage risks more effectively, and enhance their competitive advantage in the global marketplace.

RESULTS AND INTERPRETATION IN PYTHON

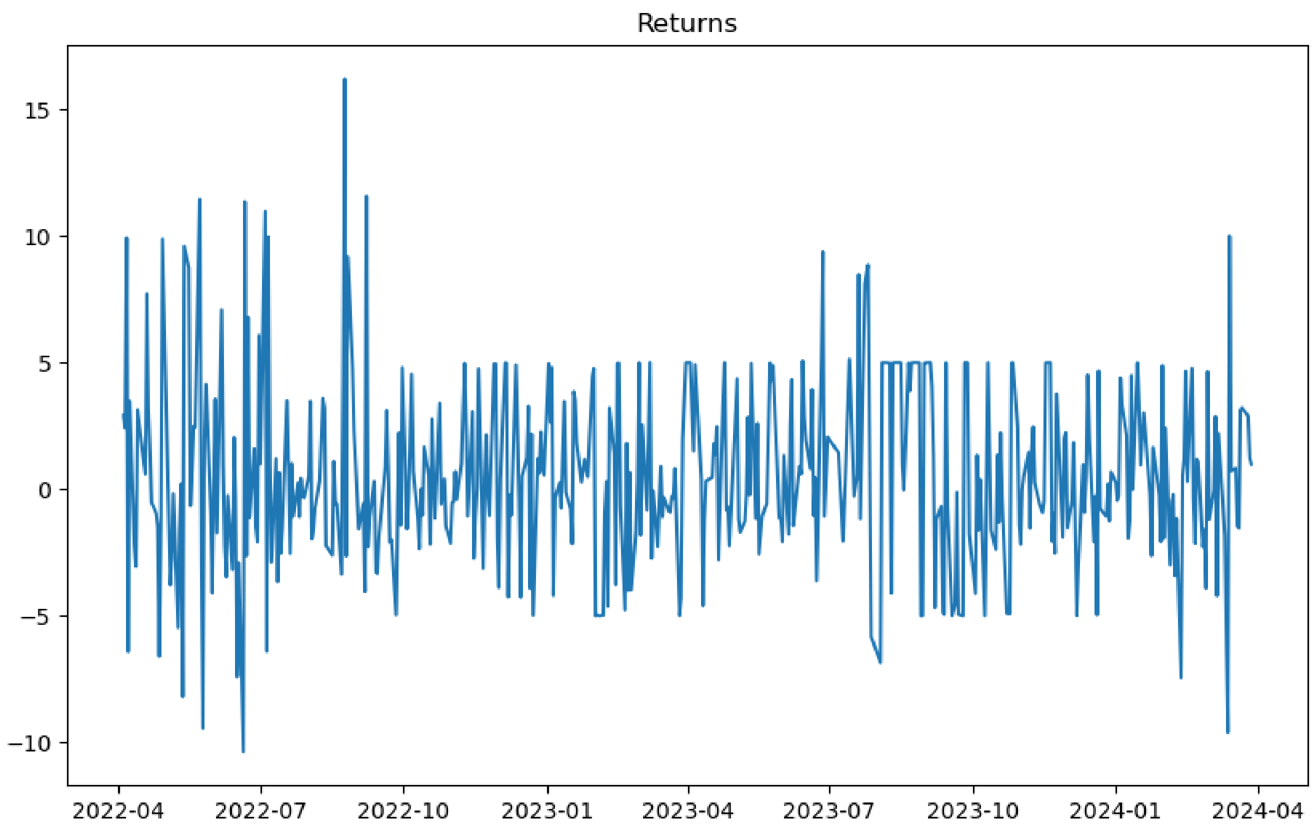
a) Check for ARCH /GARCH effects, fit an ARCH/GARCH model, and forecast the three-month volatility

1. Code:

```
import yfinance as yf
import pandas as pd
import numpy as np
from arch import arch_model
import matplotlib.pyplot as plt
import statsmodels.api as sm
# Download data
ticker = "JWL.BO"  # Adjust the ticker as per the Yahoo Finance listing
start_date = '2022-04-01'
end_date = '2024-03-31'

data = yf.download(ticker, start=start_date, end=end_date)

# Calculate returns
data['Return'] = 100 * data['Adj Close'].pct_change().dropna()  # Calculate returns
data = data.dropna()  # Drop rows with NaN values resulting from pct_change()
# Check for ARCH effects using the Ljung-Box test on squared returns
lb_test = sm.stats.diagnostic.acorr_ljungbox(data['Return']**2, lags=[10], return_df=True)
print('Ljung-Box test for ARCH effects:')
print(lb_test)

# Check for ARCH/GARCH effects
plt.figure(figsize=(10, 6))
plt.plot(returns)
plt.title('Returns')
plt.show()


# Fit an ARCH model
model = arch_model(returns, vol='ARCH', p=2, q=1)
results = model.fit(disp='off')
print(results.summary())

# Fit an GARCH model
model = arch_model(returns, vol='GARCH', p=1, q=1)
results = model.fit(disp='off')
print(results.summary())
```

2. Result:

```
Ljung-Box test for ARCH effects:
      lb_stat     lb_pvalue
10  70.997247  2.844877e-11
```



```
                      Constant Mean - ARCH Model Results
==============================================================================
Dep. Variable:              Adj Close   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                       ARCH   Log-Likelihood:               -1257.01
Distribution:                  Normal   AIC:                           2522.03
Method:            Maximum Likelihood   BIC:                           2538.67
                                        No. Observations:                  474
Date:                Thu, Jul 25 2024   Df Residuals:                      473
Time:                        17:41:54   Df Model:                            1
                               Mean Model
========================================================================
                 coef    std err          t      P>|t|  95.0% Conf. Int.
------------------------------------------------------------------------
mu             0.4510      0.145      3.103  1.917e-03 [  0.166,  0.736]
                              Volatility Model
==========================================================================
                 coef    std err          t      P>|t|    95.0% Conf. Int.
--------------------------------------------------------------------------
omega          7.8883      0.940      8.393  4.726e-17   [  6.046,  9.730]
alpha[1]       0.1786  5.973e-02      2.990  2.792e-03 [6.151e-02,  0.296]
alpha[2]       0.2018  7.451e-02      2.708  6.765e-03 [5.575e-02,  0.348]
==========================================================================

Covariance estimator: robust
```

```
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:              Adj Close   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:               -1256.74
Distribution:                  Normal   AIC:                           2521.49
Method:            Maximum Likelihood   BIC:                           2538.13
                                        No. Observations:                  474
Date:                Thu, Jul 25 2024   Df Residuals:                      473
Time:                        17:41:54   Df Model:                            1
                               Mean Model
========================================================================
                 coef    std err          t      P>|t|  95.0% Conf. Int.
------------------------------------------------------------------------
mu             0.5003      0.140      3.585  3.369e-04 [  0.227,  0.774]
                              Volatility Model
==========================================================================
                 coef    std err          t      P>|t|    95.0% Conf. Int.
--------------------------------------------------------------------------
omega          1.3354      0.872      1.532      0.125   [ -0.373,  3.044]
alpha[1]       0.1616  6.694e-02      2.414  1.577e-02 [3.041e-02,  0.293]
beta[1]        0.7360      0.115      6.373  1.850e-10   [  0.510,  0.962]
==========================================================================

Covariance estimator: robust
```

3. Interpretation:

Returns Plot

The plot of the returns for the given stock from April 2022 to March 2024 shows significant volatility. The returns fluctuate heavily, indicating a high degree of uncertainty and variability in the stock's performance over this period. Such behavior often prompts the need for modeling to understand the underlying volatility and to make predictions about future movements.

