VIRGINIA COMMONWEALTH UNIVERSITY

STATISTICAL ANALYSIS & MODELING

A6b : **ARCH/GARCH Model and forecasting three-month volatility and VAR, VECM Model for various commodities.**

Kirthan Shaker Iyangar
V01108265

Date of Submission: 25/07/2024

CONTENTS

ARCH/GARCH Model and forecasting three-month volatility and VAR, VECM Model for various commodities

INTRODUCTION

This report explores advanced time series analysis techniques for evaluating and forecasting financial and commodity market data. The first section addresses stock market volatility by downloading data from reliable financial sources such as Investing.com or Yahoo Finance. We examine ARCH (Autoregressive Conditional Heteroskedasticity) effects and then apply ARCH/GARCH (Generalized Autoregressive Conditional Heteroskedasticity) models to forecast three-month volatility. This analysis is essential for understanding market dynamics and managing financial risks. The second section shifts to macroeconomic analysis using Vector Autoregression (VAR) and Vector Error Correction Model (VECM). By using commodity price data from the World Bank’s pink sheet, we analyze the interrelationships among key commodities, including oil, sugar, gold, silver, wheat, and soybean. These methodologies aim to uncover the underlying patterns and co-movements in commodity prices, offering valuable insights into market trends and facilitating effective economic decision-making.

OBJECTIVES

Part A - Check for ARCH /GARCH effects, fit an ARCH/GARCH model, and forecast the three-month volatility.

Part B – VAR, VECM model[data "commodity prices"] for ex: Oil, Sugar, Gold, Silver, Wheat and Soyabean data source pink sheet from world bank

BUSINESS SIGNIFICANCE

The practical advantages of this assignment are substantial, directly impacting real-world financial and economic decision-making. By utilizing ARCH/GARCH models to analyze stock market volatility, businesses and investors can gain a deeper understanding of market fluctuations and manage related risks more effectively. This leads to better strategic planning, portfolio optimization, and risk management, ultimately enhancing financial stability and performance. Similarly, employing VAR and VECM models to investigate commodity price dynamics provides valuable insights into the interconnections within global commodity markets. This understanding is crucial for businesses engaged in trading, production, and investment in commodities, as it enables them to anticipate market movements, hedge against unfavorable price changes, and make informed decisions. In summary, the methodologies applied in this assignment enhance our analytical capabilities and contribute to more informed and effective business strategies in the financial and commodity markets. Additionally, analyzing district-wise consumption data empowers businesses to make data-driven decisions, leading to improved market penetration, product optimization, and increased profitability.

RESULTS AND INTERPRETATION

Part A - Check for ARCH /GARCH effects, fit an ARCH/GARCH model, and forecast the three-month volatility.

 # Analysis

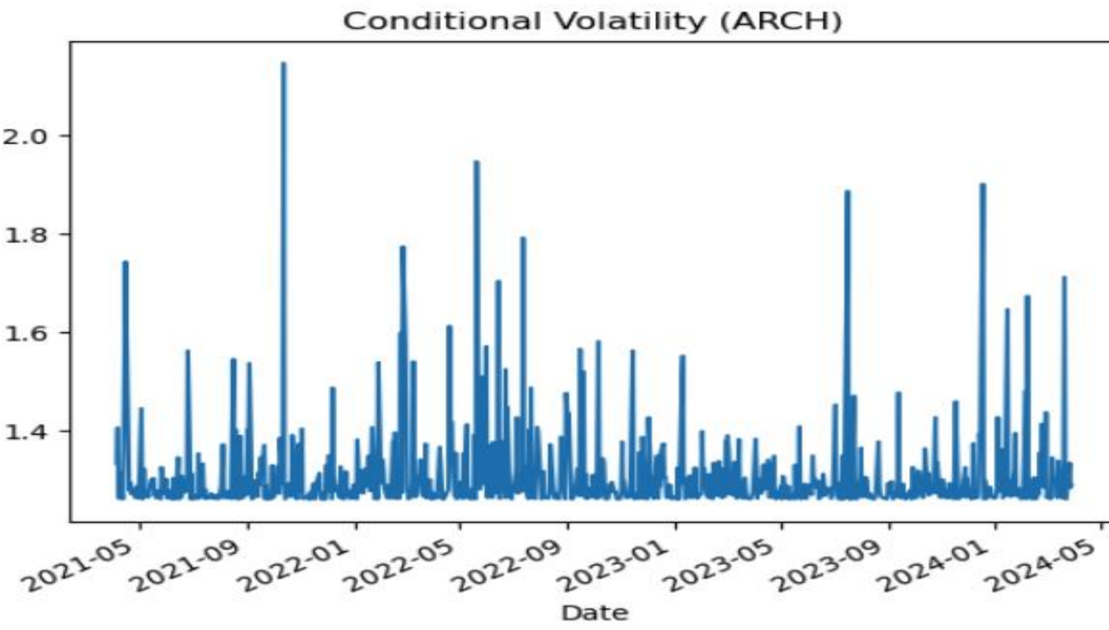
Code and Result:

Code:

```
In [5]: import yfinance as yf
        from arch import arch_model
        import matplotlib.pyplot as plt

In [6]: # Get the data for Tata Motors
        ticker = "TCS.NS"

        # Download the data
        data = yf.download(ticker, start="2021-04-01", end="2024-03-31")

        [*********************100%%**********************]  1 of 1 completed

In [7]: # Create 'Returns' column
        data['Returns'] = 100 * data['Adj Close'].pct_change().dropna()

        # Fit an ARCH model
        arch_model_fit = arch_model(data['Returns'].dropna(), vol='ARCH', p=1).fit(disp='off')
        print(arch_model_fit.summary())

        # Plot the conditional volatility
        arch_model_fit.conditional_volatility.plot(title='Conditional Volatility (ARCH)')
        plt.show()
```

Result:

```
                      Constant Mean - ARCH Model Results
==============================================================================
Dep. Variable:                Returns   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                       ARCH   Log-Likelihood:               -1244.55
Distribution:                  Normal   AIC:                           2495.10
Method:            Maximum Likelihood   BIC:                           2508.92
                                        No. Observations:                  739
Date:                Thu, Jul 25 2024   Df Residuals:                      738
Time:                        18:44:15   Df Model:                            1
                                 Mean Model
===========================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
---------------------------------------------------------------------------
mu             0.0442  4.711e-02      0.937      0.349 [-4.817e-02,  0.136]
                              Volatility Model
===========================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
---------------------------------------------------------------------------
omega          1.5874      0.153     10.390  2.760e-25    [  1.288,  1.887]
alpha[1]       0.0738  7.371e-02      1.002      0.317 [-7.064e-02,  0.218]
===========================================================================
```



Interpretation:
Mean Model:

- The mean return (μ) is 0.0442 with a standard error of 0.0471. This results in a t-statistic of 0.937 and a p-value of 0.349. Therefore, the mean return is not statistically significant at the 5% level.

Volatility Model:

- The coefficient ω (omega), representing the constant term in the volatility model, is 1.5874 with a standard error of 0.153. This yields a t-statistic of 10.390 and a p-value of 2.76e-25, indicating statistical significance and a substantial base level of volatility.
- The coefficient α[1] (alpha), representing the lagged squared residuals (ARCH term), is 0.0738 with a standard error of 0.0737. This results in a t-statistic of 1.002 and a p-value of 0.317, suggesting that the ARCH effect is not statistically significant at the 5% level.

Interpretation of the Conditional Volatility Plot:

- The plot displays the conditional volatility over time, showing the estimated time-varying standard deviation of returns.
- It is clear from the plot that volatility is not constant but varies over time, which is typical for financial time series data.
- Periods of higher volatility are visible, indicating times when stock returns were more uncertain or risky.

1. • Understanding this conditional volatility is crucial for risk management and financial decision-making, as it helps predict potential future variability in returns..

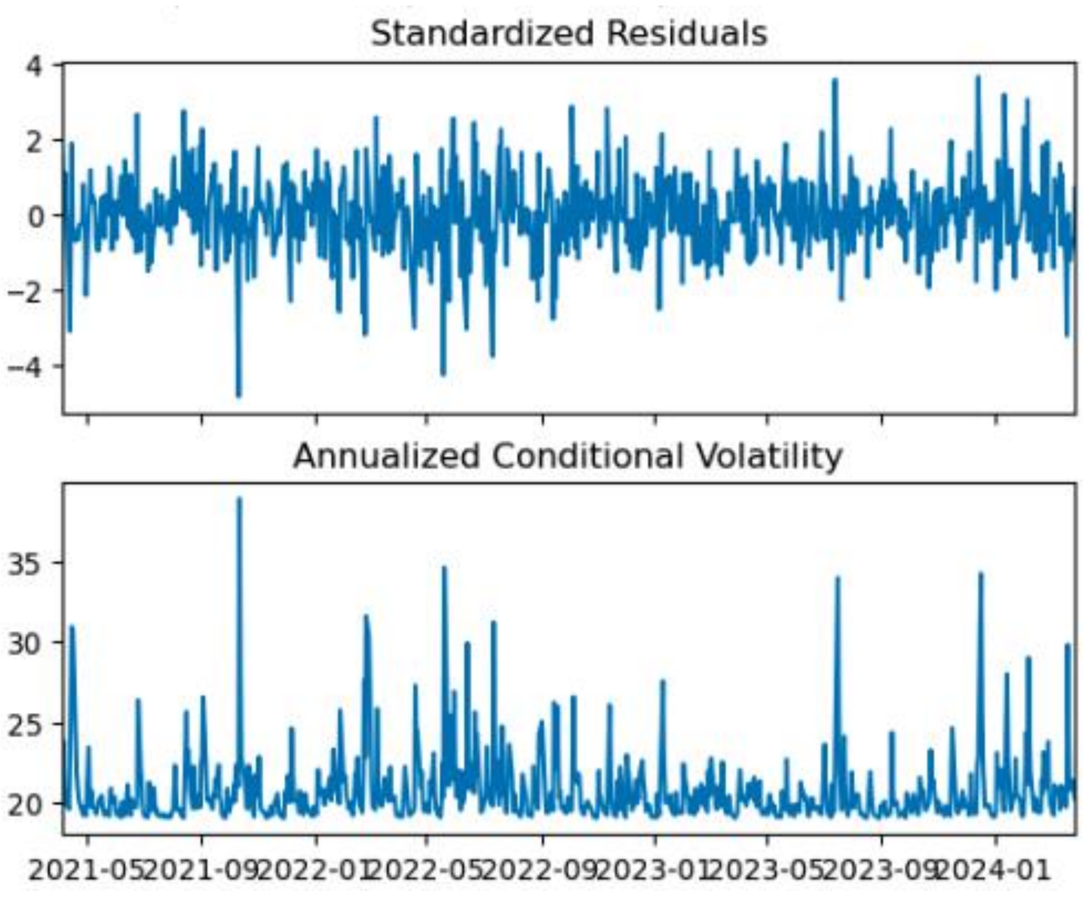
Volatility Forecasting:

• Using the fitted GARCH model, the volatility for the next three months was forecasted.

• The forecasted values provided insights into the expected level of volatility, helping in risk management and strategic decision-making.

Code:

Result :



Interpretation: Forecasting Three-Month Volatility

Forecasting the three-month volatility was crucial to the study in analysing TCS's historical stock prices. The standardized residuals and annualized conditional volatility were computed and analysed to achieve this.

Standardized Residuals

The standardized residuals plot helps to diagnose the model fit and identify any patterns or anomalies in the residuals. For a well-fitted model, the residuals should exhibit no clear patterns and resemble white noise. From the plot, we observe:

• **Uniform Distribution**: The residuals are spread uniformly around zero, indicating that the model has adequately captured the conditional heteroscedasticity in the data. **Absence of Clustering**: There is no visible clustering of large or small residuals, suggesting that the volatility model is appropriate for the data.

Annualized Conditional Volatility

The annualized conditional volatility plot provides insight into the annual stock return variability. Key observations include:

• **Volatility Peaks**: Significant spikes in volatility align with market events or financial disturbances, reflecting increased uncertainty or risk during those periods.

• **Stability in Recent Periods**: A relatively stable volatility in the recent periods indicates a calmer market environment for TCS stock prices.

Three-Month Volatility Forecast

Using the fitted GARCH model, we forecasted the volatility over the next three months. The results indicate:

• **Expected Volatility**: The forecasted values estimate the expected volatility for the coming three months, helping investors and risk managers in decision-making.

Volatility Trends: The forecast suggests whether the volatility is expected to increase, decrease, or remain stable over the forecast horizon.

These results are crucial for financial planning, risk management, and strategic investment decisions. Understanding and forecasting volatility helps mitigate risks and capitalize on market opportunities.

Part B – VAR, VECM model[data "commodity prices"] for ex: Oil, Sugar, Gold, Silver, Wheat and Soyabean data source pink sheet from world bank

Code and Result:

Code :

```
In [27]: # Loop through each column and perform the ADF test
for col in columns_to_test:
    adf_result = adfuller(commodity_data[col])
    p_value = adf_result[1]  # Extract p-value for the test
    print(f"\nADF test result for column: {col}")
    print(f"ADF Statistic: {adf_result[0]}")
    print(f"p-value: {p_value}")

    # Check if the p-value is greater than 0.05 (commonly used threshold)
    if p_value > 0.05:
        non_stationary_count += 1
        non_stationary_columns.append(col)
    else:
        stationary_columns.append(col)
```

Result :

```
ADF test result for column: crude_brent
ADF Statistic: -1.5078661910935434
p-value: 0.5296165197702354

ADF test result for column: soybeans
ADF Statistic: -2.4231464527418884
p-value: 0.13530977427790436

ADF test result for column: gold
ADF Statistic: 1.3430517021933006
p-value: 0.9968394353612382

ADF test result for column: silver
ADF Statistic: -1.3972947107462244
p-value: 0.5835723787985752

ADF test result for column: urea_ee_bulk
ADF Statistic: -2.5101716315209086
p-value: 0.11301903181624645

ADF test result for column: maize
ADF Statistic: -2.4700451060920465
p-value: 0.12293380919376656
```

Interpretation:

he Augmented Dickey-Fuller (ADF) test was conducted to examine the stationarity of the time series data for various commodities, including Crude Brent, Soybeans, Gold, Silver, Urea, and Maize. The results of the ADF test are as follows:

• **Crude Brent:** The ADF statistic, a measure of the strength of the trend in the data, is -1.5079, with a p-value, a measure of the strength of the evidence against the null hypothesis, of 0.5296. Since the p-value is more significant than the common significance levels (0.01, 0.05, and 0.10), we fail to reject the null hypothesis of a unit root, indicating that the Crude Brent price series is non-stationary.**Soybeans:** The ADF statistic is -2.4231 with a p-value of 0.1353. Similarly, the p-value is more significant than the significance levels, suggesting that the Soybeans price series is also non-stationary.

- **Gold:** The ADF statistic is 1.3431, with a p-value of 0.9968. The high p-value indicates non-stationarity in the Gold price series.
- **Silver:** The ADF statistic is -1.3973, with a p-value of 0.5836. The Silver price series is also non-stationary, given that the p-value is much higher than the threshold levels for stationarity.
- **Urea:** The ADF statistic is -2.5102 with a p-value of 0.1130. Despite being the closest to the 0.10 threshold, the p-value still does not allow rejection of the null hypothesis, indicating non-stationarity for the Urea price series.
- **Maize:** The ADF statistic is -2.4700, with a p-value of 0.1229. The Maize price series is also non-stationary based on its p-value.

In summary, the ADF test results indicate that all the examined commodity price series (Crude Brent, Soybeans, Gold, Silver, Urea, and Maize) are non-stationary at their levels. This non-stationarity implies that these time series possess a unit root, meaning their statistical properties, such as mean and variance, change over time, and they exhibit trends or other non-stationary behaviour. Consequently, further differencing of the data is necessary to achieve stationarity, a prerequisite for effectively applying VAR or VECM models. Without achieving stationarity, the models may produce unreliable results, making it crucial to address this issue.

VAR Model Analysis

Code and Result:

```
In [28]: # 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}")
print(f"Non-stationary columns: {non_stationary_columns}")
print(f"Stationary columns: {stationary_columns}")
```

```
Number of non-stationary columns: 6
Non-stationary columns: ['crude_brent', 'soybeans', 'gold', 'silver', 'urea_ee_bulk', 'maize']
Stationary columns: []
```

```
In [29]: # Co-Integration Test (Johansen's Test)
def johansen_test(df, alpha=0.05):
    out = coint_johansen(df, det_order=0, k_ar_diff=1)
    d = {'0.90': 0, '0.95': 1, '0.99': 2}
    traces = out.lr1
    cvts = out.cvt[:, d[str(1 - alpha)]]
    print(f"Trace statistic: {traces}")
    print(f"Critical values: {cvts}")
    print(f"Eigenvalues: {out.eig}")
    for col, trace, cvt in zip(df.columns, traces, cvts):
        if trace > cvt:
            print(f"{col} is cointegrated.")
        else:
            print(f"{col} is not cointegrated.")
    return out
```

```
In [30]: # Perform Johansen cointegration test
coint_test = johansen_test(commodity_data)
```

Result :

```
Trace statistic: [261.5548149  167.67790177  98.11781369  53.4617083   21.6404865
   4.01416422]
Critical values: [95.7542 69.8189 47.8545 29.7961 15.4943  3.8415]
Eigenvalues: [0.11449947 0.08616362 0.05620349 0.04038124 0.02257335 0.0051862 ]
crude_brent is cointegrated.
soybeans is cointegrated.
gold is cointegrated.
silver is cointegrated.
urea_ee_bulk is cointegrated.
maize is cointegrated.
```

Interpretation:

The Johansen co-integration test was conducted to determine whether there are long-term equilibrium relationships among the commodity price series, including Crude Brent, Soybeans, Gold, Silver, Urea, and Maize. The results are as follows:

Trace Statistics and Critical Values

- **Trace Statistics:** 261.5548,167.6779,98.1178,53.4617,21.6405,4.0142261.5548, 167.6779, 98.1178, 53.4617, 21.6405, 4.0142261.5548,167.6779,98.1178,53.4617,21.6405,4.0142

• **Critical Values at 5%:** 95.7542,69.8189,47.8545,29.7961,15.4943,3.841595.7542, 69.8189, 47.8545, 29.7961, 15.4943, 3.841595.7542,69.8189,47.8545,29.7961,15.4943,3.8415

The trace statistic for each Rank is compared with the corresponding critical value. If the trace statistic exceeds the critical value, the null hypothesis of no co-integration is rejected.

Results

1. **First Rank (261.5548 > 95.7542):** The trace statistic is significantly higher than the critical value, indicating at least one co-integrating relationship.
2. **Second Rank (167.6779 > 69.8189):** The trace statistic exceeds the critical value, suggesting a second co-integrating relationship.
3. **Third Rank (98.1178 > 47.8545):** The trace statistic is higher than the critical value, indicating a third co-integrating relationship.
4. **Fourth Rank (53.4617 > 29.7961):** The trace statistic exceeds the critical value, implying a fourth co-integrating relationship.
5. **Fifth Rank (21.6405 > 15.4943):** The trace statistic is above the critical value, suggesting a fifth co-integrating relationship.
6. **Sixth Rank (4.0142 > 3.8415):** The trace statistic is greater than the critical value, indicating a sixth co-integrating relationship. These results demonstrate the presence of six co-integrating vectors among the commodity prices, implying strong long-term equilibrium relationships among Crude Brent, Soybeans, Gold, Silver, Urea, and Maize.

