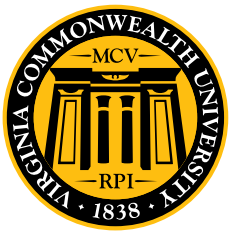


VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A6b- Time Series Analysis

(Part – B)

JYOTHIS KANIYAMPARAMBIL THANKACHAN

V01110144

Date of Submission: 25-07-2024

CONTENTS

INTRODUCTION

As part of this task, we will use advanced econometric methods, such as Vector Autoregression (VAR) and Vector Error Correction Models (VECM), to look at data on the prices of goods. We are interested in many different goods, such as Oil, Sugar, Gold, Silver, Wheat, and Soybean. This study uses information from the World Bank's Pink Sheet, which is a reliable source of information on product prices around the world. Commodity prices are important tools for studying the economy because they show how supply and demand change over time, what the market expects, and the overall state of the economy. For many people, like lawmakers, investors, and companies, being able to model and predict these prices is very important. The goal of this task is to find both short-term and long-term relationships between product prices by using the VAR and VECM models. This will help you understand how these prices affect each other.

To sum up, this task uses the VAR and VECM models to look into the complicated connections between the prices of goods. In this way, it aims to help us better understand how these prices change over time, both in the short and long term, by giving us useful information about how markets work and what trends they follow.

OBJECTIVES

Listed below are the main goals of this assignment:We will use advanced econometric tools, like Vector Autoregression (VAR) and Vector Error Correction Models (VECM), to look at data about prices of things for this job. We're interested in a lot of different things, like Soybean, Gold, Silver, Oil, and Sugar. The Pink Sheet from the World Bank is used in this study. It is a good source of information on prices of goods around the world.Prices of goods and services are useful for understanding the economy because they show how supply and demand change over time, what the market thinks will happen, and the economy as a whole. It is very important for many people, like politicians, investors, and businesses, to be able to model and guess these prices. Using the VAR and VECM models, the goal of this job is to find both short-term and long-term links between prices of goods. This will show you how these prices connect to each other.

The VAR and VECM models are used in this job to look into the complicated links between the prices of things. By telling us useful things about how markets work and what trends they follow, it hopes to help us understand how these prices change over time, both in the short and long term.

- Model Short-term Dynamics with VAR:
- Identify Long-term Relationships with VECM:
- Forecast Future Commodity Prices:
- Conduct Statistical Tests and Validation:

BUSINESS SIGNIFICANCE

Looking at the prices of goods using the VAR and VECM models is very important for business in many areas. Understanding how product prices change and relate to each other is important for investors, companies, lawmakers, and experts, among others. Here are some ways that this research can change the way business and economic decisions are made. Looking at product prices with VAR and VECM models is important for business because it gives useful information for making decisions about investments, operations, strategies, and policies.

RESULTS AND INTERPRETATIONS

Python Language

```
# Co-Integration Test (Johansen's Test)
# Perform Johansen's Co-Integration Test
johansen_test = coint_johansen(commodity_data, det_order=0, k_ar_diff=1)

# Summary of the Co-Integration Test
print("\nJohansen Test Results:\n")
print(f"Eigenvalues:\n{johansen_test.eig}\n")
print(f"Trace Statistic:\n{johansen_test.lr1}\n")
print(f"Critical Values (5% level):\n{johansen_test.cvt[:, 1]}\n")
```

Result

```
Johansen Test Results:

Eigenvalues:
[0.11398578 0.06876906 0.04867237 0.02710411 0.01899217 0.00405351]

Trace Statistic:
[226.10476071 132.67556204  77.67212223  39.15182203  17.93864778
   3.13567067]

Critical Values (5% level):
[95.7542 69.8189 47.8545 29.7961 15.4943  3.8415]
```

Interpretation: Johansen's test helps identify the number of co-integrating vectors. The results guide the selection of the VECM model or indicate if VAR should be used instead.