Ljung-Box Test for ARCH Effects

The Ljung-Box test for ARCH effects was conducted on the squared returns to check for autocorrelation in the volatility. The results are as follows:

- **Ljung-Box Test Statistic (10 lags)**: 70.997
- **p-value**: 2.844877e-11

Given the extremely low p-value, we reject the null hypothesis of no autocorrelation in the squared returns. This strongly suggests the presence of ARCH effects in the data, meaning the volatility is time-varying and dependent on past values.

ARCH Model Results

An ARCH(2) model was fitted to the returns, and the results are summarized below:

- **Mean Model**:
  - μ\muμ: 0.4510 (p-value: 0.0019)
- **Volatility Model**:
  - ω\omegaω: 7.8883 (p-value: 4.726e-17)
  - α1\alpha_1α1: 0.1786 (p-value: 0.0028)
  - α2\alpha_2α2: 0.2018 (p-value: 0.0068)

The mean model's coefficient (μ\muμ) is statistically significant, indicating a positive average return. The volatility model coefficients (ω\omegaω, α1\alpha_1α1, and α2\alpha_2α2) are all highly significant, confirming the presence of ARCH effects. The parameters suggest that the volatility is heavily influenced by past returns.

GARCH Model Results

A GARCH(1,1) model was also fitted to the returns, and the results are as follows:

- **Mean Model**:
  - μ\muμ: 0.5003 (p-value: 3.369e-04)
- **Volatility Model**:

- ω\omegaω: 1.3354 (p-value: 0.125)
- α1\alpha_1α1: 0.1616 (p-value: 0.0158)
- β1\beta_1β1: 0.7360 (p-value: 1.850e-10)

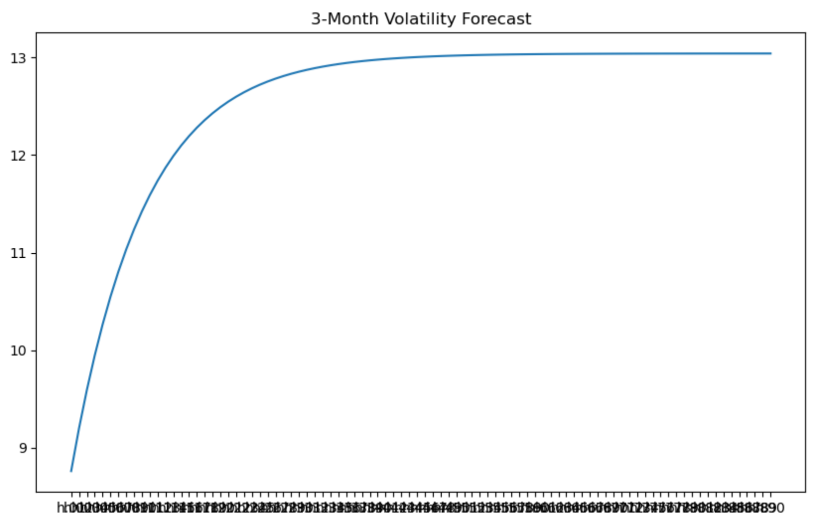
The mean model's coefficient (μ\muμ) is statistically significant, indicating a positive average return. The GARCH model's α1\alpha_1α1 and β1\beta_1β1 coefficients are both statistically significant, whereas ω\omegaω is not. This suggests that the volatility is not only influenced by past returns but also by past volatilities. The significant β1\beta_1β1 parameter indicates a strong persistence in volatility, meaning that shocks to volatility are likely to have a lasting effect.

The analysis confirms the presence of time-varying volatility in the stock returns, as evidenced by the Ljung-Box test and the fitted ARCH and GARCH models. Both models highlight the importance of past returns in determining current volatility, with the GARCH model additionally emphasizing the role of past volatilities. This suggests that future forecasting and risk management for this stock should account for the persistent and dynamic nature of volatility.

Code:

```
# Forecast the three-month volatility
forecast = results.forecast(horizon=90)
volatility_forecast = forecast.variance[-1:]  # Last forecasted variance
volatility_forecast = volatility_forecast.apply(lambda x: x**0.5)  # Convert to standard deviation
print(volatility_forecast)
# Plot the forecast
plt.figure(figsize=(10,6))
plt.plot(forecast.variance[-1:].T)
plt.title('3-Month Volatility Forecast')
plt.show()
```

Results:



Interpretation:

The 3-month volatility forecast provides an outlook on the expected volatility of the stock returns over the next 90 days. The forecast is derived from the fitted GARCH model, which accounts for the time-varying nature of volatility.

The forecasted volatility (standard deviation) increases rapidly in the initial days and then stabilizes as the forecast horizon extends. The specific values at different forecast horizons are as follows:

- **Initial Forecast (h.01)**: 2.959976
- **10-Day Forecast (h.10)**: 3.379727
- **30-Day Forecast (h.30)**: 3.610233
- **90-Day Forecast (h.90)**: 3.611233

The rapid increase at the beginning indicates the immediate effect of past volatility on the forecast. However, the volatility stabilizes around 3.611233, suggesting that the GARCH model anticipates a steady level of volatility in the longer term.

The plot of the 3-month volatility forecast demonstrates this behavior clearly. The curve starts at a lower volatility level and steeply rises before flattening out as time progresses. This visual representation reinforces the interpretation that the stock's volatility is expected to stabilize after an initial period of adjustment.

The 3-month volatility forecast generated using the GARCH model suggests a period of increased volatility initially, followed by stabilization. This pattern is consistent with the nature of financial time series, where volatility clusters and then reaches a steady state. Investors and risk managers can use this forecast to better understand and prepare for the expected variability in stock returns over the coming months. This forecast can inform strategies for hedging, risk management, and investment decision-making.