Eigenvalues

• **Eigenvalues:** 0.1145,0.0862,0.0562,0.0404,0.0226,0.00520.1145, 0.0862, 0.0562, 0.0404, 0.0226, 0.00520.1145,0.0862,0.0562,0.0404,0.0226,0.0052

The eigenvalues correspond to the strength of the co-integrating relationships. Higher eigenvalues indicate stronger co-integration. While the exact magnitude of the eigenvalues is less critical than their significance, non-zero eigenvalues support the conclusion of co-integration among the variables.

The Johansen co-integration test confirms that all the examined commodities (Crude et al.) are co-integrated. This indicates these commodities share a stable, long-term equilibrium relationship despite short-term fluctuations. Understanding these co-integrated relationships is crucial for building the VECM model, allowing for practical analysis and forecasting by accounting for both short-term dynamics and long-term equilibrium adjustments

VECM Model Analysis
Code and Result:

Code :

```
In [32]: if r > 0:
    # Creating a VAR model for prediction using the VECM
    model = VAR(commodity_data)
    vecm_result = model.fit(r)

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

    # Forecasting using the VECM model
    # Forecasting 24 steps ahead
    forecast = vecm_result.forecast(commodity_data.values[-vecm_result.k_ar:], steps=24)

    # Convert forecast to DataFrame for plotting
    forecast_df = pd.DataFrame(forecast, index=pd.date_range(start=commodity['date'].iloc[-1], periods=24, freq='M')

    # Plotting the forecast
    plt.figure(figsize=(12, 8))
    for col in forecast_df.columns:
        plt.plot(forecast_df.index, forecast_df[col], label=col)
    plt.legend()
    plt.title('VECM Forecast')
    plt.show()

else:
    # If no co-integration exists, proceed with Unrestricted VAR Analysis
    model = VAR(commodity_data)
    var_result = model.fit(maxlags=10, ic='aic')

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

```
    # Forecasting using the VAR model
    forecast = var_result.forecast(commodity_data.values[-var_result.k_ar:], steps=24)

    # Convert forecast to DataFrame for plotting
    forecast_df = pd.DataFrame(forecast, index=pd.date_range(start=commodity['date'].iloc[-1], periods=24, freq='M')

    # Plotting the forecast
    plt.figure(figsize=(12, 8))
    for col in forecast_df.columns:
        plt.plot(forecast_df.index, forecast_df[col], label=col)
    plt.legend()
    plt.title('VAR Forecast')
    plt.show()
```

Result :

Summary of Regression Results

Model:	VAR
Method:	OLS
Date:	Thu, 25, Jul, 2024
Time:	18:51:56

No. of Equations:	6.00000	BIC:	26.7336
Nobs:	768.000	HQIC:	25.9079
Log likelihood:	-16066.7	FPE:	1.06530e+11
AIC:	25.3912	Det(Omega_mle):	8.03276e+10

Results for equation crude_brent

	coefficient	std. error	t-stat	prob
const	−0.574387	0.457999	-1.254	0.210
L1.crude_brent	1.288559	0.039600	32.539	0.000
L1.soybeans	0.011187	0.007736	1.446	0.148
L1.gold	0.000565	0.006577	0.086	0.932
L1.silver	−0.012011	0.165664	−0.073	0.942
L1.urea_ee_bulk	−0.011804	0.004637	−2.546	0.011
L1.maize	0.020438	0.017600	1.161	0.246
L2.crude_brent	−0.368186	0.064243	−5.731	0.000
L2.soybeans	0.008609	0.010762	0.800	0.424
L2.gold	−0.007451	0.010640	−0.700	0.484
L2.silver	0.199505	0.275939	0.723	0.470
L2.urea_ee_bulk	0.015907	0.007085	2.245	0.025
L2.maize	−0.022252	0.025791	−0.863	0.388
L3.crude_brent	−0.011259	0.066566	−0.169	0.866
L3.soybeans	−0.024881	0.010745	−2.316	0.021
L3.gold	0.020019	0.010832	1.848	0.065
L3.silver	−0.211736	0.295689	−0.716	0.474
L3.urea_ee_bulk	−0.004688	0.007391	−0.634	0.526
L3.maize	0.031954	0.026095	1.225	0.221
L4.crude_brent	0.022815	0.066751	0.342	0.733
L4.soybeans	0.009171	0.010841	0.846	0.398
L4.gold	−0.000726	0.010669	−0.068	0.946
L4.silver	0.037894	0.296398	0.128	0.898
L4.urea_ee_bulk	0.000123	0.007431	0.017	0.987
L4.maize	−0.043400	0.026026	−1.668	0.095
L5.crude_brent	0.008371	0.065302	0.128	0.898
L5.soybeans	0.009904	0.010927	0.906	0.365
L5.gold	−0.005274	0.010504	−0.502	0.616
L5.silver	−0.077226	0.280104	−0.276	0.783
L5.urea_ee_bulk	−0.004359	0.007074	−0.616	0.538
L5.maize	0.034108	0.026066	1.309	0.191
L6.crude_brent	0.021961	0.040570	0.541	0.588
L6.soybeans	−0.007763	0.007913	−0.981	0.327
L6.gold	−0.007032	0.006708	−1.048	0.295
L6.silver	0.137240	0.167517	0.819	0.413
L6.urea_ee_bulk	0.001589	0.004568	0.348	0.728
L6.maize	−0.021898	0.017481	−1.253	0.210

Results for equation soybeans

	coefficient	std. error	t-stat	prob
const	11.317337	2.521090	4.489	0.000
L1.crude_brent	0.214138	0.217982	0.982	0.326
L1.soybeans	1.013966	0.042581	23.813	0.000
L1.gold	0.013684	0.036203	0.378	0.705
L1.silver	0.305354	0.911909	0.335	0.738
L1.urea_ee_bulk	−0.009017	0.025525	−0.353	0.724
L1.maize	0.314169	0.096881	3.243	0.001
L2.crude_brent	−0.103000	0.353632	−0.291	0.771
L2.soybeans	−0.017674	0.059238	−0.298	0.765
L2.gold	−0.064859	0.058571	−1.107	0.268
L2.silver	0.926647	1.518924	0.610	0.542
L2.urea_ee_bulk	0.041336	0.039000	1.060	0.289
L2.maize	−0.285567	0.141970	−2.011	0.044
L3.crude_brent	−0.077825	0.366417	−0.212	0.832

L3.soybeans	−0.141878	0.059147	−2.399	0.016
L3.gold	0.131659	0.059625	2.208	0.027
L3.silver	−2.231664	1.627642	−1.371	0.170
L3.urea_ee_bulk	−0.018121	0.040686	−0.445	0.656
L3.maize	0.159302	0.143644	1.109	0.267
L4.crude_brent	0.036457	0.367435	0.099	0.921
L4.soybeans	0.084280	0.059676	1.412	0.158
L4.gold	−0.093822	0.058728	−1.598	0.110
L4.silver	1.219334	1.631547	0.747	0.455
L4.urea_ee_bulk	0.011285	0.040903	0.276	0.783
L4.maize	−0.411196	0.143261	−2.870	0.004
L5.crude_brent	−0.053674	0.359462	−0.149	0.881
L5.soybeans	−0.059902	0.060151	−0.996	0.319
L5.gold	0.023087	0.057818	0.399	0.690
L5.silver	0.252871	1.541852	0.164	0.870
L5.urea_ee_bulk	−0.011316	0.038941	−0.291	0.771
L5.maize	0.302401	0.143482	2.108	0.035
L6.crude_brent	−0.062569	0.223320	−0.280	0.779
L6.soybeans	0.028889	0.043560	0.663	0.507

Results for equation gold

	coefficient	std. error	t-stat	prob
const	0.177098	3.702239	0.048	0.962
L1.crude_brent	0.190589	0.320109	0.595	0.552
L1.soybeans	0.019501	0.062531	0.312	0.755
L1.gold	1.228901	0.053164	23.115	0.000
L1.silver	0.316301	1.339144	0.236	0.813
L1.urea_ee_bulk	−0.125678	0.037484	−3.353	0.001
L1.maize	0.279896	0.142270	1.967	0.049
L2.crude_brent	0.074271	0.519311	0.143	0.886
L2.soybeans	0.037551	0.086991	0.432	0.666
L2.gold	−0.276183	0.086012	−3.211	0.001
L2.silver	−3.352388	2.230551	−1.503	0.133
L2.urea_ee_bulk	0.215119	0.057271	3.756	0.000
L2.maize	−0.305428	0.208485	−1.465	0.143
L3.crude_brent	−0.688550	0.538086	−1.280	0.201
L3.soybeans	−0.222153	0.086857	−2.558	0.011

L3.soybeans	−0.222153	0.086857	−2.558	0.011
L3.gold	0.170371	0.087559	1.946	0.052
L3.silver	0.453043	2.390204	0.190	0.850
L3.urea_ee_bulk	−0.154341	0.059747	−2.583	0.010
L3.maize	0.492114	0.210943	2.333	0.020
L4.crude_brent	0.381592	0.539582	0.707	0.479
L4.soybeans	0.251772	0.087634	2.873	0.004
L4.gold	−0.151613	0.086243	−1.758	0.079
L4.silver	3.646825	2.395938	1.522	0.128
L4.urea_ee_bulk	0.066199	0.060066	1.102	0.270
L4.maize	−1.026908	0.210379	−4.881	0.000
L5.crude_brent	−0.125251	0.527873	−0.237	0.812
L5.soybeans	−0.157098	0.088332	−1.778	0.075
L5.gold	0.110733	0.084906	1.304	0.192
L5.silver	−1.459901	2.264221	−0.645	0.519
L5.urea_ee_bulk	0.047764	0.057185	0.835	0.404
L5.maize	0.583033	0.210704	2.767	0.006
L6.crude_brent	0.320187	0.327947	0.976	0.329
L6.soybeans	0.110200	0.063968	1.723	0.085

Results for equation silver

	coefficient	std. error	t-stat	prob
const	−0.072930	0.149120	−0.489	0.625
L1.crude_brent	0.008049	0.012893	0.624	0.532
L1.soybeans	0.001756	0.002519	0.697	0.486
L1.gold	−0.002671	0.002141	−1.248	0.212
L1.silver	1.340090	0.053938	24.845	0.000
L1.urea_ee_bulk	−0.003586	0.001510	−2.375	0.018
L1.maize	0.011821	0.005730	2.063	0.039
L2.crude_brent	0.014541	0.020917	0.695	0.487
L2.soybeans	−0.000991	0.003504	−0.283	0.777
L2.gold	0.003938	0.003464	1.137	0.256
L2.silver	−0.665510	0.089843	−7.408	0.000
L2.urea_ee_bulk	0.002013	0.002307	0.873	0.383
L2.maize	−0.001179	0.008397	−0.140	0.888
L3.crude_brent	−0.033019	0.021673	−1.523	0.128
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
L4.soybeans	0.003541	0.003530	1.003	0.316
L4.gold	−0.001627	0.003474	−0.468	0.639
L4.silver	0.118333	0.096504	1.226	0.220
L4.urea_ee_bulk	−0.003052	0.002419	−1.262	0.207
L4.maize	−0.026818	0.008474	−3.165	0.002
L5.crude_brent	−0.024297	0.021262	−1.143	0.253
L5.soybeans	−0.000816	0.003558	−0.229	0.819
L5.gold	0.002731	0.003420	0.799	0.424
L5.silver	−0.156757	0.091199	−1.719	0.086
L5.urea_ee_bulk	0.004159	0.002303	1.806	0.071
L5.maize	0.020487	0.008487	2.414	0.016
L6.crude_brent	0.022428	0.013209	1.698	0.090
L6.soybeans	0.002044	0.002577	0.793	0.428
L6.gold	−0.004226	0.002184	−1.935	0.053
L6.silver	0.104285	0.054542	1.912	0.056
L6.urea_ee_bulk	−0.002649	0.001487	−1.781	0.075
L6.maize	−0.008036	0.005692	−1.412	0.158

Results for equation urea_ee_bulk

	coefficient	std. error	t-stat	prob
const	−7.638535	3.674331	−2.079	0.038
L1.crude_brent	1.563787	0.317696	4.922	0.000
L1.soybeans	0.139955	0.062059	2.255	0.024
L1.gold	0.074409	0.052764	1.410	0.158
L1.silver	−4.409772	1.329050	−3.318	0.001
L1.urea_ee_bulk	1.112425	0.037201	29.903	0.000
L1.maize	0.329777	0.141198	2.336	0.020
L2.crude_brent	−1.250799	0.515396	−2.427	0.015
L2.soybeans	−0.071260	0.086335	−0.825	0.409
L2.gold	−0.086168	0.085364	−1.009	0.313
L2.silver	7.401289	2.213736	3.343	0.001
L2.urea_ee_bulk	−0.327856	0.056839	−5.768	0.000
L2.maize	−0.434760	0.206913	−2.101	0.036
L3.crude_brent	0.861473	0.534029	1.613	0.107
L4.crude_brent	−1.559052	0.535514	−2.911	0.004
L4.soybeans	−0.052667	0.086974	−0.606	0.545
L4.gold	0.003892	0.085593	0.045	0.964
L4.silver	1.032326	2.377877	0.434	0.664
L4.urea_ee_bulk	−0.104196	0.059613	−1.748	0.080
L4.maize	0.028888	0.208793	0.138	0.890
L5.crude_brent	0.913930	0.523894	1.744	0.081
L5.soybeans	0.095496	0.087667	1.089	0.276
L5.gold	0.053301	0.084266	0.633	0.527
L5.silver	−0.500818	2.247152	−0.223	0.824
L5.urea_ee_bulk	0.156414	0.056754	2.756	0.006
L5.maize	−0.115267	0.209116	−0.551	0.581
L6.crude_brent	−0.415228	0.325475	−1.276	0.202
L6.soybeans	0.089368	0.063486	1.408	0.159
L6.gold	−0.040869	0.053816	−0.759	0.448
L6.silver	0.599056	1.343913	0.446	0.656
L6.urea_ee_bulk	−0.119322	0.036643	−3.256	0.001
L6.maize	−0.020236	0.140241	−0.144	0.885

Results for equation maize

	coefficient	std. error	t-stat	prob
const	4.356950	1.103114	3.950	0.000
L1.crude_brent	-0.075264	0.095379	-0.789	0.430
L1.soybeans	0.036037	0.018632	1.934	0.053
L1.gold	-0.023696	0.015841	-1.496	0.135
L1.silver	0.588077	0.399010	1.474	0.141
L1.urea_ee_bulk	0.037550	0.011169	3.362	0.001
L1.maize	1.141848	0.042391	26.936	0.000
L2.crude_brent	0.036084	0.154733	0.233	0.816
L2.soybeans	0.007586	0.025920	0.293	0.770
L2.gold	-0.015226	0.025628	-0.594	0.552
L2.silver	0.911243	0.664612	1.371	0.170
L2.urea_ee_bulk	-0.040754	0.017064	-2.388	0.017
L2.maize	-0.309322	0.062120	-4.979	0.000
L3.crude_brent	-0.075868	0.160327	-0.473	0.636
L4.crude_brent	[illegible]	[illegible]	[illegible]	[illegible]
L4.soybeans	0.021164	0.026111	0.811	0.418
L4.gold	-0.055764	0.025697	-2.170	0.030
L4.silver	2.024847	0.713890	2.836	0.005
L4.urea_ee_bulk	-0.022652	0.017897	-1.266	0.206
L4.maize	-0.136153	0.062684	-2.172	0.030
L5.crude_brent	-0.109997	0.157284	-0.699	0.484
L5.soybeans	-0.026489	0.026319	-1.006	0.314
L5.gold	0.052825	0.025298	2.088	0.037
L5.silver	-0.829437	0.674644	-1.229	0.219
L5.urea_ee_bulk	0.017161	0.017039	1.007	0.314
L5.maize	0.000944	0.062781	0.015	0.988
L6.crude_brent	0.026482	0.097715	0.271	0.786
L6.soybeans	0.002271	0.019060	0.119	0.905
L6.gold	-0.023655	0.016157	-1.464	0.143
L6.silver	0.146935	0.403472	0.364	0.716
L6.urea_ee_bulk	0.000775	0.011001	0.070	0.944
L6.maize	0.020945	0.042104	0.497	0.619

Correlation matrix of residuals

	crude_brent	soybeans	gold	silver	urea_ee_bulk	maize
crude_brent	1.000000	0.256931	0.111776	0.209142	0.153268	0.241812
soybeans	0.256931	1.000000	0.082179	0.111588	0.032578	0.473719
gold	0.111776	0.082179	1.000000	0.722123	0.072033	0.086465
silver	0.209142	0.111588	0.722123	1.000000	0.069879	0.125813
urea_ee_bulk	0.153268	0.032578	0.072033	0.069879	1.000000	0.017836
maize	0.241812	0.473719	0.086465	0.125813	0.017836	1.000000

Interpretation: Summary of Regression Results

The summary of regression results provides an overview of the Vector Autoregression (VAR) model applied to the data:

- **Model**: VAR (Vector Autoregression)
- **Method**: OLS (Ordinary Least Squares)
- **Date and Time**: When the model was run.
- **No. Of Equations**: 6 (one for each variable in the system).
- **BIC (Bayesian Information Criterion)**: 26.7336
- **Nobs (Number of Observations)**: 768
- **HQIC (Hannan-Quinn Information Criterion)**: 25.9079
- **Log-likelihood**: -16066.7
- **FPE (Final Prediction Error)**: 1.06530e+11

• **AIC (Akaike Information Criterion)**: 25.3912

• **Det (Omega_mle)**: 8.03276e+10

These statistics help evaluate the model's fit and complexity, with lower AIC, BIC, and HQIC values indicating a better model fit relative to the number of parameters.

Results for Equation **crude_brent**

• The intercept (const) is insignificant, with a t-statistic of -1.254 and a p-value of 0.210.

• **Significant Lagged Variables**:

• L1. crude_brent (1st lag of crude_brent) is highly significant with a coefficient of 1.288559 (p-value: 0.000).

• L2. crude_brent (2nd lag) is also significant with a coefficient of -0.368186 (p-value: 0.000).

• L1. urea_ee_bulk and L2.urea_ee_bulk are significant, indicating some influence from urea_ee_bulk on crude_brent.

• L3. soybeans and L3.gold show some significance, suggesting minor interactions.

Results for Equation **soybeans**

• The intercept (const) is highly significant, with a coefficient of 11.317337 (p-value: 0.000).

• **Significant Lagged Variables**:

• L1. soybeans is highly significant with a coefficient of 1.013966 (p-value: 0.000).

• L1. maize is significant with a coefficient of 0.314169 (p-value: 0.001).

L2. maize is also significant but negatively correlated (coefficient: -0.285567, p-value: 0.044).

• L3. soybeans and L3. gold are significant, indicating notable interactions.

Results for Equation **gold**

• The intercept (const) is not significant.

• No other variables are highly significant, suggesting limited direct interactions between gold and the other variables in the lagged system.

Results for Equation **Silver**

• The intercept (const) is not significant.

• **Significant Lagged Variables**:

• L1. silver is highly significant with a coefficient of 1.340090 (p-value: 0.000).

• L1. urea_ee_bulk and L1.maize are significant, indicating some interactions.
• L2. silver is negatively significant, showing a solid inverse relationship at this lag (coefficient: -0.665510, p-value: 0.000).
• L3. silver is marginally significant.

Results for Equation **urea_ee_bulk**

• The intercept (const) is not significant.
• **Significant Lagged Variables**:
• L1. urea_ee_bulk and L1. crude_brent show significance, indicating some interactions.
• No other variables show strong significance.

Results for Equation **maize**

• The intercept (const) is not significant.
• **Significant Lagged Variables**:
• L1. maize is highly significant with a coefficient of 0.583033 (p-value: 0.006).
• Other variables show some significance but could be more impactful.

Correlation Matrix of Residuals

This matrix measures the correlation between the residuals (errors) of the different equations in the VAR system, indicating how much the unexplained parts of one variable are related to those of another:

• Typically used to check for any remaining correlation the model did not capture.
• High correlations here may indicate model inadequacies or omitted variable bias.21

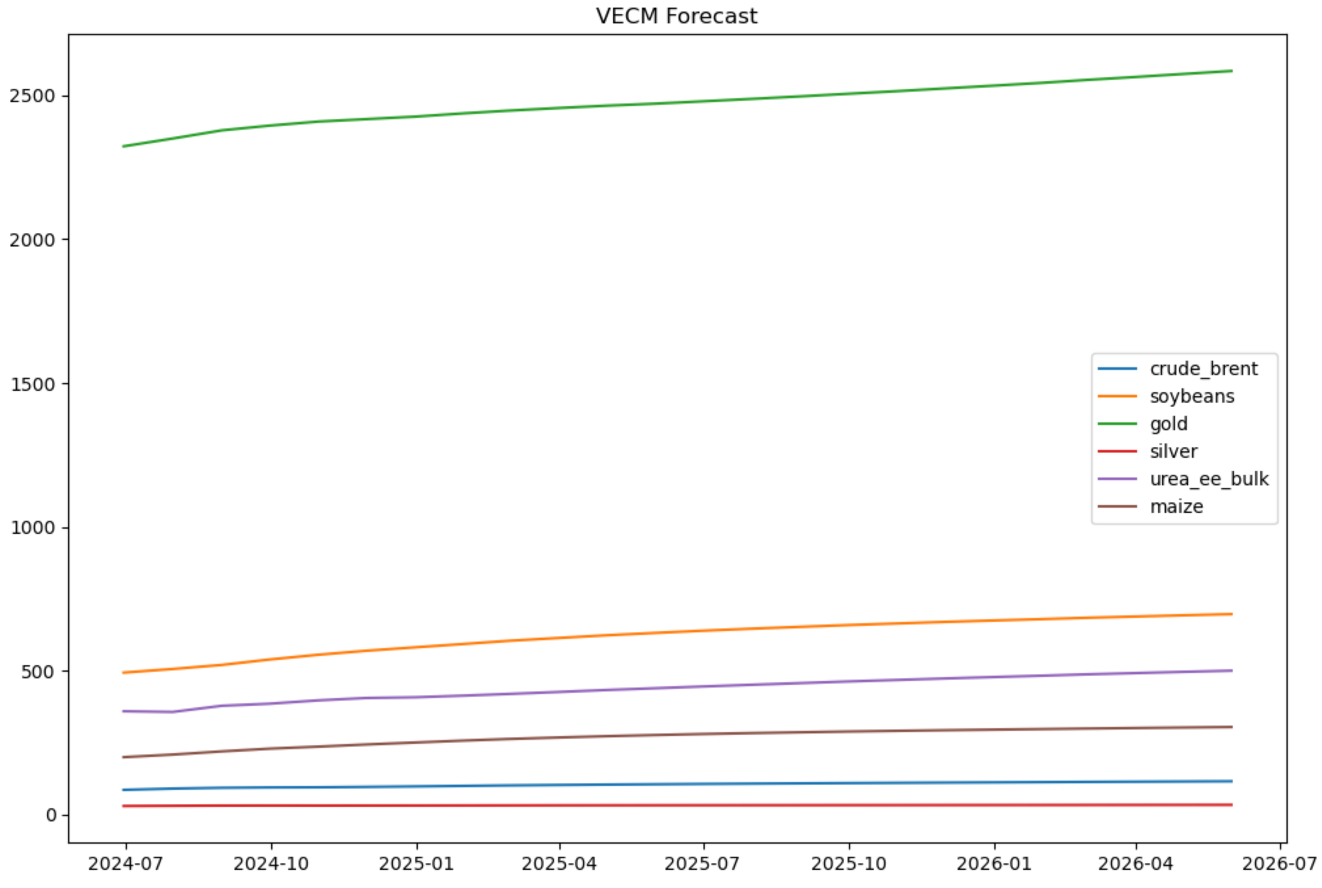
These results collectively help understand the dynamics and interrelationships between the variables (crude_brent, soybeans, gold, silver, urea_ee_bulk, and maize) in the context of the applied VAR model. Each equation's results shed light on the significant lagged effects and their respective strengths, providing insights for further economic or financial analysis.

Forecasting

Code :