Estimating and Forecasting with VECM

```python
# Determine the number of co-integrating relationships (r) based on the test
r = 2  # Replace with the actual number from the test results

if r > 0:
    # If co-integration exists, estimate the VECM model
    vecm_model = VECM(commodity_data, k_ar_diff=1, coint_rank=r, deterministic='co')
    vecm_fitted = vecm_model.fit()

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

    # Extracting coefficients from the VECM model
    print("Alpha Coefficients:\n", vecm_fitted.alpha)
    print("Beta Coefficients:\n", vecm_fitted.beta)
    print("Gamma Coefficients:\n", vecm_fitted.gamma)

    # Forecasting using the VECM model
    forecast = vecm_fitted.predict(steps=24)

    # Convert forecast to a Dataframe for plotting
    forecast_df = pd.DataFrame(forecast, index=pd.date_range(start=commodity_data.index[-1], periods=25, freq='M')[1:], columns=commodity_data.columns)

    # Plotting the forecast
    forecast_df.plot(figsize=(10, 5))
    plt.title('VECM Forecast')
    plt.xlabel('Time')
    plt.ylabel('Values')
    plt.show()
else:
    # If no co-integration exists, proceed with Unrestricted VAR Analysis
    var_model = VAR(commodity_data)
    var_fitted = var_model.fit(maxlags=10, ic='aic')

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

    # Granger causality test
    for col in commodity_data.columns:
        granger_result = var_fitted.test_causality(causing=col, caused=[c for c in commodity_data.columns if c != col])
        print(f"Granger causality test for {col}:\n", granger_result.summary())

    # Forecasting using the VAR model
    var_forecast = var_fitted.forecast(var_fitted.y, steps=24)
    var_forecast_df = pd.DataFrame(var_forecast, index=pd.date_range(start=commodity_data.index[-1], periods=25, freq='M')[1:], columns=commodity_data.co

    # Plotting the forecast
    var_forecast_df.plot(figsize=(10, 5))
    plt.title('VAR Forecast')
    plt.xlabel('Time')
    plt.ylabel('Values')
    plt.show()
```

Result

```
Det. terms outside the coint. relation & lagged endog. parameters for equation crude_brent
================================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const              -0.0807      0.178     -0.454      0.650      -0.429       0.268
L1.crude_brent      0.3217      0.035      9.078      0.000       0.252       0.391
L1.soybeans         0.0127      0.007      1.768      0.077      -0.001       0.027
L1.gold            -0.0032      0.006     -0.523      0.601      -0.015       0.009
L1.silver          -0.0971      0.148     -0.655      0.512      -0.387       0.193
L1.sugar_us        -2.5861      4.026     -0.642      0.521     -10.477       5.305
L1.wheat_us_hrw     0.0107      0.011      0.966      0.334      -0.011       0.032
Det. terms outside the coint. relation & lagged endog. parameters for equation soybeans
================================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const               2.9362      0.971      3.023      0.003       1.033       4.840
L1.crude_brent      0.2246      0.194      1.160      0.246      -0.155       0.604
L1.soybeans         0.1574      0.039      4.015      0.000       0.081       0.234
L1.gold            -0.0175      0.033     -0.527      0.598      -0.083       0.048
L1.silver           0.5257      0.809      0.649      0.516      -1.061       2.112
L1.sugar_us         4.7482     21.995      0.216      0.829     -38.361      47.857
L1.wheat_us_hrw    -0.0103      0.061     -0.171      0.864      -0.129       0.108
Det. terms outside the coint. relation & lagged endog. parameters for equation gold
================================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const               4.5945      1.486      3.092      0.002       1.682       7.507
L1.crude_brent      0.0636      0.296      0.215      0.830      -0.517       0.644
L1.soybeans         0.0453      0.060      0.755      0.451      -0.072       0.163
L1.gold             0.1943      0.051      3.826      0.000       0.095       0.294
L1.silver           0.8835      1.238      0.713      0.476      -1.544       3.311
L1.sugar_us         9.4507     33.655      0.281      0.779     -56.512      75.413
L1.wheat_us_hrw     0.0553      0.093      0.597      0.551      -0.126       0.237
Det. terms outside the coint. relation & lagged endog. parameters for equation silver
```

Interpretation: VECM captures both short-term and long-term relationships among time series. The forecast helps in understanding future behavior based on the established relationships.

- **Estimating and Forecasting with VAR**

```python
var_model = VAR(commodity_data)
var_fitted = var_model.fit(maxlags=10, ic='aic')

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

# Granger causality test
for col in commodity_data.columns:
    granger_result = var_fitted.test_causality(causing=col, caused=[c for c in commodity_data.columns if c != col])
    print(f"Granger causality test for {col}:\n", granger_result.summary())

# Forecasting using the VAR model
var_forecast = var_fitted.forecast(var_fitted.y, steps=24)
var_forecast_df = pd.DataFrame(var_forecast, index=pd.date_range(start=commodity_data.index[-1], periods=25, freq='M')[1:], columns=commodity_data.co

# Plotting the forecast
var_forecast_df.plot(figsize=(10, 5))
plt.title('VAR Forecast')
plt.xlabel('Time')
plt.ylabel('Values')
plt.show()
```