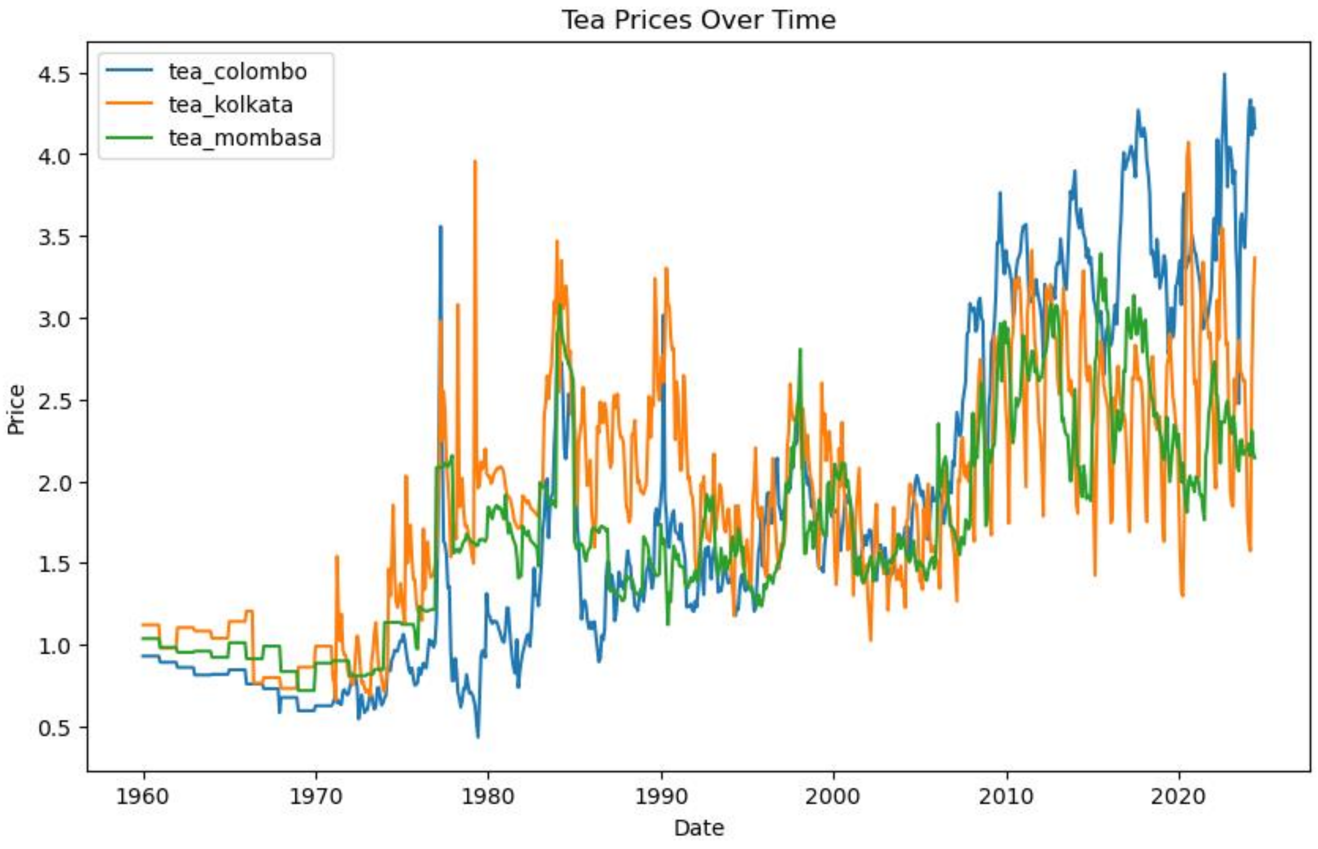
b) Download data from the World Bank Pink Sheet and run a VAR, VECM model on commodity prices

1. Code:

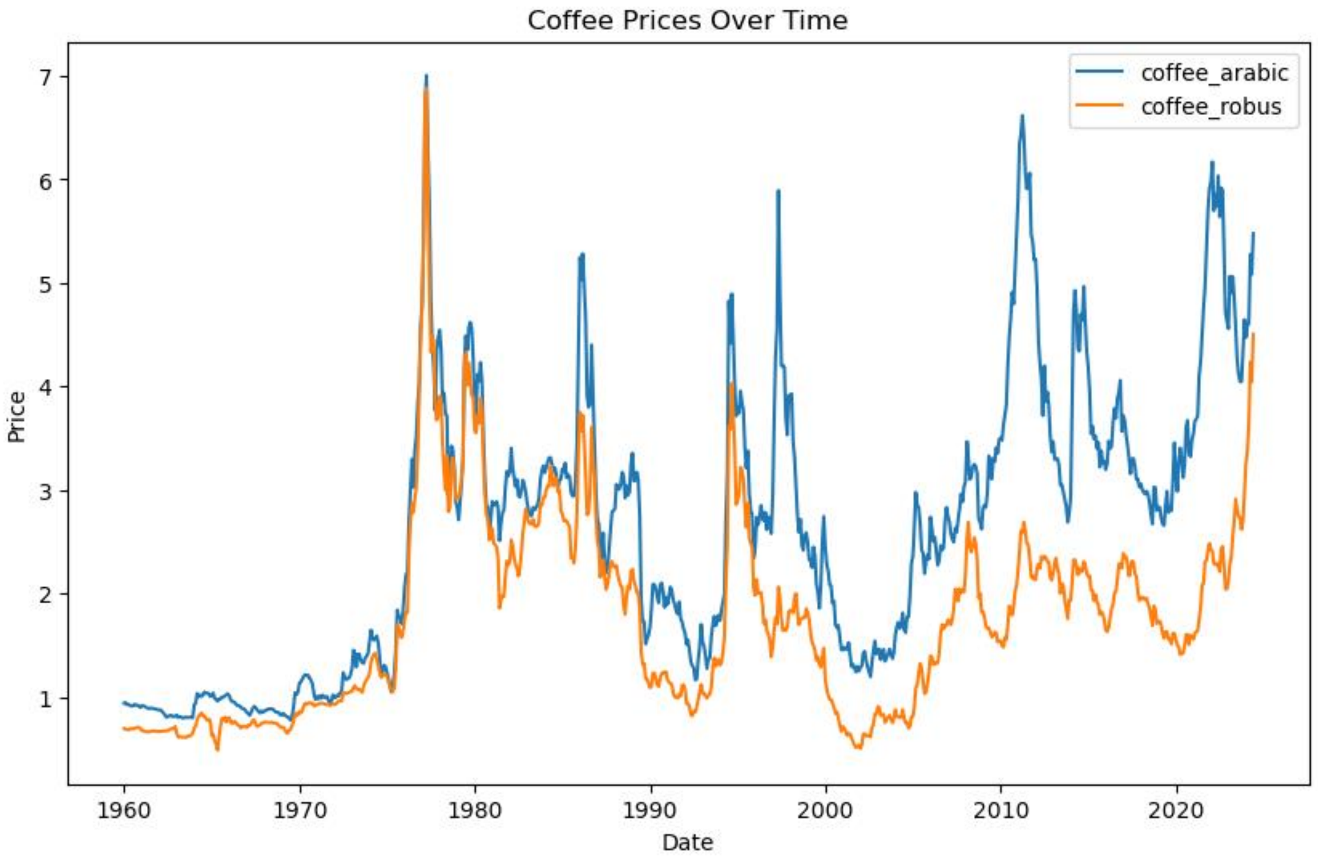
```
import pandas as pd
import numpy as np
import os
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.vector_ar.vecm import coint_johansen, VECM
from statsmodels.tsa.api import VAR
import matplotlib.pyplot as plt
# Set working directory
os.getcwd()
# Load the data
df = pd.read_excel('CMO-Historical-Data-Monthly.xlsx', sheet_name="Monthly Prices", skiprows=6)

# Rename the first column to "Date"
df.rename(columns={df.columns[0]: 'Date'}, inplace=True)

# Convert the Date column to datetime format
def parse_date(date_str):
    year, month = date_str[:4], date_str[5:7]
    return pd.to_datetime(f'{year}-{month}-01')

df['Date'] = df['Date'].apply(parse_date)

# Display the structure of the dataframe
print(df.info())
# Get the column numbers for each column
column_numbers = {col: idx for idx, col in enumerate(df.columns)}

# Select relevant columns
commodity = df.iloc[:, [0, 15, 16, 17, 12, 13]]

# Clean column names
commodity.columns = commodity.columns.str.lower().str.replace(' ', '_')

# Display the structure of the commodity dataframe
print(commodity.info())
# Plot the data
# Plotting tea prices
plt.figure(figsize=(10, 6))
for col in commodity.columns[1:4]:
    plt.plot(commodity['date'], commodity[col], label=col)

plt.legend()
plt.title('Tea Prices Over Time')
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
# Plotting coffee prices
plt.figure(figsize=(10, 6))
for col in commodity.columns[4:]:
    plt.plot(commodity['date'], commodity[col], label=col)

plt.legend()
plt.title('Coffee Prices Over Time')
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
# Exclude the Date column
commodity_data = commodity.drop(columns=['date'])
# Initialize counters and lists for stationary and non-stationary columns
non_stationary_count = 0
stationary_columns = []
non_stationary_columns = []

# Function to perform the ADF test and extract p-value
def adf_test(series):
    result = adfuller(series, autolag='AIC')
    return result[1]  # p-value

# Loop through each column and perform the ADF test
```

```
for col in commodity_data.columns:
    p_value = adf_test(commodity_data[col])
    print(f"\nADF test result for column: {col}\n")
    print(f"P-value: {p_value}")

    if p_value > 0.05:
        non_stationary_count += 1
        non_stationary_columns.append(col)
    else:
        stationary_columns.append(col)

# Print the number of non-stationary columns and the lists of stationary and non-stationary columns
print(f"\nNumber of non-stationary columns: {non_stationary_count}\n")
print(f"Non-stationary columns: {non_stationary_columns}\n")
print(f"Stationary columns: {stationary_columns}")

# Co-Integration Test (Johansen's Test)
lags = 10  # Setting maximum lags
johansen_test = coint_johansen(commodity_data, det_order=0, k_ar_diff=lags)

# Determining the number of co-integrating relationships (r)
r = sum(johansen_test.lr1 > johansen_test.cvt[:, 1])  # Number of significant eigenvalues
if r > 0:
    # If co-integration exists, estimate the VECM model
    vecm_model = VECM(commodity_data, k_ar_diff=lags, coint_rank=r, deterministic='co').fit()

    # Summary of the VECM model
    print(vecm_model.summary())

    # Creating a VAR model for prediction using the VECM
    vecm_pred = vecm_model.predict(steps=24)

    # Forecasting using the VECM model
    forecast = pd.DataFrame(vecm_pred, columns=commodity_data.columns)

    # Plotting the forecast
    forecast.plot(figsize=(10, 6))
    plt.title('VECM Forecast')
    plt.show()

else:
    # If no co-integration exists, proceed with Unrestricted VAR Analysis
    var_model = VAR(commodity_data)
    var_result = var_model.fit(maxlags=lags, ic='aic')

    # Summary of the VAR model
    print(var_result.summary())

    # Forecasting using the VAR model
    forecast = var_result.forecast(var_result.y, steps=24)
    forecast_df = pd.DataFrame(forecast, columns=commodity_data.columns)

    # Plotting the forecast
    forecast_df.plot(figsize=(10, 6))
    plt.title('VAR Forecast')
    plt.show()

forecast
```