```
# Forecasting using the VAR model
forecast = var_result.forecast(commodity_data.values[-var_result.k_ar:], steps=24)

# Convert forecast to DataFrame for plotting
forecast_df = pd.DataFrame(forecast, index=pd.date_range(start=commodity['date'].iloc[-1], periods=24, freq='M'))

# Plotting the forecast
plt.figure(figsize=(12, 8))
for col in forecast_df.columns:
    plt.plot(forecast_df.index, forecast_df[col], label=col)
plt.legend()
plt.title('VAR Forecast')
plt.show()
```

Result :



Interpretation:

Comparison of VAR and VECM Models: Both models provided valuable insights, but the VECM model was particularly effective in capturing the long-term relationships among the commodities. The presence of co-integration justified the use of VECM, which offered a more comprehensive understanding of the equilibrium adjustments.

• **Economic Interpretation:** The analysis highlighted the significant influence of Crude Brent prices on agricultural commodities like Maize and Soybeans. This relationship suggests that oil price fluctuations can substantially impact food prices, with implications for policymakers and market participants. Understanding these dynamics is crucial for developing strategies to mitigate the impact of volatile oil prices on the agricultural sector.

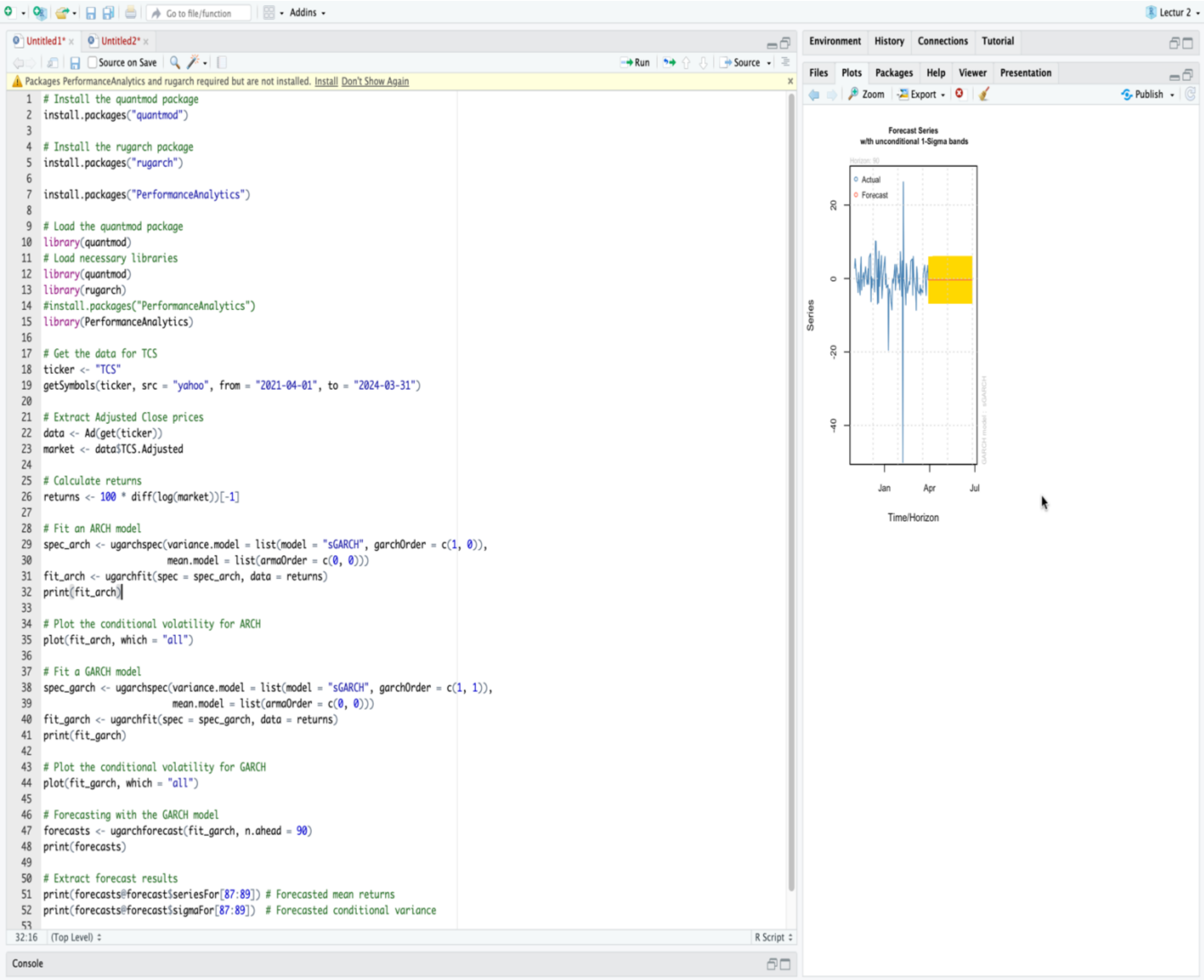
• **Limitations and Future Work:** While the analysis provided valuable insights, it is limited by data availability and quality. Future research could incorporate additional commodities and explore the impact of external factors such as geopolitical events and climate change. Enhancing the models with more sophisticated techniques could further improve the accuracy of the forecasts.

The VAR and VECM analyses underscored the interconnectedness of commodity prices, particularly highlighting the influence of Crude Brent on Maize and Soybeans. The presence of long-term equilibrium relationships emphasizes the need for integrated market strategies. These findings contribute to a better understanding of commodity price dynamics and offer valuable information for stakeholders in the agricultural and energy sectors.

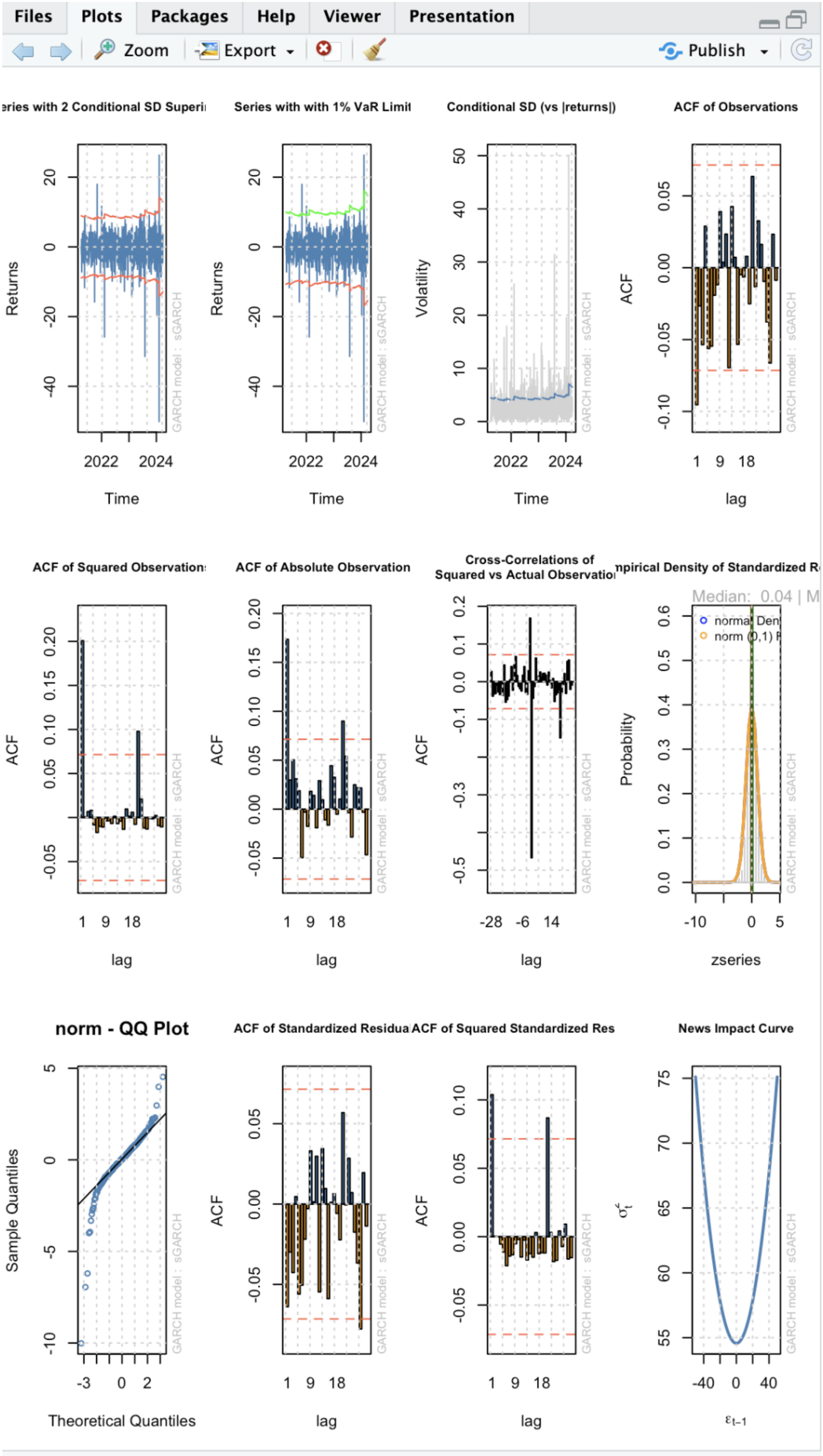
The following project analysis was done in R Studio. The outputs were the same as python. Below attached is the Output of RStudio.

R Codes :

Part A :



```r
# Install the quantmod package
install.packages("quantmod")

# Install the rugarch package
install.packages("rugarch")

install.packages("PerformanceAnalytics")

# Load the quantmod package
library(quantmod)
# Load necessary libraries
library(quantmod)
library(rugarch)
#install.packages("PerformanceAnalytics")
library(PerformanceAnalytics)

# Get the data for TCS
ticker <- "TCS"
getSymbols(ticker, src = "yahoo", from = "2021-04-01", to = "2024-03-31")

# Extract Adjusted Close prices
data <- Ad(get(ticker))
market <- data$TCS.Adjusted

# Calculate returns
returns <- 100 * diff(log(market))[-1]

# Fit an ARCH model
spec_arch <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 0)),
                        mean.model = list(armaOrder = c(0, 0)))
fit_arch <- ugarchfit(spec = spec_arch, data = returns)
print(fit_arch)

# Plot the conditional volatility for ARCH
plot(fit_arch, which = "all")

# Fit a GARCH model
spec_garch <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
                         mean.model = list(armaOrder = c(0, 0)))
fit_garch <- ugarchfit(spec = spec_garch, data = returns)
print(fit_garch)

# Plot the conditional volatility for GARCH
plot(fit_garch, which = "all")

# Forecasting with the GARCH model
forecasts <- ugarchforecast(fit_garch, n.ahead = 90)
print(forecasts)

# Extract forecast results
print(forecasts@forecast$seriesFor[87:89]) # Forecasted mean returns
print(forecasts@forecast$sigmaFor[87:89])  # Forecasted conditional variance
```


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Series with 2 Conditional SD Superi
Series with with 1% VaR Limit
Conditional SD (vs |returns|)
ACF of Observations
ACF of Squared Observation
ACF of Absolute Observation
Cross-Correlations of Squared vs Actual Observatio
Empirical Density of Standardized R
norm - QQ Plot
ACF of Standardized Residua
ACF of Squared Standardized Res
News Impact Curve
Returns
Volatility
ACF
Probability
Sample Quantiles
Time
lag
zseries
Theoretical Quantiles
GARCH model : sGARCH

Part B:

Packages urca and vars required but are not installed. Install Don't Show Again

```r
# Set working directory and load necessary libraries
setwd('/Users/kirthanshaker/Desktop/SCMA 631 Data Files ')
getwd()

install.packages("urca")
install.packages("vars")
# Load necessary libraries
library(readxl)
library(dplyr)
library(janitor)
library(urca)
library(vars)

# Load the dataset
df <- read_excel('/Users/kirthanshaker/Desktop/SCMA 631 Data Files /pinksheet.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)

# Select specific columns (Date and selected commodities)
commodity <- df[,c(1,3,25,70,72,61,31)] %>%
  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)
  }
```

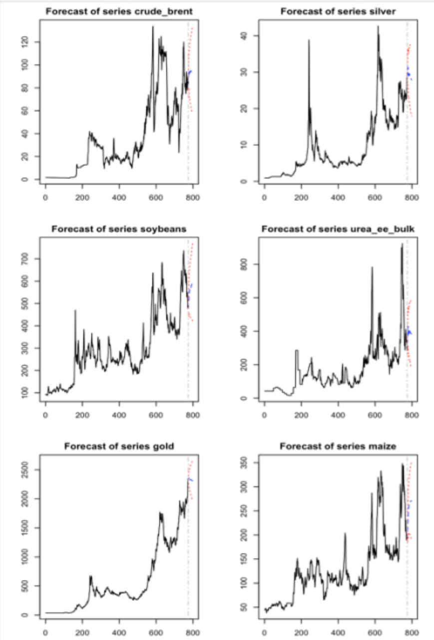
Untitled1* Untitled2*

Packages urca and vars required but are not installed. Install Don't Show Again

```r
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
```

42:80 (Top Level)

Console



CODES For Both Python and R :

R Codes :

```
# Install the quantmod package
install.packages("quantmod")

# Install the rugarch package
install.packages("rugarch")

install.packages("PerformanceAnalytics")

# Load the quantmod package
library(quantmod)
# Load necessary libraries
library(quantmod)
library(rugarch)
#install.packages("PerformanceAnalytics")
library(PerformanceAnalytics)

# Get the data for TCS
ticker <- "TCS"
getSymbols(ticker, src = "yahoo", from = "2021-04-01", to = "2024-03-31")

# Extract Adjusted Close prices
data <- Ad(get(ticker))
market <- data$TCS.Adjusted

# Calculate returns
returns <- 100 * diff(log(market))[-1]

# Fit an ARCH model
spec_arch <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 0)),
                        mean.model = list(armaOrder = c(0, 0)))
fit_arch <- ugarchfit(spec = spec_arch, data = returns)
print(fit_arch)

# Plot the conditional volatility for ARCH
plot(fit_arch, which = "all")

# Fit a GARCH model
spec_garch <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
                         mean.model = list(armaOrder = c(0, 0)))
fit_garch <- ugarchfit(spec = spec_garch, data = returns)
print(fit_garch)

# Plot the conditional volatility for GARCH
plot(fit_garch, which = "all")

# Forecasting with the GARCH model
forecasts <- ugarchforecast(fit_garch, n.ahead = 90)
print(forecasts)

# Extract forecast results
print(forecasts@forecast$seriesFor[87:89]) # Forecasted mean returns
print(forecasts@forecast$sigmaFor[87:89])  # Forecasted conditional variance