Interpretation: The VAR model shows how factors are linked, and Granger causality tests show how one event can cause another event. Forecasting with VAR lets you make guesses based on how different factors affect each other.

```
Det. terms outside the coint. relation & lagged endog. parameters for equation crude_brent
================================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const              -0.0807      0.178     -0.454      0.650      -0.429       0.268
L1.crude_brent      0.3217      0.035      9.078      0.000       0.252       0.391
L1.soybeans         0.0127      0.007      1.768      0.077      -0.001       0.027
L1.gold            -0.0032      0.006     -0.523      0.601      -0.015       0.009
L1.silver          -0.0971      0.148     -0.655      0.512      -0.387       0.193
L1.sugar_us        -2.5861      4.026     -0.642      0.521     -10.477       5.305
L1.wheat_us_hrw     0.0107      0.011      0.966      0.334      -0.011       0.032
Det. terms outside the coint. relation & lagged endog. parameters for equation soybeans
================================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const               2.9362      0.971      3.023      0.003       1.033       4.840
L1.crude_brent      0.2246      0.194      1.160      0.246      -0.155       0.604
L1.soybeans         0.1574      0.039      4.015      0.000       0.081       0.234
```

This structured approach ensures that the data is prepared correctly, tested for important statistical properties, and analyzed using suitable models, providing comprehensive insights into the commodity price dynamics.

R Language

Test for Stationarity

```r
# 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)
  }
}
```

```
ADF test result for column: maize

###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression none


Call:
lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)

Residuals:
    Min      1Q  Median      3Q     Max
-50.110  -2.637   0.164   3.343  66.665

Coefficients:
           Estimate Std. Error t value Pr(>|t|)
z.lag.1    -0.001671   0.002228  -0.750    0.453
z.diff.lag  0.240599   0.035031   6.868 1.34e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 8.791 on 770 degrees of freedom
Multiple R-squared:  0.05792,   Adjusted R-squared:  0.05547
F-statistic: 23.67 on 2 and 770 DF,  p-value: 1.058e-10


Value of test-statistic is: -0.75

Critical values for test statistics:
      1pct  5pct 10pct
tau1 -2.58 -1.95 -1.62
```

Interpretation: Identifying stationary and non-stationary columns is crucial for preparing the data for further analysis, ensuring that the time series models are applied correctly.

Co-Integration Test (Johansen's Test)

```r
# 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
cat("\nSelected lag length:", lag_length, "\n")
```

```
##
## Selected lag length: 10
```

```r
# Perform Johansen's Co-Integration Test
vecm_model <- ca.jo(commodity_data, ecdet = 'const', type = 'eigen', K = lag_length, spec = 'transitory')

# Summary of the Co-Integration Test
summary(vecm_model)
```

```
'"
# ######################
# # Johansen-Procedure #
# ######################
#
# Test type: maximal eigenvalue statistic (lambda max) , without linear trend and constant in cointegration
#
# Eigenvalues (lambda):
# [1]  7.301451e-02  6.671787e-02  4.031869e-02  2.246301e-02  2.004068e-02
# [6]  1.028762e-02 -1.526557e-16
#
# Values of teststatistic and critical values of test:
#
#           test 10pct  5pct  1pct
# r <= 5 |  7.90  7.52  9.24 12.97
# r <= 4 | 15.47 13.75 15.67 20.20
# r <= 3 | 17.36 19.77 22.00 26.81
# r <= 2 | 31.44 25.56 28.14 33.24
# r <= 1 | 52.75 31.66 34.40 39.79
# r = 0  | 57.92 37.45 40.30 46.82
#
# Eigenvectors, normalised to first column:
# (These are the cointegration relations)
#
#                 crude_brent.l1   sugar_us.l1     gold.l1    silver.l1
# crude_brent.l1       1.0000000    1.00000000  1.00000000   1.00000000
# sugar_us.l1       1662.0411948  -71.21405459 60.17328520 -10.00657821
# gold.l1             -0.2118140   -0.03714532  0.02496939   0.10082497
# silver.l1           24.5167069  -16.68586272 -3.77412375  -8.84690122
# wheat_us_hrw.l1     -6.4314141   -2.80153856 -0.06029133  -0.06902232
# soybeans.l1          0.9101583    2.51687157 -0.11133628  -0.38424046
# constant           -95.9117813 -103.67013985  5.93969138  86.24923917
#                 wheat_us_hrw.l1  soybeans.l1     constant
# crude_brent.l1        1.0000000   1.00000000   1.00000000
# sugar_us.l1        -220.0183245 -14.67469380  61.90250747
# gold.l1               0.1218462  -0.06771101  -0.33359961
```

Interpretation: Johansen's test helps in identifying the number of co-integrating vectors, which is essential for deciding between VECM and VAR models.