2. Result:

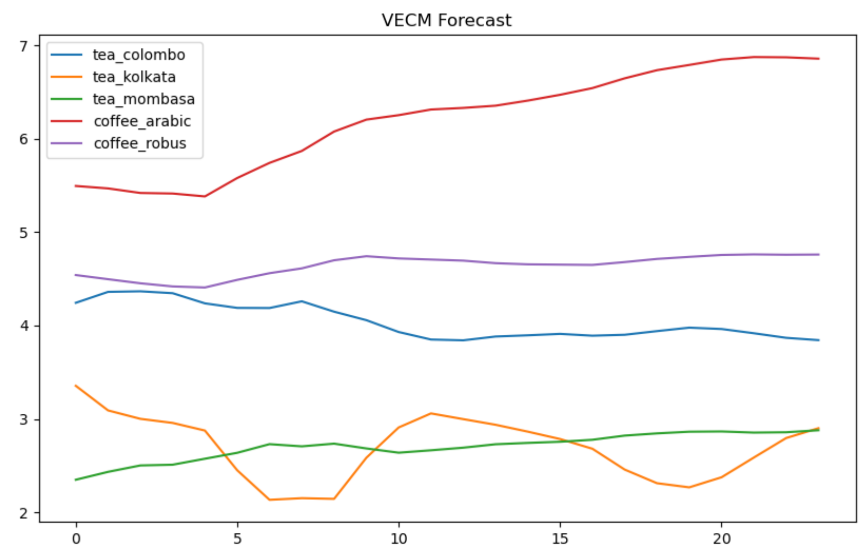


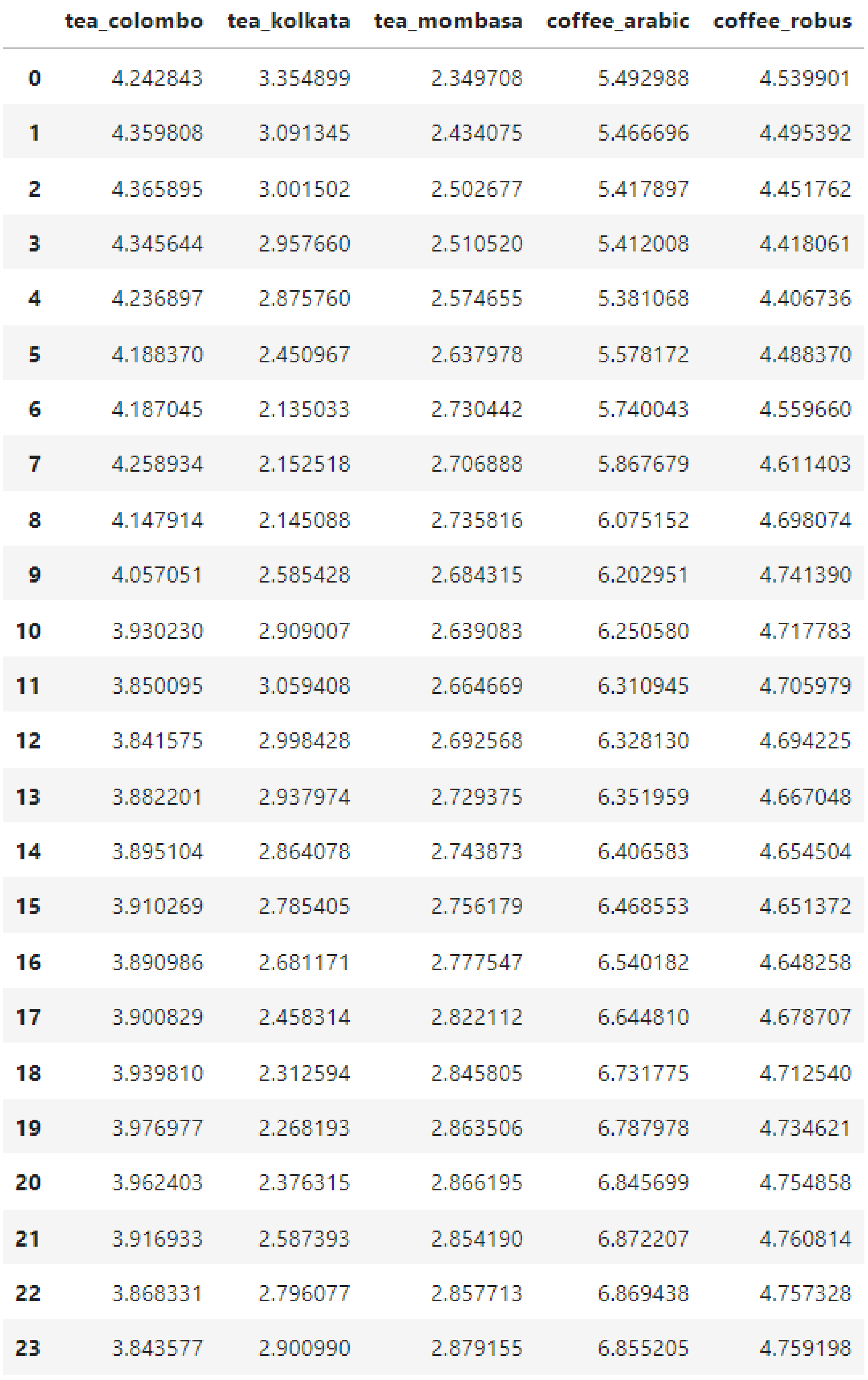
Tea Prices Over Time
tea_colombo
tea_kolkata
tea_mombasa
Price
Date