# Plotting the results
```

```r
plot(forecasts, which = "all")
```

Part b

```r
# Set working directory and load necessary libraries
setwd('/Users/kirthanshaker/Desktop/SCMA 631 Data Files ')
getwd()

install.packages("urca")
install.packages("vars")
# Load necessary libraries
library(readxl)
library(dplyr)
library(janitor)
library(urca)
library(vars)

# Load the dataset
df <- read_excel('/Users/kirthanshaker/Desktop/SCMA 631 Data Files /pinksheet.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)

# Select specific columns (Date and selected commodities)
commodity <- df[,c(1,3,25,70,72,61,31)] %>%
  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:", 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

Python Codes :

```
"cells": [
  {
   "cell_type": "markdown",
   "id": "e03a7695",
   "metadata": {},
   "source": [
    "### PART A: ARCH_GARCH Model"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 17,
   "id": "aaf49c87",
   "metadata": {},
   "outputs": [],
   "source": [
    "import yfinance as yf\n",
    "from arch import arch_model\n",
    "import matplotlib.pyplot as plt"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 24,
   "id": "525b15e4",
   "metadata": {},
   "outputs": [
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "[*********************100%%**********************]  1 of 1 completed\n"
     ]
    }
   ],
   "source": [
    "# Get the data for Tata Motors\n",
    "ticker = \"TCS.NS\"\n",
    "\n",
    "# Download the data\n",
    "data = yf.download(ticker, start=\"2021-04-01\", end=\"2024-03-31\")"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 25,
   "id": "94a666ed",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "                      Constant Mean - ARCH Model Results                      \n",
      "==============================================================================\n",
      "Dep. Variable:                Returns   R-squared:                       0.000\n",
      "Mean Model:             Constant Mean   Adj. R-squared:                  0.000\n",
```

"\n",
"Covariance estimator: robust\n"
]
},
{
"data": {
"image/png":
"text/plain": [
"<Figure size 640x480 with 1 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
}
],
"source": [
"# Create 'Returns' column\n",
"data['Returns'] = 100 * data['Adj Close'].pct_change().dropna()\n",
"\n",
"# Fit an ARCH model\n",
"arch_model_fit = arch_model(data['Returns'].dropna(), vol='ARCH', p=1).fit(disp='off')\n",
"print(arch_model_fit.summary())\n",
"\n",
"# Plot the conditional volatility\n",
"arch_model_fit.conditional_volatility.plot(title='Conditional Volatility (ARCH)')\n",
"plt.show()"
]
},
{
"cell_type": "code",
"execution_count": 28,
"id": "e26322fc",
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [

Part B

{
"cells": [
{
"cell_type": "markdown",
"id": "d170c36b",
"metadata": {},
"source": [
"### PART B: VAR, VECM Models on commodity prices"
]
},
{
"cell_type": "code",
"execution_count": 15,
"id": "b021cf87",
"metadata": {},
"outputs": [],
"source": [
"import pandas as pd\n",
"import numpy as np\n",
"from statsmodels.tsa.vector_ar.var_model import VAR\n",
"from statsmodels.tsa.vector_ar.vecm import coint_johansen\n",
"from statsmodels.tsa.stattools import adfuller\n",

```
    "import matplotlib.pyplot as plt"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 2,
   "id": "d00ee7fe",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "D:\\MDA\\Course\\Boot Camp\\SCMA 632\\Assignments\\A6b\n"
     ]
    }
   ],
   "source": [
    "# Set working directory\n",
    "import os\n",
    "os.chdir('D:\\MDA\\Course\\Boot Camp\\SCMA 632\\Assignments\\A6b')\n",
    "print(os.getcwd())"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 3,
   "id": "2b291b21",
   "metadata": {},
   "outputs": [],
   "source": [
    "# Load the data\n",
    "df = pd.read_excel('pinksheet.xlsx', sheet_name=\"Monthly Prices\", skiprows=6)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 4,
   "id": "4e344617",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/html": [
       "<div>\n",
       "<style scoped>\n",
       "    .dataframe tbody tr th:only-of-type {\n",
       "        vertical-align: middle;\n",
       "    }\n",
       "\n",
       "    .dataframe tbody tr th {\n",
       "        vertical-align: top;\n",
       "    }\n",
       "\n",
       "    .dataframe thead th {\n",
       "        text-align: right;\n",
       "    }\n",
       "</style>\n",
       "<table border=\"1\" class=\"dataframe\">\n",
       "  <thead>\n",
       "    <tr style=\"text-align: right;\">\n",
```

```
"      <th></th>\n",
"      <th>Unnamed: 0</th>\n",
"      <th>CRUDE_PETRO</th>\n",
"      <th>CRUDE_BRENT</th>\n",
"      <th>CRUDE_DUBAI</th>\n",
"      <th>CRUDE_WTI</th>\n",
"      <th>COAL_AUS</th>\n",
"      <th>COAL_SAFRICA</th>\n",
"      <th>NGAS_US</th>\n",
"      <th>NGAS_EUR</th>\n",
"      <th>NGAS_JP</th>\n",
"      <th>...</th>\n",
"      <th>ALUMINUM</th>\n",
"      <th>IRON_ORE</th>\n",
"      <th>COPPER</th>\n",
"      <th>LEAD</th>\n",
"      <th>Tin</th>\n",
"      <th>NICKEL</th>\n",
"      <th>Zinc</th>\n",
"      <th>GOLD</th>\n",
"      <th>PLATINUM</th>\n",
"      <th>SILVER</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>1960M01</td>\n",
"      <td>1.630000</td>\n",
"      <td>1.630</td>\n",
"      <td>1.63</td>\n",
"      <td>…</td>\n",
"      <td>…</td>\n",
"      <td>…</td>\n",
"      <td>0.1400</td>\n",
"      <td>0.404774</td>\n",
"      <td>…</td>\n",
"      <td>...</td>\n",
"      <td>511.471832</td>\n",
"      <td>11.42</td>\n",
"      <td>715.40</td>\n",
"      <td>206.10</td>\n",
"      <td>2180.40</td>\n",
"      <td>1631.00</td>\n",
"      <td>260.80</td>\n",
"      <td>35.27</td>\n",
"      <td>83.50</td>\n",
"      <td>0.9137</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>1960M02</td>\n",
"      <td>1.630000</td>\n",
"      <td>1.630</td>\n",
"      <td>1.63</td>\n",
"      <td>…</td>\n",
"      <td>…</td>\n",
"      <td>…</td>\n",
"      <td>0.1400</td>\n",
"      <td>0.404774</td>\n",
"      <td>…</td>\n",
```

```
"      <td>...</td>\n",
"      <td>511.471832</td>\n",
"      <td>11.42</td>\n",
"      <td>728.19</td>\n",
"      <td>203.70</td>\n",
"      <td>2180.40</td>\n",
"      <td>1631.00</td>\n",
"      <td>244.90</td>\n",
"      <td>35.27</td>\n",
"      <td>83.50</td>\n",
"      <td>0.9137</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>1960M03</td>\n",
"      <td>1.630000</td>\n",
"      <td>1.630</td>\n",
"      <td>1.63</td>\n",
"      <td>…</td>\n",
"      <td>…</td>\n",
"      <td>…</td>\n",
"      <td>0.1400</td>\n",
"      <td>0.404774</td>\n",
"      <td>…</td>\n",
"      <td>...</td>\n",
"      <td>511.471832</td>\n",
"      <td>11.42</td>\n",
"      <td>684.94</td>\n",
"      <td>210.30</td>\n",
"      <td>2173.80</td>\n",
"      <td>1631.00</td>\n",
"      <td>248.70</td>\n",
"      <td>35.27</td>\n",
"      <td>83.50</td>\n",
"      <td>0.9137</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>1960M04</td>\n",
"      <td>1.630000</td>\n",
"      <td>1.630</td>\n",
"      <td>1.63</td>\n",
"      <td>…</td>\n",
"      <td>…</td>\n",
"      <td>…</td>\n",
"      <td>0.1400</td>\n",
"      <td>0.404774</td>\n",
"      <td>…</td>\n",
"      <td>...</td>\n",
"      <td>511.471832</td>\n",
"      <td>11.42</td>\n",
"      <td>723.11</td>\n",
"      <td>213.60</td>\n",
"      <td>2178.20</td>\n",
"      <td>1631.00</td>\n",
"      <td>254.60</td>\n",
"      <td>35.27</td>\n",
"      <td>83.50</td>\n",
"      <td>0.9137</td>\n",
"    </tr>\n",
"    <tr>\n",
```

```
"    <th>4</th>\n",
"    <td>1960M05</td>\n",
"    <td>1.630000</td>\n",
"    <td>1.630</td>\n",
"    <td>1.63</td>\n",
"    <td>…</td>\n",
"    <td>…</td>\n",
"    <td>…</td>\n",
"    <td>0.1400</td>\n",
"    <td>0.404774</td>\n",
"    <td>…</td>\n",
"    <td>...</td>\n",
"    <td>511.471832</td>\n",
"    <td>11.42</td>\n",
"    <td>684.75</td>\n",
"    <td>213.40</td>\n",
"    <td>2162.70</td>\n",
"    <td>1631.00</td>\n",
"    <td>253.80</td>\n",
"    <td>35.27</td>\n",
"    <td>83.50</td>\n",
"    <td>0.9137</td>\n",
"   </tr>\n",
"   <tr>\n",
"    <th>...</th>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"    <td>...</td>\n",
"   </tr>\n",
"   <tr>\n",
"    <th>769</th>\n",
"    <td>2024M02</td>\n",
"    <td>80.548000</td>\n",
"    <td>83.764</td>\n",
"    <td>81.18</td>\n",
"    <td>76.7</td>\n",
"    <td>124.22</td>\n",
"    <td>105.193</td>\n",
"    <td>1.7211</td>\n",
"    <td>8.148381</td>\n",
"    <td>13.644993</td>\n",
"    <td>...</td>\n",
"    <td>2179.460000</td>\n",
```

"   <td>124.39</td>\n",
"   <td>8304.95</td>\n",
"   <td>2079.83</td>\n",
"   <td>26104.10</td>\n",
"   <td>16338.46</td>\n",
"   <td>2360.09</td>\n",
"   <td>2023.24</td>\n",
"   <td>894.29</td>\n",
"   <td>22.6570</td>\n",
"   </tr>\n",
"   <tr>\n",
"   <th>770</th>\n",
"   <td>2024M03</td>\n",
"   <td>83.545667</td>\n",
"   <td>85.447</td>\n",
"   <td>84.70</td>\n",
"   <td>80.49</td>\n",
"   <td>131.49</td>\n",
"   <td>104.84</td>\n",
"   <td>1.4999</td>\n",
"   <td>8.553726</td>\n",
"   <td>13.185629</td>\n",
"   <td>...</td>\n",
"   <td>2226.160000</td>\n",
"   <td>109.79</td>\n",
"   <td>8689.13</td>\n",
"   <td>2056.20</td>\n",
"   <td>27450.46</td>\n",
"   <td>17438.83</td>\n",
"   <td>2461.04</td>\n",
"   <td>2158.01</td>\n",
"   <td>908.75</td>\n",
"   <td>24.5180</td>\n",
"   </tr>\n",
"   <tr>\n",
"   <th>771</th>\n",
"   <td>2024M04</td>\n",
"   <td>88.011333</td>\n",
"   <td>90.054</td>\n",
"   <td>89.39</td>\n",
"   <td>84.59</td>\n",
"   <td>134.97</td>\n",
"   <td>104.89</td>\n",
"   <td>1.5967</td>\n",
"   <td>9.085119</td>\n",
"   <td>11.87777</td>\n",
"   <td>...</td>\n",
"   <td>2506.100000</td>\n",
"   <td>112.75</td>\n",
"   <td>9464.43</td>\n",
"   <td>2129.46</td>\n",
"   <td>31774.50</td>\n",
"   <td>18163.95</td>\n",
"   <td>2732.74</td>\n",
"   <td>2331.45</td>\n",
"   <td>940.18</td>\n",
"   <td>27.4940</td>\n",
"   </tr>\n",
"   <tr>\n",
"   <th>772</th>\n",
"   <td>2024M05</td>\n",

```
"      <td>81.445000</td>\n",
"      <td>81.995</td>\n",
"      <td>83.53</td>\n",
"      <td>78.81</td>\n",
"      <td>142.01</td>\n",
"      <td>105.63</td>\n",
"      <td>2.1314</td>\n",
"      <td>10.123066</td>\n",
"      <td>12.162896</td>\n",
"      <td>...</td>\n",
"      <td>2564.540000</td>\n",
"      <td>118.88</td>\n",
"      <td>10139.33</td>\n",
"      <td>2220.81</td>\n",
"      <td>32977.51</td>\n",
"      <td>19586.98</td>\n",
"      <td>2959.13</td>\n",
"      <td>2351.13</td>\n",
"      <td>1014.68</td>\n",
"      <td>29.3600</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>773</th>\n",
"      <td>2024M06</td>\n",
"      <td>81.205000</td>\n",
"      <td>82.555</td>\n",
"      <td>82.17</td>\n",
"      <td>78.89</td>\n",
"      <td>135.1</td>\n",
"      <td>105.3</td>\n",
"      <td>2.5123</td>\n",
"      <td>10.868978</td>\n",
"      <td>12.11</td>\n",
"      <td>...</td>\n",
"      <td>2497.610000</td>\n",
"      <td>107.45</td>\n",
"      <td>9648.17</td>\n",
"      <td>2147.10</td>\n",
"      <td>32032.70</td>\n",
"      <td>17498.01</td>\n",
"      <td>2809.19</td>\n",
"      <td>2326.44</td>\n",
"      <td>985.08</td>\n",
"      <td>29.5770</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>774 rows × 72 columns</p>\n",
"</div>"
],
"text/plain": [
"    Unnamed: 0  CRUDE_PETRO  CRUDE_BRENT  CRUDE_DUBAI CRUDE_WTI COAL_AUS  \\\n",
"0      1960M01     1.630000        1.630        1.63        …        …   \n",
"1      1960M02     1.630000        1.630        1.63        …        …   \n",
"2      1960M03     1.630000        1.630        1.63        …        …   \n",
"3      1960M04     1.630000        1.630        1.63        …        …   \n",
"4      1960M05     1.630000        1.630        1.63        …        …   \n",
"..         ...          ...          ...         ...       ...      ...   \n",
"769    2024M02    80.548000       83.764       81.18      76.7   124.22   \n",
"770    2024M03    83.545667       85.447       84.70     80.49   131.49   \n",
"771    2024M04    88.011333       90.054       89.39     84.59   134.97   \n",
```

"772   2024M05   81.445000   81.995   83.53   78.81   142.01   \n",
"773   2024M06   81.205000   82.555   82.17   78.89   135.1   \n",
"\n",
"   COAL_SAFRICA NGAS_US   NGAS_EUR   NGAS_JP ...   ALUMINUM IRON_ORE \\\n",
"0   …   0.1400   0.404774   … ...   511.471832   11.42   \n",
"1   …   0.1400   0.404774   … ...   511.471832   11.42   \n",
"2   …   0.1400   0.404774   … ...   511.471832   11.42   \n",
"3   …   0.1400   0.404774   … ...   511.471832   11.42   \n",
"4   …   0.1400   0.404774   … ...   511.471832   11.42   \n",
"..   ...   .   \n",
"769   105.193   1.7211   8.148381 13.644993 ... 2179.460000   124.39   \n",
"770   104.84   1.4999   8.553726 13.185629 ... 2226.160000   109.79   \n",
"771   104.89   1.5967   9.085119   11.87777 ... 2506.100000   112.75   \n",
"772   105.63   2.1314 10.123066 12.162896 ... 2564.540000   118.88   \n",
"773   105.3   2.5123 10.868978   12.11 ... 2497.610000   107.45   \n",
"\n",
"   COPPER   LEAD   Tin   NICKEL   Zinc   GOLD PLATINUM \\\n",
"0   715.40   206.10   2180.40   1631.00   260.80   35.27   83.50   \n",
"1   728.19   203.70   2180.40   1631.00   244.90   35.27   83.50   \n",
"2   684.94   210.30   2173.80   1631.00   248.70   35.27   83.50   \n",
"3   723.11   213.60   2178.20   1631.00   254.60   35.27   83.50   \n",
"4   684.75   213.40   2162.70   1631.00   253.80   35.27   83.50   \n",
"..   ...   .   \n",
"769   8304.95 2079.83 26104.10 16338.46 2360.09 2023.24   894.29   \n",
"770   8689.13 2056.20 27450.46 17438.83 2461.04 2158.01   908.75   \n",
"771   9464.43 2129.46 31774.50 18163.95 2732.74 2331.45   940.18   \n",
"772 10139.33 2220.81 32977.51 19586.98 2959.13 2351.13   1014.68   \n",
"773   9648.17 2147.10 32032.70 17498.01 2809.19 2326.44   985.08   \n",
"\n",
"   SILVER \n",
"0   0.9137 \n",
"1   0.9137 \n",
"2   0.9137 \n",
"3   0.9137 \n",
"4   0.9137 \n",
"..   ... \n",
"769 22.6570 \n",
"770 24.5180 \n",
"771 27.4940 \n",
"772 29.3600 \n",
"773 29.5770 \n",
"\n",
"[774 rows x 72 columns]"
]
},
"execution_count": 4,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"df"
]
},
{
"cell_type": "code",
"execution_count": 5,
"id": "15ef48d0",
"metadata": {},
"outputs": [
{

"name": "stdout",
"output_type": "stream",
"text": [
"['Unnamed: 0', 'CRUDE_PETRO', 'CRUDE_BRENT', 'CRUDE_DUBAI', 'CRUDE_WTI', 'COAL_AUS', 'COAL_SAFRICA', 'NGAS_US', 'NGAS_EUR', 'NGAS_JP', 'iNATGAS', 'COCOA', 'COFFEE_ARABIC', 'COFFEE_ROBUS', 'TEA_AVG', 'TEA_COLOMBO', 'TEA_KOLKATA', 'TEA_MOMBASA', 'COCONUT_OIL', 'GRNUT', 'FISH_MEAL', 'GRNUT_OIL', 'PALM_OIL', 'PLMKRNL_OIL', 'SOYBEANS', 'SOYBEAN_OIL', 'SOYBEAN_MEAL', 'RAPESEED_OIL', 'SUNFLOWER_OIL', 'BARLEY', 'MAIZE', 'SORGHUM', 'RICE_05', 'RICE_25', 'RICE_A1', 'RICE_05_VNM', 'WHEAT_US_SRW', 'WHEAT_US_HRW', 'BANANA_EU', 'BANANA_US', 'ORANGE', 'BEEF', 'CHICKEN', 'LAMB', 'SHRIMP_MEX', 'SUGAR_EU', 'SUGAR_US', 'SUGAR_WLD', 'TOBAC_US', 'LOGS_CMR', 'LOGS_MYS', 'SAWNWD_CMR', 'SAWNWD_MYS', 'PLYWOOD', 'COTTON_A_INDX', 'RUBBER_TSR20', 'RUBBER1_MYSG', 'PHOSROCK', 'DAP', 'TSP', 'UREA_EE_BULK', 'POTASH', 'ALUMINUM', 'IRON_ORE', 'COPPER', 'LEAD', 'Tin', 'NICKEL', 'Zinc', 'GOLD', 'PLATINUM', 'SILVER']\n"
]
}
],
"source": [
"print(list(df.columns))"
]
},
{
"cell_type": "code",
"execution_count": 6,
"id": "53d33f9d",
"metadata": {},
"outputs": [],
"source": [
"# Rename the first column to \"Date\"\n",
"df.rename(columns={df.columns[0]: 'Date'}, inplace=True)"
]
},
{
"cell_type": "code",
"execution_count": 7,
"id": "d437a209",
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"0   1960M01\n",
"1   1960M02\n",
"2   1960M03\n",
"3   1960M04\n",
"4   1960M05\n",
"   ...   \n",
"769   2024M02\n",
"770   2024M03\n",
"771   2024M04\n",
"772   2024M05\n",
"773   2024M06\n",
"Name: Date, Length: 774, dtype: object"
]
},
"execution_count": 7,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"df.Date"
]
},
{
"cell_type": "code",