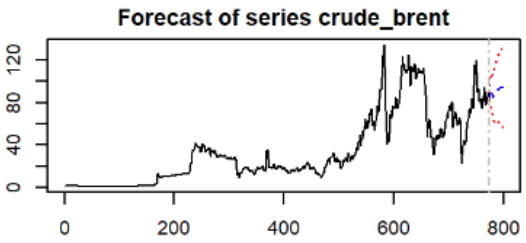
Estimate and Forecast with VECM

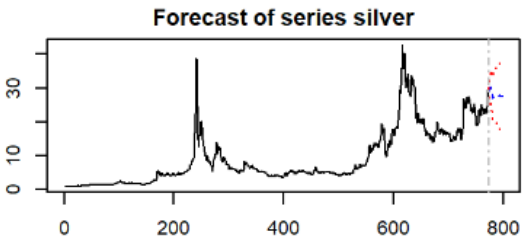
```r
# Determine the number of co-integrating relationships (r) based on the test
r <- 2  # Replace with the actual number from the test results

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)
```

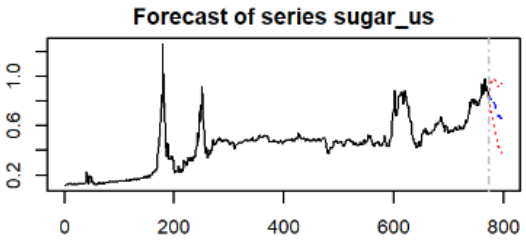
```r
} else {
  # If no co-integration exists, proceed with Unrestricted VAR Analysis
  var_model <- VAR(commodity_data, p = lag_length, type = "const")

  # 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)
}
```

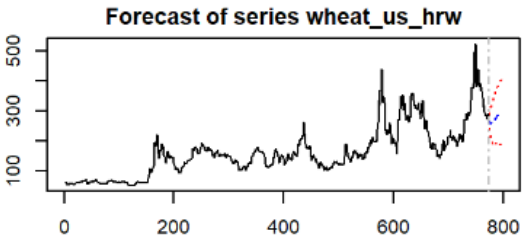
Interpretation: The VECM approach captures both short-term and long-term dynamics, while the VAR approach focuses on interdependencies among variables. Forecasting provides future trends based on the selected model.