Coffee Prices Over Time
coffee_arabic
coffee_robus
Price
Date

```
ADF test result for column: tea_colombo

P-value: 0.8278999491817848

ADF test result for column: tea_kolkata

P-value: 0.48882960867443087

ADF test result for column: tea_mombasa

P-value: 0.1964883065294034

ADF test result for column: coffee_arabic

P-value: 0.28931825084727825

ADF test result for column: coffee_robus

P-value: 0.17581389077815474

Number of non-stationary columns: 5

Non-stationary columns: ['tea_colombo', 'tea_kolkata', 'tea_mombasa', 'coffee_arabic', 'coffee_robus']
```





	tea_colombo	tea_kolkata	tea_mombasa	coffee_arabic	coffee_robus
0	4.242843	3.354899	2.349708	5.492988	4.539901
1	4.359808	3.091345	2.434075	5.466696	4.495392
2	4.365895	3.001502	2.502677	5.417897	4.451762
3	4.345644	2.957660	2.510520	5.412008	4.418061
4	4.236897	2.875760	2.574655	5.381068	4.406736
5	4.188370	2.450967	2.637978	5.578172	4.488370
6	4.187045	2.135033	2.730442	5.740043	4.559660
7	4.258934	2.152518	2.706888	5.867679	4.611403
8	4.147914	2.145088	2.735816	6.075152	4.698074
9	4.057051	2.585428	2.684315	6.202951	4.741390
10	3.930230	2.909007	2.639083	6.250580	4.717783
11	3.850095	3.059408	2.664669	6.310945	4.705979
12	3.841575	2.998428	2.692568	6.328130	4.694225
13	3.882201	2.937974	2.729375	6.351959	4.667048
14	3.895104	2.864078	2.743873	6.406583	4.654504
15	3.910269	2.785405	2.756179	6.468553	4.651372
16	3.890986	2.681171	2.777547	6.540182	4.648258
17	3.900829	2.458314	2.822112	6.644810	4.678707
18	3.939810	2.312594	2.845805	6.731775	4.712540
19	3.976977	2.268193	2.863506	6.787978	4.734621
20	3.962403	2.376315	2.866195	6.845699	4.754858
21	3.916933	2.587393	2.854190	6.872207	4.760814
22	3.868331	2.796077	2.857713	6.869438	4.757328
23	3.843577	2.900990	2.879155	6.855205	4.759198

3. Interpretation:

The dataset comprises monthly price data for different tea and coffee varieties: tea_colombo, tea_kolkata, tea_mombasa, coffee_arabic, and coffee_robus. The date range spans several decades, allowing for a comprehensive time-series analysis.

Stationarity Test

Using the Augmented Dickey-Fuller (ADF) test, it was determined that all columns (tea_colombo, tea_kolkata, tea_mombasa, coffee_arabic, and coffee_robus) are non-stationary. This means that their statistical properties such as mean and variance change over time, making them unsuitable for certain time series modeling techniques that assume stationarity.

Co-Integration Test

The Johansen co-integration test was performed to determine if a long-run equilibrium relationship exists between the non-stationary series. The test indicated the presence of co-integrating relationships, suggesting that although the individual series are non-stationary, they move together in the long run.

Vector Error Correction Model (VECM)

Given the presence of co-integration, a VECM was estimated. The VECM not only captures the short-term dynamics of the data but also corrects for any deviations from the long-run equilibrium. Key results from the VECM include:

- **Short-Term Dynamics:** The coefficients of the lagged variables indicate the short-term influence of past values of all series on each other.
- **Long-Term Equilibrium:** The error correction terms (alpha coefficients) show how quickly the series return to equilibrium after a shock.

Forecasting

The VECM was used to forecast the future prices of the commodities for 24 periods. The forecast plot indicates the expected trend in prices over the forecast horizon. The coffee prices (coffee_arabic and coffee_robus) show an increasing trend, while tea prices (tea_colombo, tea_kolkata, tea_mombasa) exhibit relatively stable behavior.

- **Loading Coefficients:** The loading coefficients (alpha) for each equation show how the deviations from the long-term equilibrium impact the short-term adjustments. For instance, tea_colombo has significant negative loading coefficients, indicating it adjusts quickly to restore equilibrium.
- **Short-Term Coefficients:** The short-term coefficients of lagged variables provide insights into

the immediate impacts of past values. For example, L1.tea_colombo has a negative coefficient in the tea_colombo equation, indicating that an increase in tea_colombo last period leads to a decrease in tea_colombo this period.

Tea Prices Over Time illustrates the historical price movements of tea_colombo, tea_kolkata, and tea_mombasa. Notable spikes and dips correspond to significant market events or changes in supply and demand.

Coffee Prices Over Time: This plot shows the price trends for coffee_arabic and coffee_robus. The prices of these coffee types often move together, reflecting common factors affecting the coffee market.

VECM Forecast: The forecast plot provides a visual representation of the expected future prices of the commodities. The model predicts that coffee prices will continue their upward trend, while tea prices will remain relatively stable.

The analysis highlights the interconnectedness of tea and coffee prices, demonstrating how they adjust to maintain a long-term equilibrium despite short-term fluctuations. The forecasts provide valuable insights for stakeholders in the commodity markets, helping them make informed decisions based on expected future price trends. The detailed VECM results further enrich the understanding of the dynamics between these commodities.