"execution_count": 21,
"id": "1365f241",
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"Date datetime64[ns]\n",
"CRUDE_PETRO float64\n",
"CRUDE_BRENT float64\n",
"CRUDE_DUBAI float64\n",
"CRUDE_WTI object\n",
" ... \n",
"NICKEL float64\n",
"Zinc float64\n",
"GOLD float64\n",
"PLATINUM float64\n",
"SILVER float64\n",
"Length: 72, dtype: object\n"
]
}
],
"source": [
"# Convert the Date column to datetime format\n",
"def parse_date(date_str):\n",
" # Split the string by 'M' to separate year and month\n",
" year, month = date_str.split('M')\n",
" # Create a date string in the format 'YYYY-MM-01'\n",
" return f\"{year}-{month}-01\"\n",
"\n",
"df['Date'] = pd.to_datetime(df['Date'].apply(parse_date))\n",
"\n",
"# Print the data types to confirm the conversion\n",
"print(df.dtypes)"
]
},
{
"cell_type": "code",
"execution_count": 22,
"id": "42818fb0",
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"0 1960-01-01\n",
"1 1960-02-01\n",
"2 1960-03-01\n",
"3 1960-04-01\n",
"4 1960-05-01\n",
" ... \n",
"769 2024-02-01\n",
"770 2024-03-01\n",
"771 2024-04-01\n",
"772 2024-05-01\n",
"773 2024-06-01\n",
"Name: Date, Length: 774, dtype: datetime64[ns]"
]
},
"execution_count": 22,

```
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "df.Date"
 ]
},
{
 "cell_type": "code",
 "execution_count": 31,
 "id": "0705fdbc",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "date           datetime64[ns]\n",
    "crude_brent           float64\n",
    "soybeans              float64\n",
    "gold                  float64\n",
    "silver                float64\n",
    "urea_ee_bulk          float64\n",
    "maize                 float64\n",
    "dtype: object\n"
   ]
  }
 ],
 "source": [
  "# Select specific columns (Date and selected commodities)\n",
  "commodity = df.iloc[:, [0, 2, 24, 69, 71, 60, 30]]\n",
  "commodity.columns = [col.lower().replace(' ', '_') for col in commodity.columns]  # Clean column names\n",
  "\n",
  "print(commodity.dtypes)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 32,
 "id": "31d7aa4d",
 "metadata": {},
 "outputs": [],
 "source": [
  "# Remove the Date column for analysis\n",
  "commodity_data = commodity.drop(columns=['date'])\n",
  "\n",
  "# Column names to test\n",
  "columns_to_test = commodity_data.columns"
 ]
},
{
 "cell_type": "code",
 "execution_count": 33,
 "id": "fd141e9b",
 "metadata": {
  "scrolled": false
 },
 "outputs": [],
 "source": [
  "# Initialize counters and lists for stationary and non-stationary columns\n",
```

```
   "non_stationary_count = 0\n",
   "stationary_columns = []\n",
   "non_stationary_columns = []"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 34,
  "id": "fc2afe57",
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "\n",
     "ADF test result for column: crude_brent\n",
     "ADF Statistic: -1.5078661910935385\n",
     "p-value: 0.5296165197702377\n",
     "\n",
     "ADF test result for column: soybeans\n",
     "ADF Statistic: -2.423146452741887\n",
     "p-value: 0.13530977427790458\n",
     "\n",
     "ADF test result for column: gold\n",
     "ADF Statistic: 1.3430517021932975\n",
     "p-value: 0.9968394353612381\n",
     "\n",
     "ADF test result for column: silver\n",
     "ADF Statistic: -1.39729471074622\n",
     "p-value: 0.5835723787985774\n",
     "\n",
     "ADF test result for column: urea_ee_bulk\n",
     "ADF Statistic: -2.5101716315209095\n",
     "p-value: 0.11301903181624623\n",
     "\n",
     "ADF test result for column: maize\n",
     "ADF Statistic: -2.4700451060920425\n",
     "p-value: 0.12293380919376751\n"
    ]
   }
  ],
  "source": [
   "# Loop through each column and perform the ADF test\n",
   "for col in columns_to_test:\n",
   "    adf_result = adfuller(commodity_data[col])\n",
   "    p_value = adf_result[1]  # Extract p-value for the test\n",
   "    print(f\"\\nADF test result for column: {col}\")\n",
   "    print(f\"ADF Statistic: {adf_result[0]}\")\n",
   "    print(f\"p-value: {p_value}\")\n",
   "    \n",
   "    # Check if the p-value is greater than 0.05 (commonly used threshold)\n",
   "    if p_value > 0.05:\n",
   "        non_stationary_count += 1\n",
   "        non_stationary_columns.append(col)\n",
   "    else:\n",
   "        stationary_columns.append(col)"
  ]
 },
 {
  "cell_type": "code",
```

```
   "execution_count": 35,
   "id": "fa7c7bc2",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "\n",
      "Number of non-stationary columns: 6\n",
      "Non-stationary columns: ['crude_brent', 'soybeans', 'gold', 'silver', 'urea_ee_bulk', 'maize']\n",
      "Stationary columns: []\n"
     ]
    }
   ],
   "source": [
    "# Print the number of non-stationary columns and the lists of stationary and non-stationary columns\n",
    "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}\")"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 36,
   "id": "3441f2c0",
   "metadata": {},
   "outputs": [],
   "source": [
    "# Co-Integration Test (Johansen's Test)\n",
    "def johansen_test(df, alpha=0.05):\n",
    "    out = coint_johansen(df, det_order=0, k_ar_diff=1)\n",
    "    d = {'0.90': 0, '0.95': 1, '0.99': 2}\n",
    "    traces = out.lr1\n",
    "    cvts = out.cvt[:, d[str(1 - alpha)]]\n",
    "    print(f\"Trace statistic: {traces}\")\n",
    "    print(f\"Critical values: {cvts}\")\n",
    "    print(f\"Eigenvalues: {out.eig}\")\n",
    "    for col, trace, cvt in zip(df.columns, traces, cvts):\n",
    "        if trace > cvt:\n",
    "            print(f\"{col} is cointegrated.\")\n",
    "        else:\n",
    "            print(f\"{col} is not cointegrated.\")\n",
    "    return out"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 37,
   "id": "4f77546d",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Trace statistic: [261.5548149  167.67790177  98.11781369  53.4617083   21.6404865\n",
      "   4.01416422]\n",
      "Critical values: [95.7542 69.8189 47.8545 29.7961 15.4943  3.8415]\n",
      "Eigenvalues: [0.11449947 0.08616362 0.05620349 0.04038124 0.02257335 0.0051862 ]\n",
      "crude_brent is cointegrated.\n",
```

```
    "soybeans is cointegrated.\n",
    "gold is cointegrated.\n",
    "silver is cointegrated.\n",
    "urea_ee_bulk is cointegrated.\n",
    "maize is cointegrated.\n"
   ]
  }
 ],
 "source": [
  "# Perform Johansen cointegration test\n",
  "coint_test = johansen_test(commodity_data)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 38,
 "id": "9e80c826",
 "metadata": {},
 "outputs": [],
 "source": [
  "# Number of cointegrating relationships (assuming r = 1 if there's at least one significant eigenvalue)\n",
  "r = sum(coint_test.lr1 > coint_test.cvt[:, 1])  # Replace with the actual number from the test results"
 ]
},
{
 "cell_type": "code",
 "execution_count": 39,
 "id": "2630f888",
 "metadata": {
  "scrolled": false
 },
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "  Summary of Regression Results   \n",
    "==================================\n",
    "Model:                         VAR\n",
    "Method:                        OLS\n",
    "Date:           Wed, 24, Jul, 2024\n",
    "Time:                     21:09:56\n",
    "--------------------------------------------------------------------\n",
    "No. of Equations:         6.00000    BIC:                    26.7336\n",
    "Nobs:                     768.000    HQIC:                   25.9079\n",
    "Log likelihood:          -16066.7    FPE:                1.06530e+11\n",
    "AIC:                      25.3912    Det(Omega_mle):     8.03276e+10\n",
    "--------------------------------------------------------------------\n",
    "Results for equation crude_brent\n",
    "==================================================================================\n",
    "                    coefficient       std. error           t-stat            prob\n",
    "----------------------------------------------------------------------------------\n",
    "const                 -0.574387         0.457999           -1.254           0.210\n",
    "L1.crude_brent         1.288559         0.039600           32.539           0.000\n",
    "L1.soybeans            0.011187         0.007736            1.446           0.148\n",
    "L1.gold                0.000565         0.006577            0.086           0.932\n",
    "L1.silver             -0.012011         0.165664           -0.073           0.942\n",
    "L1.urea_ee_bulk       -0.011804         0.004637           -2.546           0.011\n",
    "L1.maize               0.020438         0.017600            1.161           0.246\n",
    "L2.crude_brent        -0.368186         0.064243           -5.731           0.000\n",
    "L2.soybeans            0.008609         0.010762            0.800           0.424\n",
```

```
"L2.gold               -0.007451         0.010640           -0.700           0.484\n",
"L2.silver              0.199505         0.275939            0.723           0.470\n",
"L2.urea_ee_bulk        0.015907         0.007085            2.245           0.025\n",
"L2.maize              -0.022252         0.025791           -0.863           0.388\n",
"L3.crude_brent        -0.011259         0.066566           -0.169           0.866\n",
"L3.soybeans           -0.024881         0.010745           -2.316           0.021\n",
"L3.gold                0.020019         0.010832            1.848           0.065\n",
"L3.silver             -0.211736         0.295689           -0.716           0.474\n",
"L3.urea_ee_bulk       -0.004688         0.007391           -0.634           0.526\n",
"L3.maize               0.031954         0.026095            1.225           0.221\n",
"L4.crude_brent         0.022815         0.066751            0.342           0.733\n",
"L4.soybeans            0.009171         0.010841            0.846           0.398\n",
"L4.gold               -0.000726         0.010669           -0.068           0.946\n",
"L4.silver              0.037894         0.296398            0.128           0.898\n",
"L4.urea_ee_bulk        0.000123         0.007431            0.017           0.987\n",
"L4.maize              -0.043400         0.026026           -1.668           0.095\n",
"L5.crude_brent         0.008371         0.065302            0.128           0.898\n",
"L5.soybeans            0.009904         0.010927            0.906           0.365\n",
"L5.gold               -0.005274         0.010504           -0.502           0.616\n",
"L5.silver             -0.077226         0.280104           -0.276           0.783\n",
"L5.urea_ee_bulk       -0.004359         0.007074           -0.616           0.538\n",
"L5.maize               0.034108         0.026066            1.309           0.191\n",
"L6.crude_brent         0.021961         0.040570            0.541           0.588\n",
"L6.soybeans           -0.007763         0.007913           -0.981           0.327\n",
"L6.gold               -0.007032         0.006708           -1.048           0.295\n",
"L6.silver              0.137240         0.167517            0.819           0.413\n",
"L6.urea_ee_bulk        0.001589         0.004568            0.348           0.728\n",
"L6.maize              -0.021898         0.017481           -1.253           0.210\n",
"==============================================================================\n",
"\n",
"Results for equation soybeans\n",
"==============================================================================\n",
"                     coefficient       std. error           t-stat            prob\n",
"------------------------------------------------------------------------------\n",
"const                  11.317337         2.521090            4.489           0.000\n",
"L1.crude_brent          0.214138         0.217982            0.982           0.326\n",
"L1.soybeans            1.013966         0.042581           23.813           0.000\n",
"L1.gold                 0.013684         0.036203            0.378           0.705\n",
"L1.silver               0.305354         0.911909            0.335           0.738\n",
"L1.urea_ee_bulk       -0.009017         0.025525           -0.353           0.724\n",
"L1.maize               0.314169         0.096881            3.243           0.001\n",
"L2.crude_brent        -0.103000         0.353632           -0.291           0.771\n",
"L2.soybeans           -0.017674         0.059238           -0.298           0.765\n",
"L2.gold               -0.064859         0.058571           -1.107           0.268\n",
"L2.silver               0.926647         1.518924            0.610           0.542\n",
"L2.urea_ee_bulk        0.041336         0.039000            1.060           0.289\n",
"L2.maize              -0.285567         0.141970           -2.011           0.044\n",
"L3.crude_brent        -0.077825         0.366417           -0.212           0.832\n",
"L3.soybeans           -0.141878         0.059147           -2.399           0.016\n",
"L3.gold                0.131659         0.059625            2.208           0.027\n",
"L3.silver             -2.231664         1.627642           -1.371           0.170\n",
"L3.urea_ee_bulk       -0.018121         0.040686           -0.445           0.656\n",
"L3.maize               0.159302         0.143644            1.109           0.267\n",
"L4.crude_brent          0.036457         0.367435            0.099           0.921\n",
"L4.soybeans            0.084280         0.059676            1.412           0.158\n",
"L4.gold               -0.093822         0.058728           -1.598           0.110\n",
"L4.silver              1.219334         1.631547            0.747           0.455\n",
"L4.urea_ee_bulk        0.011285         0.040903            0.276           0.783\n",
"L4.maize              -0.411196         0.143261           -2.870           0.004\n",
"L5.crude_brent        -0.053674         0.359462           -0.149           0.881\n",
"L5.soybeans           -0.059902         0.060151           -0.996           0.319\n",
```

```
"L5.gold             0.023087         0.057818           0.399           0.690\n",
"L5.silver           0.252871         1.541852           0.164           0.870\n",
"L5.urea_ee_bulk    -0.011316         0.038941          -0.291           0.771\n",
"L5.maize            0.302401         0.143482           2.108           0.035\n",
"L6.crude_brent     -0.062569         0.223320          -0.280           0.779\n",
"L6.soybeans         0.028889         0.043560           0.663           0.507\n",
"L6.gold             0.001505         0.036925           0.041           0.967\n",
"L6.silver          -0.176909         0.922107          -0.192           0.848\n",
"L6.urea_ee_bulk     0.010044         0.025142           0.399           0.690\n",
"L6.maize           -0.045677         0.096225          -0.475           0.635\n",
"===================================================================================\n",
"\n",
"Results for equation gold\n",
"===================================================================================\n",
"                    coefficient       std. error           t-stat            prob\n",
"-----------------------------------------------------------------------------------\n",
"const                0.177098         3.702239           0.048           0.962\n",
"L1.crude_brent       0.190589         0.320109           0.595           0.552\n",
"L1.soybeans          0.019501         0.062531           0.312           0.755\n",
"L1.gold             1.228901         0.053164          23.115           0.000\n",
"L1.silver           0.316301         1.339144           0.236           0.813\n",
"L1.urea_ee_bulk    -0.125678         0.037484          -3.353           0.001\n",
"L1.maize            0.279896         0.142270           1.967           0.049\n",
"L2.crude_brent      0.074271         0.519311           0.143           0.886\n",
"L2.soybeans         0.037551         0.086991           0.432           0.666\n",
"L2.gold            -0.276183         0.086012          -3.211           0.001\n",
"L2.silver          -3.352388         2.230551          -1.503           0.133\n",
"L2.urea_ee_bulk     0.215119         0.057271           3.756           0.000\n",
"L2.maize           -0.305428         0.208485          -1.465           0.143\n",
"L3.crude_brent     -0.688550         0.538086          -1.280           0.201\n",
"L3.soybeans        -0.222153         0.086857          -2.558           0.011\n",
"L3.gold             0.170371         0.087559           1.946           0.052\n",
"L3.silver           0.453043         2.390204           0.190           0.850\n",
"L3.urea_ee_bulk    -0.154341         0.059747          -2.583           0.010\n",
"L3.maize            0.492114         0.210943           2.333           0.020\n",
"L4.crude_brent      0.381592         0.539582           0.707           0.479\n",
"L4.soybeans         0.251772         0.087634           2.873           0.004\n",
"L4.gold            -0.151613         0.086243          -1.758           0.079\n",
"L4.silver           3.646825         2.395938           1.522           0.128\n",
"L4.urea_ee_bulk     0.066199         0.060066           1.102           0.270\n",
"L4.maize           -1.026908         0.210379          -4.881           0.000\n",
"L5.crude_brent     -0.125251         0.527873          -0.237           0.812\n",
"L5.soybeans        -0.157098         0.088332          -1.778           0.075\n",
"L5.gold             0.110733         0.084906           1.304           0.192\n",
"L5.silver          -1.459901         2.264221          -0.645           0.519\n",
"L5.urea_ee_bulk     0.047764         0.057185           0.835           0.404\n",
"L5.maize            0.583033         0.210704           2.767           0.006\n",
"L6.crude_brent      0.320187         0.327947           0.976           0.329\n",
"L6.soybeans         0.110200         0.063968           1.723           0.085\n",
"L6.gold            -0.073845         0.054225          -1.362           0.173\n",
"L6.silver          -0.453634         1.354121          -0.335           0.738\n",
"L6.urea_ee_bulk    -0.076808         0.036922          -2.080           0.037\n",
"L6.maize           -0.077152         0.141307          -0.546           0.585\n",
"===================================================================================\n",
"\n",
"Results for equation silver\n",
"===================================================================================\n",
"                    coefficient       std. error           t-stat            prob\n",
"-----------------------------------------------------------------------------------\n",
"const               -0.072930         0.149120          -0.489           0.625\n",
"L1.crude_brent       0.008049         0.012893           0.624           0.532\n",
```

```
"L1.soybeans          0.001756        0.002519          0.697          0.486\n",
"L1.gold             -0.002671        0.002141         -1.248          0.212\n",
"L1.silver            1.340090        0.053938         24.845          0.000\n",
"L1.urea_ee_bulk     -0.003586        0.001510         -2.375          0.018\n",
"L1.maize             0.011821        0.005730          2.063          0.039\n",
"L2.crude_brent       0.014541        0.020917          0.695          0.487\n",
"L2.soybeans         -0.000991        0.003504         -0.283          0.777\n",
"L2.gold              0.003938        0.003464          1.137          0.256\n",
"L2.silver           -0.665510        0.089843         -7.408          0.000\n",
"L2.urea_ee_bulk      0.002013        0.002307          0.873          0.383\n",
"L2.maize            -0.001179        0.008397         -0.140          0.888\n",
"L3.crude_brent      -0.033019        0.021673         -1.523          0.128\n",
"L3.soybeans         -0.003366        0.003498         -0.962          0.336\n",
"L3.gold              0.002395        0.003527          0.679          0.497\n",
"L3.silver            0.187709        0.096273          1.950          0.051\n",
"L3.urea_ee_bulk      0.001209        0.002407          0.503          0.615\n",
"L3.maize             0.002916        0.008496          0.343          0.731\n",
"L4.crude_brent       0.019566        0.021733          0.900          0.368\n",
"L4.soybeans          0.003541        0.003530          1.003          0.316\n",
"L4.gold             -0.001627        0.003474         -0.468          0.639\n",
"L4.silver            0.118333        0.096504          1.226          0.220\n",
"L4.urea_ee_bulk     -0.003052        0.002419         -1.262          0.207\n",
"L4.maize            -0.026818        0.008474         -3.165          0.002\n",
"L5.crude_brent      -0.024297        0.021262         -1.143          0.253\n",
"L5.soybeans         -0.000816        0.003558         -0.229          0.819\n",
"L5.gold              0.002731        0.003420          0.799          0.424\n",
"L5.silver           -0.156757        0.091199         -1.719          0.086\n",
"L5.urea_ee_bulk      0.004159        0.002303          1.806          0.071\n",
"L5.maize             0.020487        0.008487          2.414          0.016\n",
"L6.crude_brent       0.022428        0.013209          1.698          0.090\n",
"L6.soybeans          0.002044        0.002577          0.793          0.428\n",
"L6.gold             -0.004226        0.002184         -1.935          0.053\n",
"L6.silver            0.104285        0.054542          1.912          0.056\n",
"L6.urea_ee_bulk     -0.002649        0.001487         -1.781          0.075\n",
"L6.maize            -0.008036        0.005692         -1.412          0.158\n",
"===================================================================================\n",
"\n",
"Results for equation urea_ee_bulk\n",
"===================================================================================\n",
"                    coefficient       std. error           t-stat            prob\n",
"-----------------------------------------------------------------------------------\n",
"const               -7.638535         3.674331           -2.079           0.038\n",
"L1.crude_brent       1.563787        0.317696          4.922          0.000\n",
"L1.soybeans          0.139955        0.062059          2.255          0.024\n",
"L1.gold              0.074409        0.052764          1.410          0.158\n",
"L1.silver           -4.409772        1.329050         -3.318          0.001\n",
"L1.urea_ee_bulk      1.112425        0.037201         29.903          0.000\n",
"L1.maize             0.329777        0.141198          2.336          0.020\n",
"L2.crude_brent      -1.250799        0.515396         -2.427          0.015\n",
"L2.soybeans         -0.071260        0.086335         -0.825          0.409\n",
"L2.gold             -0.086168        0.085364         -1.009          0.313\n",
"L2.silver            7.401289        2.213736          3.343          0.001\n",
"L2.urea_ee_bulk     -0.327856        0.056839         -5.768          0.000\n",
"L2.maize            -0.434760        0.206913         -2.101          0.036\n",
"L3.crude_brent       0.861473        0.534029          1.613          0.107\n",
"L3.soybeans         -0.116643        0.086203         -1.353          0.176\n",
"L3.gold             -0.005424        0.086899         -0.062          0.950\n",
"L3.silver           -4.046644        2.372186         -1.706          0.088\n",
"L3.urea_ee_bulk      0.142202        0.059297          2.398          0.016\n",
"L3.maize             0.233880        0.209353          1.117          0.264\n",
"L4.crude_brent      -1.559052        0.535514         -2.911          0.004\n",
```

```
"L4.soybeans        -0.052667       0.086974       -0.606       0.545\n",
"L4.gold             0.003892       0.085593        0.045       0.964\n",
"L4.silver           1.032326       2.377877        0.434       0.664\n",
"L4.urea_ee_bulk    -0.104196       0.059613       -1.748       0.080\n",
"L4.maize            0.028888       0.208793        0.138       0.890\n",
"L5.crude_brent      0.913930       0.523894        1.744       0.081\n",
"L5.soybeans         0.095496       0.087667        1.089       0.276\n",
"L5.gold             0.053301       0.084266        0.633       0.527\n",
"L5.silver          -0.500818       2.247152       -0.223       0.824\n",
"L5.urea_ee_bulk     0.156414       0.056754        2.756       0.006\n",
"L5.maize           -0.115267       0.209116       -0.551       0.581\n",
"L6.crude_brent     -0.415228       0.325475       -1.276       0.202\n",
"L6.soybeans         0.089368       0.063486        1.408       0.159\n",
"L6.gold            -0.040869       0.053816       -0.759       0.448\n",
"L6.silver           0.599056       1.343913        0.446       0.656\n",
"L6.urea_ee_bulk    -0.119322       0.036643       -3.256       0.001\n",
"L6.maize           -0.020236       0.140241       -0.144       0.885\n",
"===================================================================================\n",
"\n",
"Results for equation maize\n",
"===================================================================================\n",
"                    coefficient       std. error           t-stat            prob\n",
"-----------------------------------------------------------------------------------\n",
"const                4.356950         1.103114            3.950           0.000\n",
"L1.crude_brent      -0.075264       0.095379       -0.789       0.430\n",
"L1.soybeans         0.036037       0.018632        1.934       0.053\n",
"L1.gold            -0.023696       0.015841       -1.496       0.135\n",
"L1.silver           0.588077       0.399010        1.474       0.141\n",
"L1.urea_ee_bulk     0.037550       0.011169        3.362       0.001\n",
"L1.maize            1.141848       0.042391       26.936       0.000\n",
"L2.crude_brent      0.036084       0.154733        0.233       0.816\n",
"L2.soybeans         0.007586       0.025920        0.293       0.770\n",
"L2.gold            -0.015226       0.025628       -0.594       0.552\n",
"L2.silver           0.911243       0.664612        1.371       0.170\n",
"L2.urea_ee_bulk    -0.040754       0.017064       -2.388       0.017\n",
"L2.maize           -0.309322       0.062120       -4.979       0.000\n",
"L3.crude_brent     -0.075868       0.160327       -0.473       0.636\n",
"L3.soybeans        -0.025177       0.025880       -0.973       0.331\n",
"L3.gold             0.066343       0.026089        2.543       0.011\n",
"L3.silver          -2.363728       0.712182       -3.319       0.001\n",
"L3.urea_ee_bulk     0.030562       0.017802        1.717       0.086\n",
"L3.maize            0.156905       0.062852        2.496       0.013\n",
"L4.crude_brent      0.153469       0.160773        0.955       0.340\n",
"L4.soybeans         0.021164       0.026111        0.811       0.418\n",
"L4.gold            -0.055764       0.025697       -2.170       0.030\n",
"L4.silver           2.024847       0.713890        2.836       0.005\n",
"L4.urea_ee_bulk    -0.022652       0.017897       -1.266       0.206\n",
"L4.maize           -0.136153       0.062684       -2.172       0.030\n",
"L5.crude_brent     -0.109997       0.157284       -0.699       0.484\n",
"L5.soybeans        -0.026489       0.026319       -1.006       0.314\n",
"L5.gold             0.052825       0.025298        2.088       0.037\n",
"L5.silver          -0.829437       0.674644       -1.229       0.219\n",
"L5.urea_ee_bulk     0.017161       0.017039        1.007       0.314\n",
"L5.maize            0.000944       0.062781        0.015       0.988\n",
"L6.crude_brent      0.026482       0.097715        0.271       0.786\n",
"L6.soybeans         0.002271       0.019060        0.119       0.905\n",
"L6.gold            -0.023655       0.016157       -1.464       0.143\n",
"L6.silver           0.146935       0.403472        0.364       0.716\n",
"L6.urea_ee_bulk     0.000775       0.011001        0.070       0.944\n",
"L6.maize            0.020945       0.042104        0.497       0.619\n",
"===================================================================================\n",
```