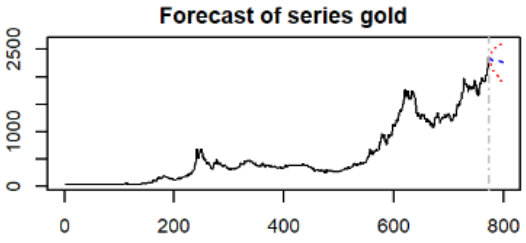
```
##                  crude_brent.d    sugar_us.d        gold.d      silver.d
## ect1              2.276104e-03  7.179337e-06   0.015055098  1.575744e-03
## ect2              1.986914e+00 -4.321011e-02  -0.907642645 -1.205711e-01
## crude_brent.dl1   2.833570e-01  2.078668e-04  -0.244455497 -7.598070e-03
## sugar_us.dl1     -4.781836e+00  1.811301e-01  17.296649715  2.387263e+00
## gold.dl1         -6.667878e-04 -1.686852e-05   0.212476331 -3.018627e-03
## silver.dl1       -5.342232e-02  3.652562e-03   1.353914253  3.856808e-01
## wheat_us_hrw.dl1  2.076949e-02  1.850560e-04   0.184499251  6.226225e-03
## soybeans.dl1      6.310291e-03  6.000438e-05   0.017999544  2.218970e-05
## crude_brent.dl2  -1.198311e-01  5.232410e-04   0.307640352  9.639662e-03
## sugar_us.dl2      2.463380e+00 -2.821421e-02   1.667880054 -2.011341e+00
## gold.dl2         -8.205605e-03  8.095397e-05  -0.043613430  5.907728e-04
## silver.dl2        2.128201e-01 -3.403922e-03  -1.832704366 -2.403710e-01
## wheat_us_hrw.dl2  4.722683e-02 -1.233883e-04  -0.180871686  6.962152e-03
## soybeans.dl2      6.756600e-03  3.230871e-05   0.066356759 -1.030332e-03
## crude_brent.dl3  -7.720171e-02 -3.343312e-04  -0.486162616 -1.981703e-02
## sugar_us.dl3     -5.352107e+00  1.691311e-01  -7.031266782 -2.929916e-01
## gold.dl3          8.503357e-03  3.277013e-05   0.084860363  1.224192e-03
## silver.dl3       -7.907566e-02  1.346325e-03  -1.750722438 -4.237639e-02
## wheat_us_hrw.dl3  2.445242e-02 -3.192438e-04   0.460869308  1.396215e-02
## soybeans.dl3     -7.724794e-03  5.364251e-05  -0.195177201 -5.335368e-03
## crude_brent.dl4  -7.793757e-02 -5.080951e-04  -0.453000787 -5.758950e-03
## sugar_us.dl4     -4.735158e-01 -6.340283e-02  -1.656614455 -3.933908e-01
## gold.dl4          1.583654e-02 -1.174445e-04  -0.037429279  1.552373e-03
## silver.dl4       -7.000784e-02  7.362215e-03   2.333952112  1.538093e-02
## wheat_us_hrw.dl4  1.740753e-02  6.221185e-06  -0.016383770  6.053955e-03
## soybeans.dl4      9.214911e-05 -9.148843e-05   0.009760085 -3.391807e-03
## crude_brent.dl5  -6.162562e-03  2.544312e-04   0.124051150 -1.945852e-02
## sugar_us.dl5      5.801956e-01  1.538242e-01  28.074013007  3.449596e-01
## gold.dl5          8.327369e-04 -5.307964e-05   0.132173001  4.591323e-03
## silver.dl5        4.954381e-02  1.661651e-03   0.115354579 -6.725733e-02
## wheat_us_hrw.dl5  4.903291e-02 -7.437533e-06  -0.093933970  2.750572e-05
## soybeans.dl5      3.803223e-03 -1.356424e-04  -0.077648655 -5.279878e-04
## crude_brent.dl6  -1.328615e-01  1.764409e-04  -0.569862176 -1.719900e-02
## sugar_us.dl6      3.456587e+00 -5.785792e-02  18.231869522  9.065648e-01
```

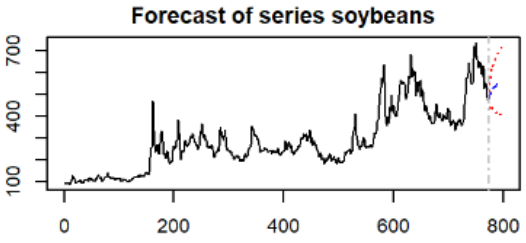

Forecast of series crude_brent
0
40
80
120
0
200
400
600
800


Forecast of series silver
0
10
30
0
200
400
600
800


Forecast of series sugar_us
0.2
0.6
1.0
0
200
400
600
800


Forecast of series wheat_us_hrw
100
300
500
0
200
400
600
800


Forecast of series gold
0
1000
2500
0
200
400
600
800


Forecast of series soybeans
100
400
700
0
200
400
600
800

RECOMMENDATIONS

According to the study of commodity prices using the VAR and VECM models, the following suggestions are made for buyers, companies, and policymakers: The suggestions that came from the VAR and VECM analyses stress how important it is to use insights to make better choices, control risks, and plan strategically. Investors, companies, and policymakers can better manage commodity markets, improve performance, and reach their financial and strategic goals by following these suggestions.

CODES

R

```
# Set working directory and load necessary libraries

setwd('D:\\Assignments_SCMA632')  # Set the working directory to the location of your files

getwd()  # Verify the current working directory


# Install necessary packages if they are not already installed

if (!require(readxl)) install.packages("readxl")

if (!require(dplyr)) install.packages("dplyr")

if (!require(janitor)) install.packages("janitor")

if (!require(urca)) install.packages("urca")

if (!require(vars)) install.packages("vars")


# Load necessary libraries

library(readxl)  # For reading Excel files

library(dplyr)  # For data manipulation

library(janitor)  # For cleaning column names

library(urca)  # For unit root and cointegration tests

library(vars)  # For VAR and VECM modeling


# Load the dataset and sheet

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)  # Check the structure of the dataframe

# Select specific columns (Date and selected commodities)
commodity <- df[,c(1,3,47,70,72,38,25)] %>%
  clean_names()  # Clean the column names for easier manipulation

str(commodity)  # Check the structure of the cleaned dataframe

# 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:", 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
cat("\nSelected lag length:", lag_length, "\n")

# Perform Johansen's Co-Integration Test
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
r <- 2  # 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  # Display the forecast results
```