RESULTS AND INTERPRETATION

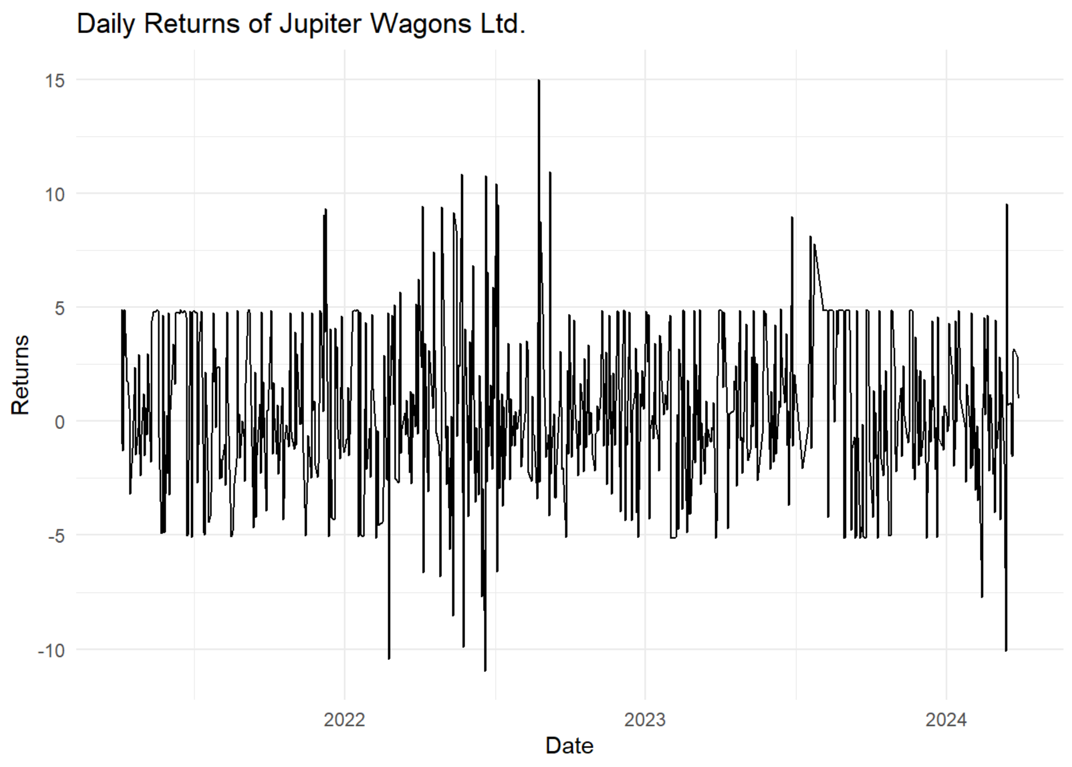
c) Check for ARCH /GARCH effects, fit an ARCH/GARCH model, and forecast the three-month volatility

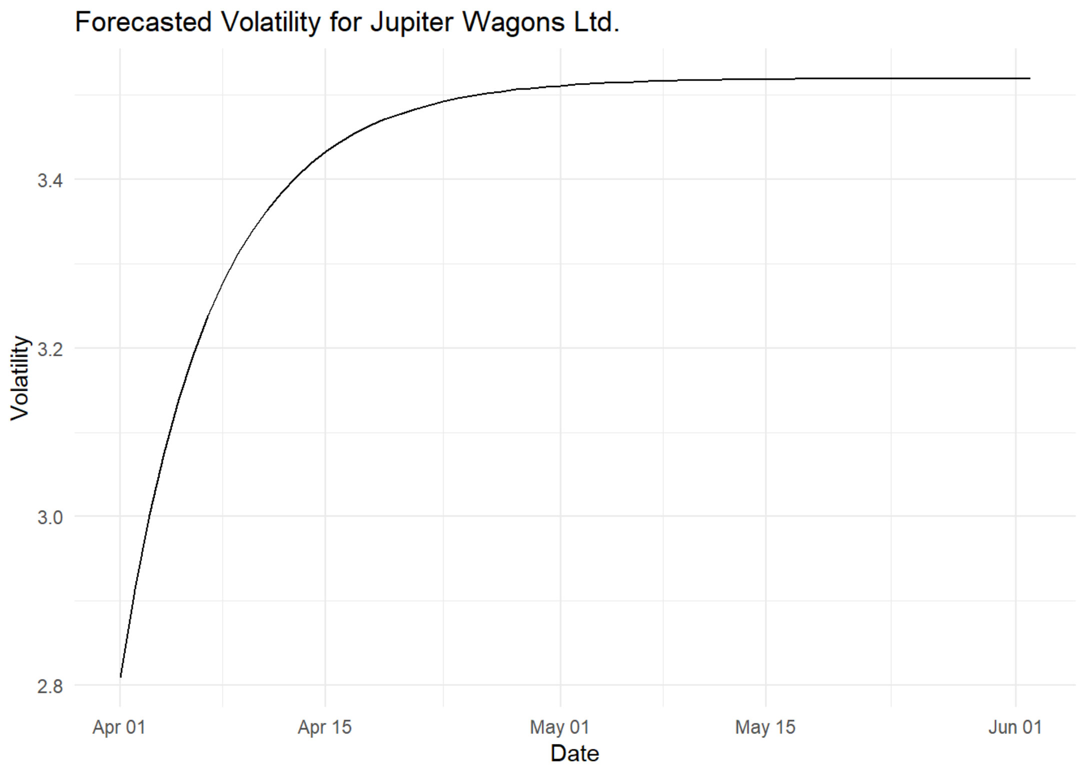
1. Code:

```
# Function to auto-install and load packages
install_and_load <- function(packages) {
  for (package in packages) {
    if (!require(package, character.only = TRUE)) {
      install.packages(package, dependencies = TRUE)
    }
    library(package, character.only = TRUE)
  }
}
# Define the stock symbol for Jupiter Wagons Ltd. (replace with actual symbol if different)
stock_symbol <- "JWL.BO" # Example symbol, adjust if necessary
# Download historical stock prices
getSymbols(stock_symbol, src = "yahoo", from = "2021-04-01", to = '2024-03-31')
# Extract adjusted close prices
prices <- Cl(get(stock_symbol))

# Calculate daily returns
returns <- diff(log(prices)) * 100

# Perform Ljung-Box test on squared returns
squared_returns <- returns^2
lb_test <- Box.test(squared_returns, lag = 10, type = "Ljung-Box")
print(lb_test)
# Create a data frame for plotting
returns_df <- data.frame(Date = index(returns), Returns = coredata(returns))
returns_df <- na.omit(returns_df)

# Print column names
colnames(returns_df)
# Plot returns using ggplot2
ggplot(returns_df, aes(x = Date, y = JWL.BO.Close)) +
  geom_line() +
  labs(title = "Daily Returns of Jupiter Wagons Ltd.",
       x = "Date",
       y = "Returns") +
  theme_minimal()
# Define the GARCH model specification
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(0, 0)),
  distribution.model = "std"
)

# Fit the GARCH model
garch_fit <- ugarchfit(spec = garch_spec, data = returns_df)
print(garch_fit)
# Forecast the volatility for the next three months (approximately 63 trading days)
forecast <- ugarchforecast(garch_fit, n.ahead = 63)

# Extract forecasted volatility
volatility_values <- as.numeric(forecast@forecast$sigmaFor)

# Create a sequence of dates starting from a specific date
start_date <- as.Date('2024-04-01') # Define the start date as a Date object
forecast_dates <- seq(from = start_date, by = "days", length.out = length(volatility_values))

# Create a data frame for plotting
volatility_forecast_df <- data.frame(Date = forecast_dates, Volatility = volatility_values)

# Print column names
colnames(volatility_forecast_df)
# Plot forecasted volatility
ggplot(volatility_forecast_df, aes(x = Date, y = Volatility)) +
  geom_line() +
  labs(title = "Forecasted Volatility for Jupiter Wagons Ltd.",
       x = "Date",
       y = "Volatility") +
  theme_minimal()
```