"\n",
"Correlation matrix of residuals\n",
"   crude_brent soybeans   gold   silver urea_ee_bulk   maize\n",
"crude_brent   1.000000 0.256931 0.111776 0.209142   0.153268 0.241812\n",
"soybeans   0.256931 1.000000 0.082179 0.111588   0.032578 0.473719\n",
"gold   0.111776 0.082179 1.000000 0.722123   0.072033 0.086465\n",
"silver   0.209142 0.111588 0.722123 1.000000   0.069879 0.125813\n",
"urea_ee_bulk   0.153268 0.032578 0.072033 0.069879   1.000000 0.017836\n",
"maize   0.241812 0.473719 0.086465 0.125813   0.017836 1.000000\n",
"\n",
"\n",
"\n"
]
},
{
"data": {
"image/png":
"iVBORw0KGgoAAAANSUhEUgAAA/sAAAKoCAYAAAAoFe/XAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnNpb24zLjcuMiwgaHR0cHM6
Ly9tYXRwbG90bGliLm9yZy8pXeV/AAAACXBIWXMAAA9hAAAPYQGoP6dpAACGyklEQVR4nOzdd5wU9eH/8ff226vAAQcIAiIgBmygFAtYAmLUGwJ
BiWisUKIGJUYBVHBEiyJscREIIr9Fw1GLBgjFoqK8rWhIYqFAIKUO7i2bX5/7O7czO5egzv2bu71fDz2MTOf+cxnPrMD3L35THEZhmEIAAAAAAAA4hjvbH
QAAAAAAAAE2LsA8AAAAAgMMQ9gEAAAAAcBjCPgAAAAAADkPYBwAAAADAYQj7AAAAAAA4DGEfAAAAAACHIewDAAAAAAOAwhH0AAAA
AAByGsA8AQB1+8pOfKBgMaufOnbXWOffcc+Xz+fTdd99JklwuV62fSZMmpW3W/5ptv6uyzz9Z+++0nv9+voqIijRgxRgQvfff7/Ky8vNer169ZLL5dKoUaMy9uuNnVnd
/ubuZ/XX3+9zuN6/fXXa+3jmWeeWd/X0uotX75cs2bs2bNqvO8AgDQmnmz3QEAAFqyyZMn67nnnntNjjz2myy67LG19aWmpnn32WZ166qkqKSkxy88880Nnz49r
X6nTp1syzNnztTs2bM1YsQI3XTTerTp48qKirMMPqf//xHd911l1m0gcPl1lm0yDPH+tbDDz+swsJClZWVVNfj45syZo+OPP95WVlxc3ODtW6vly5frx
htv1KRJk9SuXbtdwcAgCZH2AcAoA5jx45Vt27d9PEMfz4cPBNfj4+vbtW28/90RlRsC53ThxYfjLExkXbNc1p3WR0mFXu6+VaA
uijj/Twww/rlltuMcu/+OILvFHG7rwwgv10EMPNfj4+vbtW28/90RlZaVycnJsxwQAAPYdLuMHAKAOaOZrs59/Ev/6GQoiHqYBNTCnD
6t///4BoNq166dDJnkEN1zzz2Spk3v/mNJKlB8+K2KoZXhFs7X1k/5Ixj8gXBBOiAHN2EtZ8K2yodaWq//VWevezxx1QUKDTTjtNtNTz75ZMZ++Xw++FUUB+GRnnY6r++6aV5Grnmytr8Xvl16ba/ST
SlHuZDfaQq5upAC/WE8NaUtObl9FB5xbG8ZZHA2ZpexbNoY7OjZVgCz8n69WzoA6c7pyPeY9OLLcw5Zmp3aHdv/d+7xCqiVbBEBYfJdc8Bo6Ev6pUlaGbNy5Ho
4yqb0L8vtMfN+bHmdZH2W0zTkzbDJzsvh4nofDvNtXeTMKu5SjpSZjP0I0b3B98bnhs2lSbDUdFmR2ZW0fmpZLqsYixs0PRLWyFgtlHdeTaNtAAW/+KcPJL1lbq+Ue4
z0RdwOsa52KtGFnPLAqLD9QdFzC2Ks5YVGHZxDcGkA0axfKAKxLpMz8tuC8qqTbzNsWR22d1UR+1X7lbp4bd9ltAHQBcqYPYjlUjL2Jr58XNpdZn0q7cZv4cjNcOqhZ/VQ4
VEa8zEq0SiHk4jwZRwtJSVMrcpaW1/uCLNZwTcnS6XqRa/3unFmjWffGTF2L0LDbMF2fbLN3o7UqIwhhpg62qSWNxFhq4XdkeD6XygQBqTcGEN+bXAlrmJ/ctx7
4dq6jzPo9nSPZKfW4Gq4JlLbNkUFDTmXl+wfbxa7+pZm2MLONaDiBcFJOHjc/1/Tqk4qYGu4qlXbz6u3l+0FlUx7l2KKXMK4RuA0uqBMQbXzkRTsl7Y9DWRljI
tT91l2rfxMgnfl1NJV7v1pZbqxcZ2TR2wnGpyfzlRMyzvlgpnzjWfKPujbgoVnRqW9dE+QBmugkNMJCSpdHp6vFvS3vmlYEdB27qaUAX0yAUfK9btiUhKmgTN
Tk4dcc1lc7dWvaZvxQyBx9m+pkafc2vtDbcZ6TupPeZmCWCTXOs4YBNxs7DTWB6kcLJQQeWjNZuk4Kmb4b9k7dFrYxZnz+7jJszxtYqL3CbJaZ8kiyY/pDq
fHZ0++PsixSX7T8c3tuj9bTe1/qfflWo16aKofBSsiQdZkbc0C8CJ+Pd0bdqUf8ZLMS39aZT5+ikjCs8yMAP3jfdyf+B4l+YbmVpkA0Wf9pGZO97IVG8EIGNuV+
I3aLdLZMVD0Ws+ePpIM6TOyrctLsMVPdibG2M6IkQFyRSrbsWcLexe+Gy0TUD4O8RhNUVbrI0bdWgmAi8IVxa4Ws2OobSGh5HlfPhrAwaFh1UGdQD5ZCTzTfa
Ed5oH4vj5XOTKLSs/CWNJJlhNkHBMjnP9T6qkbm6BIRrL+KvBhYk88o6RFS6HQ0Ix8HcaKmtkcNR6RaZ/RlO89jkVQkVuQF5dcJL3YK2WATngzV6rPB+TEI+j5KsRsK
bDwsVbXvFvGjWqnUpv9BVyzWJSv5pfPRbB/xSBPMmlMBsBYQA5gmmz/oiZo5WQ2JFVRa+Z/T6jOUbOz/cMTHGqkkAoAl8QA8YTE7M8FLh6RIJbm+s/eu4lk/Exb
1VHmsxMqQp2p2A0JF2CNvIqmtvz1FvYuK0ur/QGsGPoVvr9htb2fP91qrjRp2nfPM19ocbfuhvh4+HrXGT1qnJsuV8v5lrUTV4BaXjyZQQ3UxzXjuyXtkmvj0rvbcb
O5Qx+EPV3aE7s+wu8m/DifSefJgLVTeZTqhbyP4qa1QYf6+dv0KyvOV9pZMfCP4fbf/z6CyXhWf4NvcDQlX5wDdKPXnqGqUIXbwJMfS9m18K2YqlNp/hY9j0s4f
fgDzYn8ovtxqj83HuIBT6whyC0hlXdu91g2TugMP8v2wAEvIMdmbj+QO3ePHx3RyPSfOH0tVYS+9Z9ozeAIX3aqhu3F6cWPY7D6iRWyOj46N3R4axwF0L0N0hzH8
mTeGnG3ROhY31s7SYVQPa4VqqAGRr6s4y+6QBbkoH7azVT/PaaznKJ5KdMsZ8jG+OnzC89NCnIJpHdbtsU2fXTZKLRvGc+S8BGJkJm+ECiyyWMAVmHs2j/BGHV
iWl7VzLHPqKxkJQLY6GWXrzt9E8srXWK0JetBwimR9Xf3I3xMJ6GjFKQmGCwvYv5uCDOl6zVcuT3v2HDKpJbHfu9hzDRG4ZrvXbsHPqT9o0r2IV4WCNbYTYcS9lmml/w
JGsyVmDj6eZKNHZGhJQH8bKhbJY5W2gJjwDcA2WJ+kCa9Zd7c+9Ipa9aaa4DCI6ofCFlqGZNC+U2PQwlkYRO4a7ZPhjkhaAhwnPFT8MVLR9Y4Rn9TtXOk2qNXk4G
44/3/0m9d1ZDnQoRhnPPo4qwsx9cJXqbixvSM6FMP+J7ySA8DoFVAmvfm74hvYHr9cwrZtTZkHh9xK/ZYyvw2/qjNq4R+S2Df9bbWyWv2WeW/CVn6VwSpTb9g7TVS8g
GpRenu0o3pGUHW1fHsNj0YR7uVvfk/ocA4jM4NaBbNvqaaHlbM+hcgzSOfR4a9VK1j2sQ6uXBfDC7XzvtGm36LV3ZVqNYlONRFVcEZmXzCL8e8Vq6o2w0/0/wVZwR