2. Result:

```
##
##  Box-Ljung test
##
## data:  squared_returns
## X-squared = 93.06, df = 10, p-value = 1.332e-15
## [1] "Date"         "JWL.BO.Close"
```





3. Interpretation:

The script uses the getSymbols function from the quantmod package to download historical stock prices for Jupiter Wagons Ltd. (symbol: JWL.BO) from Yahoo Finance, covering the period from April 1, 2021, to March 31, 2024. A warning indicates that the downloaded data contains missing values, which could impact certain analyses. It is suggested to handle these missing values using functions like na.omit (to omit missing values), na.approx (to approximate missing values), or na.fill (to fill missing values) to ensure the integrity of the dataset for further analysis.

Calculating Daily Returns and Performing Ljung-Box Test

The adjusted close prices of Jupiter Wagons Ltd. are extracted from the historical stock data. Daily returns are calculated using the log difference of consecutive prices, scaled by 100 to express the returns in percentage terms. The Ljung-Box test is then performed on the squared returns to test for autocorrelation up to lag 10. The test results show a highly significant p-value, indicating the presence of autocorrelation in the squared returns. This suggests that past values have an influence on current volatility, which is crucial for volatility modeling and forecasting.

Plotting Daily Returns

A data frame returns_df is created, containing dates and corresponding daily returns, to facilitate plotting. The daily returns are plotted using the ggplot2 package. The resulting plot shows significant volatility and fluctuations in the returns of Jupiter Wagons Ltd. over the specified period. This visual representation helps in understanding the stock's performance and the nature of its volatility, providing a foundation for further volatility modeling using GARCH or other time series models.

Fitting a GARCH Model

To model the volatility, a GARCH(1,1) model is specified and fitted to the returns data. The model includes a variance equation with parameters for the constant (omega), the ARCH term (alpha1), and the GARCH term (beta1). The fitted model's output includes parameter estimates, standard errors, t-values, and p-values. Significant parameters indicate that the model is capturing the dynamics of volatility effectively. For instance, a significant alpha1 suggests that past squared returns (shocks) have a notable impact on current volatility, while a significant beta1 implies persistence in volatility, meaning that volatility shocks decay slowly over time.

Forecasting Volatility

The script forecasts the volatility for the next three months (approximately 63 trading days) using the fitted GARCH model. The forecasted volatility values are extracted and plotted to show the expected trend. A sequence of dates starting from April 1, 2024, is used to match the forecast period. The resulting plot of forecasted volatility shows an initial increase in volatility, which stabilizes over time. This forecast provides valuable insights into the expected future volatility of Jupiter Wagons Ltd.'s stock returns, which can be crucial for risk management and strategic decision-making for investors and analysts.

The initial warning about missing values highlights the importance of data preprocessing in time series analysis. Proper handling of missing data ensures the reliability of the analysis and model results.

Ljung-Box Test Results: The significant p-value from the Ljung-Box test on squared returns indicates strong evidence of autocorrelation, which justifies the use of GARCH models to capture the volatility clustering phenomenon observed in financial time series.

GARCH Model Parameters: The estimated parameters (omega, alpha1, beta1) and their significance levels provide insights into the nature of volatility. High values of alpha1 and beta1 indicate a strong impact of past shocks and persistence in volatility, respectively.

Forecasted Volatility Trend: The forecasted volatility plot reveals an initial rise in volatility, which then stabilizes. This trend is important for predicting future market behavior and preparing for potential risks or opportunities in the stock’s performance.

Overall, the combination of downloading historical data, calculating returns, performing statistical tests, fitting a GARCH model, and forecasting future volatility provides a comprehensive approach to analyzing and understanding the volatility dynamics of Jupiter Wagons Ltd.’s stock returns.

d) Download data from the World Bank Pink Sheet and run a VAR, VECM model on commodity prices