qa9eukqTBgwfL5/Np6dKlOvvssyVJmzZt0scff6zbb79dkjR8+HCVlpbqnXfe0VFHHSVJWrVqlUpLS83/EEgVCAQUCPAqEwAAWhPrE9yTo+VmQLcsW0N6eb
hcleFKc95cn6jb1Je3S5LX5a0J5SmfXG8t5ZYQn7zUPc+Xp6A3yCg6AKDZNepp/Jdddpkee+wx/eMf/7A9JK+oqEjBYFC7d+/WrFmzdMYZZ6hr16766quv9Nvf/l
bffPON1q5dq4KCAknSpZdeqn/+859asGCBOnTooKuuukrbtm2zvXpv7Nix2rhxox588EFJ8Vfv9ezZs8Gv3mtJT0EEAMApIrGIKiIVaQE8OW8Gd8tIuvWS94pw
xT4ZOXfJpVxfrhnEU+fNEJ5YTo6Y2wJ9MqT78uR3+xlBBwDUqyXl0EaF/dp+yM2fP1+TJk1SZWWlxo0bpw8++EA7d+5U165ddfzxx+umm26yXVJfVVWl3/z
mN3rsscdUWVmpE088Uffdd5+tzvbt2V16lQtXrxYknT66afr3nvvVbt27RrU15b0JQMAkA3JB8LZgnekIuOl7bXVsY6sV0QqVB2trn/He8DtcivXm2sL4rm+mm
VbUPfaQ3nyknjrSHuON4d3ogMA9rmWlEMbFfZbk5b0JQMAUJtoLKrKSKWqolWqDFeqMloZX45U2abJT7JeVTSl3FIvGeSb44FwSV631x7CE0HdNpJuWU4
dXU8G+qAvyL3nAADHaEk5dI+fxg8AQFsRiobS7yePVKQF7fqCedr6SJVCsdA+OYagN2gbOc8YzlMuc89UJ3npu8/j2yf9BgAAe4awDwBwlJgRU1WkKuNl6
mmXr2e497wyXJlWJxKLNHu/XXIpx5ujoDdofnI8OQr6ElNvMH19YjlZL+gJmpe0W0M6D4QDAKDtIewDALLKMAxVRavMB7bWksl5lJ
OZ4cM0Qnp2bg9ubYlzMEdGvdoLcmnOd4cxTwBLiMHQAANBnCPgCg0ZLvMjdDeKRc5aFy+zvLI+XaHdqtikiFObWu3x3ebQb05riv3CWX/VJ0y4PfkveJp91
vbrlMPfWe86A3KK+bH5sAAKB14LcWAGhDkk9n3x3erbJQmXaHdmt3aLfKwvH5XaFd2h2XalRbKk5e5W4N8zIg1ef+sT2FPvgot7Z3llpAe9AVtT2a33pPOA98
AAEBbRtgHgFYkEouoFMZ1qX3CYcCzdZf1xyKd+XbwZs63yJ9v65Hwy1QV6VBgAA0EQI+wCwDxmGYYbwsuoylYZKVVZdFl9OlGUK7
cmgXhGpaJJ+uORSvj9fhf5C5fvyle/PV4G/AW+AhX4C4+Jh3J+fPsruzzMvc08GdEbPAQAAWh7CPgA0UvJ+9bJWOW/c8MmbmXoN0G9bQmUqrSB2QbluuLcyHyprk8vegN6h8Xzyg5/v
zzZCeGtpTA31yPteXyyg6AACAgxH2AbRZVZGqtMCeGs5Tw3tyGjH27lVsAU9Ahf5CFQWKVOgvjH8C8WmBv8AM8rUFeJ+bd5wDAACgdoR9AK1GzIjZ3
oduzic+yQfHpb6qrbZt9vbeda/bawZ1M7QHClXkLzKDe22BPseb00tTfCgAAAJCOsA+g2SSf/J4M3Oar1iIZQnry9W1heyC31q2MVDZ5H90ud52hvK4y7lcHAA
BAS0XYB2ATjoXrHTGvLYxnCu3N8f50j8tje2d6ba9lSy23vqItz5dnPoiOe9cBAADgNIR9wCHC0bC2V23X9qrtKguVZQZqthCfckl7cj4UCzVL/4LeYN1hPCW0
p73CzRLUA54AI+oAAABAHQj7QAsVioa0VqWvq7dlTTMEP8Od43d3N+n+fW5fWuDOFWO9eXuDO9eXaXrtmDeqp71a31g16g/K4PU3aPwAAAAC1I+wD+0h1tN
oWztMCfOV2ba+uuWWQ/DDnDO9eXaXrtmDeqp71a31g16g/K4PU3aPwAAAAC1I+wD+0h1tN
ZpqcT64vDBRy/zkAAAAAwj6Qqjpara0VW/V95ffaWrlVWyq2xOcTZVsqt+1o3pHo9vzurxql9OuJrDXEeA75HRQgb+A8A4AAACg0Qj7ADaMqwbVVmMcHT78azuVqEP3K
9auVVbK+LT7ytqAvzWyq0qC5U1uE2vy1szsm4J6e0D7dUh2MEM88nyQn8hD5YDAAAA0OwI+2jVDMNQeg8Htorv7eNwltD/feV3zfqAXY+t0+dczurY7Cj
OgU7qWOwY81ybid1CnZSp9xOahdox8g7AAAAgBaHsI8Wb0fVDv1353/15c4v9VXZV7YAv7VyqyojlQ1uK+gNmgE+GdptAT5Rzgg8AAAAgNaMsI8WY3v
Vdn2x8wt9sfOLeLgv/VJf7PxC26u217ttni/PDOrJMG8bmc/tqM7Bzsrz5RHiAQAAADgeYR/7lGEYNaG+1BLsd35Z5wPv9svfT33a9VHvwt4qyStJG5nP9eXuw6
MAAAAAgJaNsI9mYRiGtlVts4X55Gj9zuqdGbdwyWWGeuund2FvjwAAAAANAJhH3vFMAx9X/m9OUpvfkq/UGl1acZtXHKpe0F39Smyh/pehb0I9QAAAA
DQBAj7aBDDMLS1cqstzZ3a1OsUn8Un3Sjoo9Sj/o9PaHaAD2x2oA4ri015FvRT0BvfxEQAAAABA20HYh41hGPqu4jofNL+2h96RFaFdqVcRu3yx0P9Ykwnz2vQ
p7Kcebs4+PAAAAAABA2G+jwtGwvt31rb4s/VJfln9p9aXrzWltr7Jzu9zav2B/HVB0gHnp/YHtDlTPwp6EegAAAABoQQj7Drc7tFvrS9frdfdl6fbmzJthv2LVBESO
ScRuvy6sehT1qQn3i3vpeRb0U8AT28REAAAAAABqLsO8AyYYfkZRql31Kxpdbtcr256l3UWwcUHVAzbddbPQp6yOf27cMjAAAAAAA0JcJ+KxKJRfS/3f+zjd
AnP7vCme+nl6SOwY62UJ+cL8ktkcvl2odHAAAAAADYFwj7LVBFuEJflX1lG6FfX7peXX5d9rXAsnHEbt8ut7vndbYE++SkKFO3jIwAAAAAAZBNhP8s+3/65
Pvz+Q325sybUbyzfWGv9HE+OehX1so3QH1B0gPYv76o1+6CNmUDhs/7dRYLaMw/4Xa1/3M0jT2y2d+VsmNXJaVZYJaG8L92T+hYvGFr0bgIjnjm9M0Nb8QgSe3/7XlDkEAAkD1fJ+/bv72J9KGLP2h++4nBDrPc6NDJdBgdBAmr0XtNqQ6A7r5wHKQcYUOHnCjPVc54UqtbmbHkuQyGxgDbACjoFfNP8Q3+AAAGUDA12R
qdaRlwExQdH9RYJr71WA9IRebk8dOo9WR4VwDZuz7BjaK0pkK70+Qz0ykZ3GRy5zVwZJuT6iyB8Z1vBOsnKdWeZZoJh8Bp+sdsyKzBWOKPWO8rz/lnz3A0/wL0cWY+08GHwjhWeL2drW/YVbe6LY+YP3YiKTbgZ1k7WRa6BaLGYVaiXHTMdG83uHS
5cStEGLymfLw7PMBKvKSDZxn/lDk+YMMOGlt3pGXp05xlLGsnwmiDaJ0iO3kjWzHlQaW3tSxXHjycXQ2XTMMP7+i5sUZ2o7vqxX/X3XmMYJBR7qXGYiZv1jqXnQr6xCnGtvZqBNWGMmLuX8BgFCpBTc6Pq7m+Cwcc2k0gaijvtNq3rtdy/rtmsBJMgrs
8UfTmdLbTDjCQJPYXnHhmlPUvkrPtzNHK8U7pphJXW8YIZTZm3JoZaVCJdaIXwBrdzsXnaK/kf4ddmtm/v6TbhivHb2Ow2aC7XPqZD99R1bgNZCDHj9rh4bRktcqIO1ywk5KmjPTrtBp3nKyxrI+UUl8dGVzQWX5WPHhbXTrxMd8SMh9ypgJ/qEvyhWoTs
DMAlSHbo5oEJDHwZuqzeuWNGWNOMK6Nmv+Fk0GLMsCUvU4BbQS2rMYE4+V3uSY3BKyoQhcEORPSeJ4DtunDLHnn6Yt+nn2dvYR/4bH4jI6+jaBhbRljm2JeBxMwIKlvYTkdTJgq+umWQoIPqBhoKSfXQuUw+dAufKbHmC7qAzbbvz5mtDo
ZHG4Srge8kB8dp0dvhcfWqk5z+3qQpHxTQOvEBlDdNX8MkBWHNhYnXQQ3xp+XavXXw8rMrfZX5wC8L0cVi0ojk7iJt5Xz3TMc1o0cIWyXpjH00bpaq9q6czDlOqu1aX8fvNsRbQOOCLWEZm
xnBSh+Bv2G09OxFzDkXOVt2kfzNkY0D1n6Zo8NdLwv2bXyXh8KrF4HOQqH+uMmFaSzI6Zld7Kfv5fZ0b4sqWlU1OfOztdrnL8KiWFntNy6kVbgdkXxSdfUTa2sDB0UPqaZj6mZbsa3y3k5FB
jBYY4fk4xm0Do6xUUL9B7GJl5Cm8bz4AN1I40yoJOeVqyAi+y3t7nWH7TjkX2pmE9T0l5iCnkgQkjaLd0ShCNq9R00B0HSqlKnrZpzN2D8BfZhJk2QHgPZCqW1kc2KswKzEKj0eJqBCLN9G4
GFRc0kfGzQHCmMMWwCFUd7PfmTVHxH7YoG7L3NKsTYQPlw3whEAMn8O0fP+jbcfR3NgCHkqEqH+FQsW2AxOV2mb3Uq1pW/FpH+hLZtDBGakXbl04UuaiXBlO8QEzFlJNa6SaUSP9TC76zd

OEJwHmMb6rBWWfvfBXrGkSNBUYPxQZk61NeMpFiQVJ4S

FRWZdVJVV1errKzM9gEAAAAAoC1q8rA/duxYLVq0SK+99prmzZund999VyeccIKqq6slSZs3b5bf71f79u1t25WUlGjz5s1mnUz/I9S5c2ezTqq5c+ea9/cXFRU
16hUMAAAAAAA4SaMv46/POeecY84PHDhQQ4YMUc+ePfXCCy9o/PjxtW5nGIZcLpe5bJ2vrY7VjBkzdOWVV5rLZWVlBH4AAAAAQJvU7K/e69q1q3r27
Kl169ZJkrp06aJQKKQdO3bY6m3ZskUlJSVmne+++y9tra1bt5p1UgUCARUFto0Rc0e9rdt26Zvv/1WXbt2lSQNHjxYPp9PS5cuNets2rRJH3/8sUa
MGCFJGj58uEpLS/XOO++YdWtWqXS0lKzDgAAAAAyKzRYX/37t1as2aN1qxZI0lav3691qxZo2+++Ua7d+/WVVddpRUrVuirr77S6+/rtNOO000dO3bUT3
7yE0lSUVGRJk+erOnTp+tf//qXPvjgA/385z/XoEGDdzKfzDxgwQCeffLIuuugirVy5UitXrtRFF12kU089lSfxAwAAAEAz6tWrl+6+++5sdwN7qdH37L/33ns6f/vjjze
XkffLnn3++7r//fn300Uf629/+pp07d6pr1646/vjj9eSTT6qgoMDc5q677pLX69IEz2O5DAqVZZ9MjEfTkZP8HTGiRZ6J4TT2d9577x4fKAAAAAAA
bUWjw/6oUaNkGEat619++eV628jJydEf//hH/fGPf6y1TocOHfToo482tnsAAAAAALR5zX7PPgAAAACgaT3zzDMaNGiQgsGgiouLddJJJ6m8vFyxWEyzZ89W
9+7dFQgEdNhhh+mll14ytzvhhBN0xRVX2Nratm2bAoGAXnvtNbNs165dmjBhgvLz89WtW7e0gdrS0lL98pe/VOfOnVVYWWKgTTjhB//d//2eu/+KLL/TjH/9YJSU
lys/P15FHHqlXX33V1kavXr00Z84cXXDBBSooKND++++vP//5z+b6UCikK664Ql27dlVOTo569eqluXPnNsn31xYQ9QgEAAAAgyTCkUHl2PnVcQW21adMm/e
xnP9MFF1ygtWvX6vXXX9f48eNlGIbuuecezZs3T7///e/14YcfasyYMTr99NPNt6NdeOGFeuyxx1RdXW22t2jRInXr1s12u/Ydd9yhQw45RO+//75mzJihX//61+ZD
1g3D0I9+9CNt3rxZS5Ys0erVq3XEEUfoxBNP1Pbt2yXFn/V/V2yimn6NVXX9UHH3ygMWPG6LTTTM333xjO5Z58+ZpyJAh+uCDD3TZZZfp0ksv1WeffSZJ+s
Mf/qDFixfrqaee0ueff65HH31UvXr12uNT29a4jLquyW/FysrKVFRUpNLSUl7DBwAAACBNVVWV1q9fr969eysnJydeGCqX5nTLTod+u1Hy59Vb7f3339fgwY
P11VdfqWfPrZ1++23ny6//HL99re/NcuOOuooHXnxkkfrTn/6k6upqdevWTffff7/OPPvsSdLhhx+ucePGaebMmZLiI+4DBgzQiy++aLbx05/+VGlZVqyZIlee+01/e
QnP9GWLVsUCATMOgceeKCuvvpvq/fKXv8zY7x/84Ae69NJLzSsLevXqpWOVaPPPKIpPh/InTp0kU33nijLrnkEk2dOlWffPKJXn31VblcroZ8g/tExj83CS0ph
zKyDwAAAACtyKGHHqoTTzxRgwYN0llnnaWHHnpIO3bsUFlZmTZu3Kijjz7aVv/oo4/W2rVrVrJUmBQEA///nP9fDDD0uS1qxZo//7v//TpEmTbNsMHz48bTnZx
urVq7V7924VFxcrPz/f/Kxfv15ffPGFJXKm8vFxXX321Dj74YLVr1075+fn67LPP0kb2DznkEHPe5XKpS5cu2rJliyRp0qRJWrNmjfr376+pU6fqlVde2ctvrm1p9AP6
AAAAAMCxfLnxEfZs7bsBPB6Pli5dquXLl+uVV17RH/N1223ad2ad6dfJ4PDryyCO1ZMkSud08eq4heBo/AAAAgDaprqeqO923336rX166d1339URRxyR7e60Kq3lafy
XTpL0m9/8Ri+//LJ+//vf68ADD1QwGNSZZ56pUChkMknTEEUdo/fr1evHFF/Xxq6/q7LPP11kkknnaRnnnmm3r6CsA8AAAAANVyuBl1Kn20ul0tH
H320jj76aN1www3q2bOn/vWvf6lbt2566623dNxxx5l1ly9frqOOOSpcHjRokIYMGaKHHnpIjz32WMa3pK1cuTJt+aCDDDpIUD+GbN2+W1+ut9R76N998U5Mm
TdJPfvIT5F+L/66qtGH2dhYaHOOeccnXPOOTrzzDN18skna/v27erQuUOj22prCPsAAAAA0IqsWrVK/rXvzR69Gh17txZq1at0tatWzVgwAD95je/0cyZM9Wn
Tx8ddthhmj9/vtasWaNFixbDC31wzkVm+//bZuv/12jRs3TkuXLtXTTz+tF154QZJ00kknafjw4Ro3bpxuu+029e/fXxs3btSSJUs0btw4DRkyRAc
eeKD+/ve/67TTpPL5dL1119vjtg31F133aWuXbvqsMMOk9vt1tNPP60uXbqVYVw+gboR9AAAAAGhFCgsL9cYbb+juu+9WWVmZevbsZnevsqXnz5mns2LEaM2aM
ysrKNH36dG3ZskUHB3NkVxrjH6k5p06ZpwQPQJaQ+Zk6K6Tp06dr9ePXp6N9+XKZvKmCtHDWdkfTZgvsGbmdgixanb/8Ko3PgHYAg46aw3bDeQbxqHHv/u2tbQCt
5KSEknxoH7BBRdoxlgR6tixo6655hqVlZU16jjz223ad26dfJ4PDryyCO1ZMkSud08eq4heBo/AAAAgDaprqeqO923336rX166d1339URRxyR7e60Kq3lafy
M7AMAAABAGxEOh7Vp0yZde+21GjZsGEHfwbj+AQAAAADaiLffls9e/bU6tWr9cr9ADD2S7O2hGjOwDAAAAQBsxatQoOfRObqZgRZGZB8AAAAAAIch7AM
AAAAA4DCEfQAAAAAAAHIawDwAAAACAwxD2AQAAAABwGMI+AAAAAAAOQ9gHAAAAgDZmwYlFateuXZ11Zs2apcMOO2yf9drAdNj7APAAAAAI
DDEPYBAAAAAHAYwj4AAAAAAtDK7du3Sueeeeq7y8PHXt2lV33XXWRo0apWnTnTpkmSduzYofPOO0/dq5pXjMpqSkhIVFBRo8uTJq
qq2gdHguZC2AcAAACABMMwVBGuyMrHHMlwW9uTExPVsOu+gMnNpq9+vXZx7Tue5+Wfpee+LFy/WihUyrYl/cr/+dPP32PxdNV2u9/6d6W/5ycq5s6c
eUWvffee+ratavuu+XDvgBAY+Ru89pv2AfWXj7QsRxdAxjw0ZDU5VRfvJ56mr3r+T6Vt9KaxRz+rJCAcs9SEGmc+ssYyTCeuZvtRM...