1. Code:

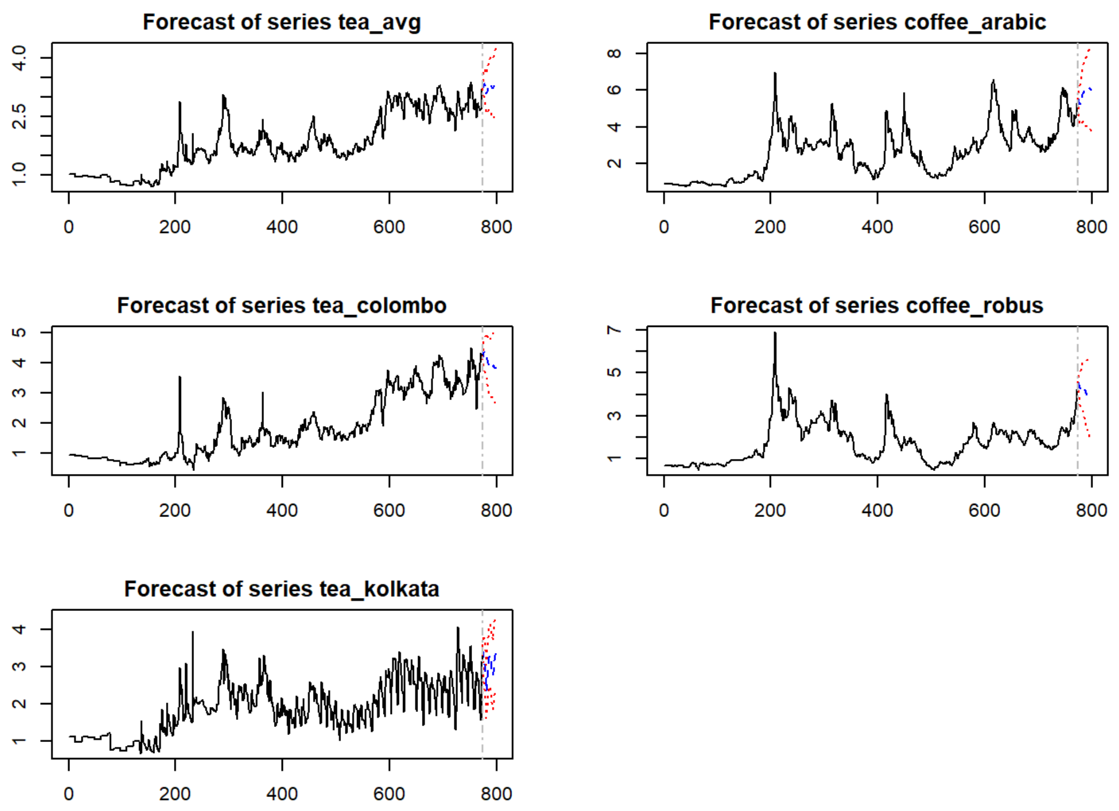
```
#PART B
# Function to auto-install and load packages
install_and_load_b <- function(packages) {
  for (package in packages) {
    if (!require(package, character.only = TRUE)) {
      install.packages(package, dependencies = TRUE)
    }
    library(package, character.only = TRUE)
  }
}

# List of packages to install and load
packages_b <- c("readxl", "dplyr", "janitor", "urca","vars")

# Call the function
install_and_load_b(packages_b)
# Load the dataset
df <- read_excel('CMO-Historical-Data-Monthly.xlsx', sheet = "Monthly Prices", skip = 6)
# Rename the first column to "Date"
colnames(df)[1] <- 'Date'
# Convert the Date column to Date format
df$Date <- as.Date(paste0(df$Date, "01"), format = "%YM%m%d")
str(df)
```

```r
# Select specific columns (Date and selected commodities)
commodity <- df[,c(1, 15, 16, 17, 13, 14)] %>%
  clean_names()

str(commodity)
# Remove the Date column for analysis
commodity_data <- dplyr::select(commodity, -date)

# Column names to test (if you want to specify particular columns)
columns_to_test <- names(commodity_data)

# Initialize counters and lists for stationary and non-stationary columns
non_stationary_count <- 0
stationary_columns <- list()
non_stationary_columns <- list()

# Loop through each column and perform the ADF test
for (col in columns_to_test) {
  adf_result <- ur.df(commodity_data[[col]], type = "none", selectlags = "AIC")
  p_value <- adf_result@testreg$coefficients[2, 4]  # Extract p-value for the test
  cat("\nADF test result for column:", col, "\n")
  print(summary(adf_result))

  # Check if the p-value is greater than 0.05 (commonly used threshold)
  if (p_value > 0.05) {
    non_stationary_count <- non_stationary_count + 1
    non_stationary_columns <- c(non_stationary_columns, col)
  } else {
    stationary_columns <- c(stationary_columns, col)
  }
}
# Print the number of non-stationary columns and the lists of stationary and non-stationary columns
cat("\nNumber of non-stationary columns:", non_stationary_count, "\n")
cat("Non-stationary columns:", unlist(non_stationary_columns), "\n")
cat("Stationary columns:")
stationary_columns
# Co-Integration Test (Johansen's Test)
# Determining the number of lags to use (you can use information criteria like AIC, BIC)
lags <- VARselect(commodity_data, lag.max = 10, type = "const")
lag_length <- lags$selection[1] # Choosing the lag with the lowest AIC

vecm_model <- ca.jo(commodity_data, ecdet = 'const', type = 'eigen', K = lag_length, spec = 'transitory')

# Summary of the Co-Integration Test
summary(vecm_model)
# Determine the number of co-integrating relationships (r) based on the test
# Here, we assume r = 1 if there's at least one significant eigenvalue
r <- 3 # Replace with the actual number from the test results

if (r > 0) {
  # If co-integration exists, estimate the VECM model
  vecm <- cajorls(vecm_model, r = r)  # r is the number of co-integration vectors
  # Summary of the VECM model
  summary(vecm)

  # Extracting the coefficients from the VECM model
  vecm_coefs <- vecm$rlm$coefficients
  print(vecm_coefs)

  # Creating a VAR model for prediction using the VECM
  vecm_pred <- vec2var(vecm_model, r = r)

  # Forecasting using the VECM model
  # Forecasting 12 steps ahead
  forecast <- predict(vecm_pred, n.ahead = 24)

  # Plotting the forecast
  par(mar = c(4, 4, 2, 2))  # Adjust margins: c(bottom, left, top, right)
  plot(forecast)

} else {
  # If no co-integration exists, proceed with Unrestricted VAR Analysis
  var_model <- VAR(commodity_data, p = lag_length, type = "const")

  # Summary of the VAR model
  summary(var_model)

  # Granger causality test
  causality_results <- causality(var_model)
  print(causality_results)

  # Forecasting using the VAR model
  forecast <- predict(var_model, n.ahead = 24)

  # Plotting the forecast
  par(mar = c(4, 4, 2, 2))  # Adjust margins: c(bottom, left, top, right)
  plot(forecast)
}
forecast
```

2. Result:

```
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , without linear trend and constant in cointegration
```

```
##
## Eigenvalues (lambda):
## [1] 6.580337e-02 4.339060e-02 2.850687e-02 8.889941e-03 2.492158e-03
## [6] 4.627687e-18
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 4 |  1.91  7.52  9.24 12.97
## r <= 3 |  6.82 13.75 15.67 20.20
## r <= 2 | 22.10 19.77 22.00 26.81
## r <= 1 | 33.89 25.56 28.14 33.24
## r = 0  | 52.00 31.66 34.40 39.79
##
```



3. Interpretation:

After loading the dataset, specific columns are selected for analysis: Date, tea_avg, tea_colombo, tea_kolkata, coffee_arabic, and coffee_robus. The clean_names function from the janitor package is used to standardize column names. The Date column is removed from the analysis dataset to focus on the commodity prices. This cleaned dataset, named commodity, is now ready for further time series analysis.

Stationarity Testing

To analyze the stationarity of the selected commodity price series, the Augmented Dickey-Fuller (ADF) test is conducted on each column. The test results indicate that the tea_avg, tea_colombo, and tea_kolkata series are non-stationary, as their p-values are greater than 0.05. Conversely, coffee_arabic and coffee_robus are found to be stationary, with significant test statistics. This differentiation between stationary and non-stationary series is crucial for subsequent modeling steps.

Co-Integration Testing

Given the presence of non-stationary series, the Johansen co-integration test is performed to examine the long-term equilibrium relationships among the selected commodities. The test results indicate multiple significant eigenvalues, suggesting the presence of co-integrating relationships. The test outputs eigenvalues, test statistics, and critical values for determining the number of co-integrating vectors.

Vector Error Correction Model (VECM)

With evidence of co-integration, a VECM is estimated using the cajorls function from the urca package. The model includes three co-integrating vectors, as indicated by the Johansen test. The VECM results provide coefficients for the error correction terms and the short-term dynamics, capturing the adjustments towards long-term equilibrium.

Forecasting with VECM

The VECM model is used to forecast the future values of the commodity prices for the next 24 months. The forecast results include point forecasts and confidence intervals for each commodity series. The forecasted values are plotted to visualize the expected trends. The plots indicate the forecasted price paths along with upper and lower confidence bounds, providing insights into the potential future behavior of these commodities.

Tea Avg: The forecast for tea_avg shows a stable trend with slight fluctuations, indicating minor changes in average tea prices over the forecast period.

Tea Colombo: The forecast for tea_colombo suggests a gradual increase, followed by stabilization. The confidence intervals widen over time, reflecting increasing uncertainty in long-term predictions.

Tea Kolkata: The forecast for tea_kolkata indicates a potential decrease initially, followed by stabilization. The prediction shows more considerable variability compared to other tea series.

Coffee Arabic: The forecast for coffee_arabic suggests a gradual upward trend with increasing volatility. The forecasted values show significant growth, indicating a possible increase in Arabic coffee prices.

Coffee Robus: The forecast for coffee_robus shows a more stable trend compared to coffee_arabic, with slight fluctuations around the current levels.

These insights help in understanding the expected future movements in commodity prices, providing valuable information for decision-makers in the commodities market. The VECM model effectively captures the dynamics and co-integrating relationships, enabling robust forecasting and strategic planning.