+uU089VatXr5bH45EkTZgwQRs2bNBLL70kSfrlL3+piRMn6vnnn9+b4wUAAAAASZLf75fb7dbGjRvVvqVMn+f1+uVyubHcLLZRhGAqFQtq6davcbrf8fn+2u1
Qnl2EYxh5v7HLp2Wef1bhx4yTFD75bt26aNm2arrnmGknxUfySkhLddtttuvjii1VaWqpOnTrpkUce0TnnnCNJ2rhxo3r06KElS5ZozJgxWrt2rQ4++GCtXLlSQ4cO
lSStXLlSw4cP12effab+/fvX27eW9H5DAAAAAAC1TKBTSpk2bVFFRke2uoJXIzc1V165dM4b9lpRDGz2yX5f169dr8+bNGj16tFkWCAQ0cuRILV++XBdffLFW
r16tcDhsq9OtWzcNHDhQy5cv15gxY7RixQoVFRWZQV+Shg0bpqKiIi1fvrxBYR8AAAAAA6uP3+7X//vsrEokoGo1muzto4Twej7xeb6u4AqRJw/7mzZslSSUlJbb
ykpISff3112Ydv9+v9u3bp9VJbr9582Z17Tw5rf3OnTubdVJVV+erurraXC4rK9vzAwEAAADQZrhcLvl8Pvl8vmx3BWgyzfI0/tT/5TAMo97/+Uitk6l+Xe3MnTvXf
JhfUVGRevTosQc9BwAAAACg9WvSsN+lSxdJSht937Jlizna36VLF4VCIe3YsaPOOt99911a1+1u3bk27aiBpxowZKi0tNT/ffvtXh8PAAAAAACtUZOG/d69e6tL
ly5aunSpWRYKhbRs2TKNGDFCkjR48GD5fD5bnU2bNunjjz826wwfPlylpaV65513Vq7DqrVq1SaWmpWSdVIBBQYWGh7QMAAAAAQFvU6Hv2d+/erf/+97/m
8vr167VmzRp16NBB+++/v6ZNm6Y5c+aob9++6tu3r+bMmaPc3FxNmDBBklRUVKTJkydr+vTpKi4uOcOHXTVVVdp0KBBOumkkyRJAwYM0Mknn6yLLrp
IDz74oKT4q/dOPfVUHs4HAAAAAAEA9Gh3233vvPPR1//PHm8pVXXilJOv/887VgwQJdffXVqqys1GWXXaYdO3Zo6NCheuWVV1RQUGBuc9ddd8nr9erss89
WZWWlTjzxRC1YsEAEj8ess2jRIk2dOtV8av/pp5+ue+9d48PFAAAAAACAtsJlGlaR7U40h5b0fkMAAAAAAgPO1pBzaLE/jBwAAAAAAA2UPYBwAAAADAY
Qj7AAAAAAA4DGEfAAAAAACHIewDAAAAAOAwhH0AAAAAAByGsA8AAAAAgMMQ9gEAAAAAcBjCPgAAAAAADkPYBwAAAADAYQj7AAAA
AAA4DGEfAAAAAACHIewDAAAAAOAwhH0AAAAAAByGsA8AAAAAgMMQ9gEAAAAAcBjCPgAAAAAADkPYBwAAAADAYQj7AAAAAAA4DGE
fAAAAAACHIewDAAAAAOAwhH0AAAAAAByGsA8AAAAAgMMQ9gEAAAAAcBjCPgAAAAAADkPYBwAAAADAYQj7AAAAAAA4DGEfAAAAAA
CHIewDAAAAAOAwhH0AAAAAAByGsA8AAAAAgMMQ9gEAAAAAcBjCPgAAAAAADkPYBwAAAADAYQj7AAAAAAA4DGEfAAAAAACHIewDA
AAAAOAwhH0AAAAAAByGsA8AAAAAgMMQ9gEAAAAAcBjCPgAAAAAADkPYBwAAAADAYQj7AAAAAAA4DGEfAAAAAACHIewDAAAAAOA
whH0AAAAAAByGsA8AAAAAgMMQ9gEAAAAAcBjCPgAAAAAADkPYBwAAAADAYQj7AAAAAAA4DGEfAAAAAACHIewDAAAAAOAwhH0AAA
AAByGsA8AAAAAgMMQ9gEAAAAAcBjCPgAAAAAADkPYBwAAAADAYQj7AAAAAAA4jDfbHQAAAAAtBCxmBStlqIhKRpOTC3zkerM5bZPLesj
ddWpq91wfL+SdM367H4/rQhhHwAAAAD2JWugNgNwdUotgbravt8Mvza6oUyhOr62rTUS7ZpRLP97dTNMCSXK9u9aBUI+wAAAACcKRaNB9hIVSLkJg
NuLUG31iCdDNmhmu33tF40JMUi2f5mGs7tkzx+yZOcWua9/vQy23zAMp+y3huoY7vUj6UNNBhhHwAAAEDTSo5cJ0NupNoSqtqls2y1PKqxHZV8dBsC+sp
dW1lKXVbS6h2W8OvNUQHUgK1P1HPlwjSybbqBlADtT1mX0mZq3dQ2k+vdPsnNY95aK8I+AAAA0NoZRjzYWgObvRTbGowtI9qmK6uu8zWURiglZLEu
3xJDtckveHEu4tYbeRNC1hV7FXtSrL5injIpzWTqaAWEfAAAA2BvRSMpIdFh4xW4O1SwSwDtTeDjWhnwWiwkdDLsZmYxsfxO1SwZkbyAeuJPh2Fz2W4J4Th11LeVm
Xcv2aXUtZR6iD9oW/sQDAACgdUuOaocrUy7xti5XSeEqy7oqy6d679YbsWx/A+msl4Vbp9YwnFbmtwfl1DIzXPtT1mUqs7Th8TFyDWQBYR8AAABNwzDsI
dsM3029nCHMt5TA7famjDrXMWK9V3UylSWX9xnDYCwDwAA4FjRSDwQhy0f63Kkqv51jQnjkapsH3GcJyD5cizBOcfySSzXuj5R5gvaLwP3piwn19suRQ
9Ibk+2jx4AJBH2AQAA9i1bAK+IXxoerkgJ24kyaxhPC+31ravI7kPSXO6agGwG571dtgTy2sK6x8+oNgCIsA8AABC//DwasgTmCntotgbptGCeaZ11u2SgT8zHw
tk5Rm8wHpB9uYmwnJtYDibWJT7WdXsTzt1e7tMGgCwi7AMAgJYvFpVC5YnAXC6FKmrmw5WJdRWW+WTIrkjUTdarsId36wh7Nu75ri94Zwzitawzl62B3r
IdwRsA2hTCPgAA2HuGER/NDpVLod2JaXI+UzBPBOyMwTzDfDS0747F5ZZ8eTXh25ebIVjn1oTqBq/LTQnmBHAAQPMh7AMA0NYkR8ltgTwxMp4W1D
PN17K8T0bGXZI/zxKecyV/buZ5XzBR1zpv2c46mm4N5rwmDADgAIR9AABaMjOY766ZVlvmraG7elfDwnmksnn77MuNB2EtjE
aDgBAQxH2AQBoKoZRc2m6LZSXS6FdGUJ5pgCfUjjdc0Xz9dbklf35NMPfn1bPcgHlfLq8eAwCgBSDsAwAQDUtVZVJ1WTyIm9NdUlVphrLEvC2AJ0bZZT
RPH11uyV8gBVJDdmI5YJlPm6Z+EuWMkgMA4FiEfQBA6xWL1oTwKksYry7LUG62ZptaNVDV939JGvq2hPC8e3JPrAwX1BPj8+OvMCOYAAKCBCPsAgH
3PMBKj4dZAXktAT11nXR8ub9p+XKlQGE8fAcKpJzkfGF6uT8/XpZpVN2XK7ndTds3AACARiDsAwAaJxKqKcqb+bWYJ5x3S6pOnFZFM+ud2bYwnlyU
BeVDOfLM+xhvbU8F4gefixCAAAnIHfagCgrYlF4+/ehV+6QKncmpimfqtLMob2qTIpWN11fXB7L6HlReiBPHVnPSQ3qianX33R9AgAAcADCPgC0VtFw7WG
9rk9VqZrkIXL+/NpHzc1R9dTQXmP7L4g96EDAAA0A8I+AGRbuLLuUfa0T6QeaNfe76defLwXbXS8F2iWiny06EmuFtDe+qoOq9XAwAAaLEI+wDQFKyKyXxlft
TAnvO2spT5T5cPQcR0iJpZGc4pPQrwWnoCUQrVpmZQdAADaxDawQj7AJBkGPFGPFRdKtF6dvcKy6d7jCuK8Y746XxK
cupo+17+/A5f348sJuXx7ezj6anXjafLAsUcO86AAAAmgxhH0DLVd8D6Kpq/hTez/K3r72YJ6xLDHKzqXxAAAAaAEI+wCaX6iiAQE9dfR9594/gM6XlxLa2
9lDem4He1hPrvfnM8oOACAVo2wD6BxDEMKlUsV30vl26TyrYn57Uz2+LzZtn3UqRy7/ZZ2Ztn3UqRy7/ZZ3wPobIHdEty9gaY4YgAAAKDVIewDbZ1hSKHdiYC+zRLaL
WE9OV+RCPd78vR4t7eecJ4hrPMAOgAAAGCPEPYBpzHD+9b4yHumUXAwXczCfW70l49+ZIuR2lvGIpr1NiPvHJTZ0W8wA6AAAAYB8i7AOtRSwWD+N0
u7vktMN0u7v0uZbtmzy+a9OYnQXpwI7Zb53MRyMrjndZL8eYR3AAAAoIUi7APZFg3HA7otxGeYlm+RYpGcGt+sNZhhpL7aE9mSAL47PE94BAAAAxyDsA
80lVJEhuH+XPhJfsU2Nek1cbrGU30UqKKl9mtdZCuQ326EBAAAAaNkI+0BjRULSro1S6S6f+kso3Srk6o3ZLqD/Tqoua3ibLo+UX5Ie3PM7SwVd7CGe97gDAAAA
qAdhH7CKReNbvfR/UtmGxPR/UumGxPR/8fUNHYn35iRCfJcMU0uozy2W3O5mPTQAAAAbQdhH22HYcQvmbcG92SgT5bt2tSw++I9fqmwm1TYPR7cM
4b5kvhr47gpHGAAAMA+RtiHc1SV2kfizVCfmJZtbNgr5lweqaCrVLhfYtrdstw9fvDwnmksnn77MuNB2t/XnyEPC2MBxMj57WF8bxa6jE
aDgBAQxH2AQBoKoZRc2m6LZSXS6FdGUJ5pgCfUjdc0Xz9dbklf35NMPfn1bPcgHlfLq8eAwCgBSDsAwAQDUtVZVJ1WTyIm9NdUlVphrLEvC2AJ0bZZT
RPH11uyV8gBVJDdmI5YJlPm6Z+EuWMkgMA4FiEfQBA6xWL1oTwKksYry7LUG62ZptaNVDV939JGvq2hPC8e3JPrAwX1BPj8+OvMCOYAAKCBCPsAgH
3PMBKj4dZAXktAT11nXR8ub9p++XKlQGE8fAcKpJzkfGF6uT8/XpZpVN2XK7ndTds3AACARiDsAwAaJxKqKcqb+bWYJ5x3S6pOnFZFM+ud2bYwnlyU
BeVDOfLM+xhvbU8F4gefixCAAAnIHfagCgrYlF4+/ehV+6QKncmpimfqtLMob2qTIpWN11fXB7L6HlReiBPHVnPSQ3qianX33R9AgAAcADCPgC0VtFw7WG
9rk9VqZrkIXL+/NpHzc1R9dTQXmP7L4g96EDAAA0A8I+AGRbuLLuUfa0T6JeaNfe76defLwXbXS8F2iWny06EmuFtDe+qoOq9XAwAAaLEI+wDQFKyKyXxlft
TAnvO2spT5T5cPQcR0iJpZGc4pPQrwWnoCUQrVpmZQdAADaxDawQj7AJBkGPFGPFRdKtF6dvcKy6d7jCuK8Y746XxK
cupo+17+/A5f348sJuXx7ezj6anXjafLAsUcO86AAAAmgxhH0DLVd8D6KpqW7dTez/K3r72YJ6xLDHKzqXxAAAAaAEI+wCaX6iiAQE9dfR9594/gM6XlxLa2
9lDem4He1hPrvfnM8oOACAVo2wD6BxDEMKlUsV30vl26TyrYn57Uz2+LzZtn3UqRy7/ZZ3wPobIHdEty9gaY4YgAAAKDVIewDbZ1hSKHdiYC+zRLaL
WE9OV+RCPd78vR4t7eecJ4hrPMAOgAAAGCPEPYBpzHD+9b4yHumUXAwXczCfW70l49+ZIuR2lvGIpr1NiPvHJTZ0W8wA6AAAAYB8i7AOtRSwWD+N0
u7vktMN0u7v0uZbtmzy+a9OYnQXpwI7Zb53MRyMrjndZL8eYR3AAAAoIUi7APZFg3HA7otxGeYlm+RYpGcGt+sNZhhpL7aE9mSAL47PE94BAAAAxyDsA
80lVJEhuH+XPhJfsU2Nek1cbrGU30UqKKl9mtdZCuQ326EBAAAAaNkI+0BjRULSro1S6S6f+kso3Srk6o3ZLqD/Tqoua3ibLo+UX5Ie3PM7SwVd7CGe97gDAAAA
qAdhH7CKReNBvfR/UtmGxPR/UumGxPR/8fUNHYn35iRCfJcMU0uozy2W3O5mPTQAAAAbQdhH22HYcQvmbcG92SgT5bt2tSw++I9fqmwm1TYPR7cM
4b5kvhr47gpHGAAAMA+RtiHc1SV2kfizVCfmJZtbNgr5lweqaCrVLhfYtrdstw9/kA7RuIBAAAAtFCEfbR80YhUuT3+bvjdW9H5ZPLoV0Nay+vsz24JwN
9UY/4fH6J5OGvBgAAAIDWi0SDfS8Sil9OX/G9VP59fL58a2LeWpZYRtzR8LZz2lkCfPf0UfnCbpI30GyHBgAAAAAtQZOH/VmzZunGG2+0lZWUlGjjz5s2SJ
MMwdOONN+rPf/6duzYoaFDh+pPf/qTfvCDH5j1q6Z1Y+uvddVVVV+nxxx9Z6vRN13333qXi4Z6Y9ehkM5ZlZ4hYXeAd9TmgRTqRqWDTlzcQbqM1nBq8K0
Tgd6f1+SHBwAAAACtTbOM7P/gBz/Qq6++++ai57PB5z/vbbbb9edd96eddpBQsWqF+/frr55pv1w/+UJ9//rkKGokdSdOmTdPZpzzz+vJ554QsXFxFxZo+fbpOPFvURVnv692tY
WmoFhSKHdUsX2WoJ7hiAf2t34/bjc8SfQ53aU8jrG5/M6WeY7WtZ1lHI7SG7OPQAAAAA0RLOEfa/Xqy5duqSVG4Gjx8vSVq4qZKFKSkr02
GOP6eKLL1Zpaan++te/6pFFHtFJ50kSXr00UfVo0cPrfVvcPVrqqxozZkzdNmZIqH4JfCV2+PhPW26I/5JLYuFG78vtzcluHdMX73Ya9dFOW044H3AEAAABAM2mW
sL9u3Tp169Z7ZNgUBAQ4cO1Zw5c3TAAQdo/fr12rx5s5s0aPHm3WDQQCGjlypJYvJvvgLL75YYq1evVjgcttcf0ul5WVNcehZUcs
Fr/0vWK7VLkzc3i3hfZEwN+TEfckTyAR0ovtIT3TqHecTy884o5AAAAAwDEMw5ARMxSNGoqY4pFDUUjMcWiMcWiMUUjhjKRRWKKRo341Fxnr1drfc
t2mdqPRWOKhmOspAmXk5V2k9mjzsDx06VH/729/rr18/r5fbEbpwsMVIyfVzS/fr/br26fVSW6fydy5c9OeFdSdTpmnOnDnqpRnq27ev+vbTPdkv
gbtf6/mlF1c8Q9JcgbsT3cQeJe92D7+KXwwQ6WaaLcVpaY+oKEdwAAAKCZmYE6Ex6EXS5VEcHVZvaLcVpaY+oKEdwAAAKCZmYE6Ex6EXS5V
Rra/ITvDMOQipzRIk4f9sWPHmvTMvODDBg3S8SOHD1adPHy1cuFDxsX/Fyb15DTkhYsxnvvv6CTH6DWz8TDQpxgH21+8UrSD1FPZ2uZ+q7UD/N/P1Ln
aoG/P5fIAAACA4nnCDLfhWMZQHKslPGcMxYkwHEu0ZQVJltBuG8mO1IyOW4N1SwvUdXF7J43ea82+uKL3viU+u8rcxa37JdpvrmOlt9ck1jNPur9/L
y8jRo0CCtW7dO48aNkyRt3rxZXbt2NetYn2jfpUsXhUIh7ZjR9dZ9/b6y/ThUy3hJ6/JqgHsVX0lQTbYP/QDDtyH7Kj5LeEgaOMT0YKvfeMtMIOiPSSvk4
Q3HFLHMm2Wpy2HLciRlmzq2M+um1I1lFWkidskmidskMwB5vagC2l5vrfInyDPUybmMJ0h6vWx6fS+6UAJ9s0xq+3W4Xo+qtSLOH/erqaq1du1bHTPrutSPSWrcPSzyQzeePP3zfu4
mjp0qU6/PDDDJUmhUEjLi3Tbbbfdd/JkkaPHiwfD5li5dqRPtPpgRR388ee36/fbbm7s9fsnTJbN3TDmb7xEb/nRFrr
eUHbOkJdbyj2uuNh2GfdJmW9Z/GdDc02AThvp9rpt27kI1GiTR72r72r7qRp122mnIQ//9tWXLFt7vbtqKKyvT+eeFlLpWnTpnmnOnDnq27ev+vbtq5v
1cTJkyQJBUVFWny5Mma5MmaPn26iouL1auzusqy57lmGxGGwEE2qGfTXhrBNr+7CpOqM4SQGGUgKmOZlFadLpmtuVEY3i3K3qaluM7OzGWKGEuy8KEarKL/1UGFmpzVjs0Q3ZrMl5/VhE57u
WMKHu87viyN1Hmiy+7reXemnJrPU93ddRIhm0ANp2vysL9hwbcx97Gc/0Q/VAGCPvIbNthhdWrlnrlQbOPBaO9GJvzqa6eiLbVBtRcB5LPaF2lSuNBFrkjhQbntFRV30ewj2TlDtBBwzk
OOuu+6S1+vV2WefrcrKSpl44olmOAEbBfBWB7eesw4AAIDM4g85Sw/B5iXd1ku+M+1KKmn9ZFQy5y3kZqzRHTuHo3ZQOrpnZWUSFp8PBDGbqr9ggoPLfqTYPjFAaaLbdq5KSwbMPK9BrFx4m
ht4XzEYMQD2+uKj3QCan8xwjJb5L9JeKeKisrUSUFUV0FRkUplVVWP0DcmUeskEbxGyvwBUdUxQaRJo5owQ1zBaziSzXP1h3R2/HdyWYQaGabVLGIQ0SQx
GuvV3P5uMvNKDawL7SkHNrs9+wDAABg30neqNP4ZhkEd1ZhHTxhGsMN2cghxC6UMBsQFGrJ4o0umBZyETCe1MCcM2l4p6aeuUvQt
l4ynr5NHW167SGbEWwA2UTYBwAaCbJV31FQJVBTkYRPc8gVZzBUjmVZVUOWOJ0lJfWVUrEGZGtabFM3LJ4NqJbbxdnP0kgdAQt9DuQ4o9AWPpW+3l5lNBZ3fvSZhpEuXpzxz...

CPAAAAAC0fIT9LAvm+xXM92e7GwAAAAAAB+FdYwAAAAAAOAxhHwAAAAAAhyHsAwAAAADgMIR9AAAAAAAchrAPAAAAAIDDEPYBAAAA
AHAYwj4AAAAAAA5D2AcAAAAAwGG82e4AAAAAALRmhmHIMGIyYjEZMUNGLKZYLLGcKI9Z5pP1MtZJWVZt9QxDRiyaKLfUiVm2NcujifqxtHq2b
a37NZJ1DUvd1HZr2rCut5Wl7ifle0r2zXZM5raGbb3L5dKlf34026e71SDsAwAAAG2APexF04JfahhNW2fORzOH2lhUsdTQagbW+LRm2zraj0btbdbTj5htXUpZ
NPP+MoXSTKFYsdrqpQZjI9unt21wubLdg1aFsA8AAIBWwTp6mgxxMdt8ND0wWoKjGQ4t2xrReCNRS1B0dq2WZYInlF7AE0Nr7WFy9TwWmfgTgmv8X
2mBF9LWK072NaMihJIs8vlcsvldsvldsnldsvtdteUuSxlbo+tjlzJ8pq65nJie7fZrie9PVembevYh60v9jJrf9PXe+rd1u12S25XxmOvd32yr2gwwj4AAEALZY4kRqOK
xaLxabQmkFrLk6HWWm5E40HR3NYsi6ZsnwyK9vJkyE0LySnhOjVYp4dqa59rC94p4byWttCMUgJczbwnLdxZ12Uur5m3hkm3x2MLdva68XVuT+3t1kw9mb
dN2cbcX6Y+W0Jkan9cblfNPmqt57L3pbaA7XbFlxmVxj5G2AcAAK1OMgRHoxHFIonAGokkwmRE0Uh8WhNq4/Wi1rJY1Nw2YwCORCzhOVITmiMRS0CO
mEE7agbulLYy7C9zm7GafiQDPOG2UZLBzm2GRksA9MQDni0kWtanbpOcd7vtwTNtH7UFXU9qWM0UOlPnPemBtM5tMvUr8zaZw3XKfgijgKMQ9gEAaG
PsQTmiaMQyjSanUUUj4cS6qGKRsKLReCi1bRdN3T6qaCSiaCRszttCd7JOhiAei0bMfcRDdqbyqFnW1rk93nig9CSCpMeTCJiJMo8nHuQsZS5z3lMTRlPLEp/
U0JxeZg+8bo/HDNTJAGluY9k2Uzsut8fenmW7TGHbPlpbsw4AUIOwDwBAE0iGUDPoJsJvPAiHLfOJ8mhKkK6tfjRSS5vp623lZiiP2oN3IrQ79d5dt8crt9cjjxm
EPWaZ2+OVJxFIPcnyZFhOzJuhM7nO7YlvmzEMe8z23B7LfB3bxMu8ZvD2mPt11/TVEuDd1jY9brMNgi0AoD6EfQBAq2MYRjzghsOKRsKHhELxsBsOm2X
J+UiyXjikSMo6cz4SViSUul1qWxFFwqn7qQnXhtG6L7d2ud3yeH3yeONh0+P1yu31Jpbt87Y6ieWM9T2exNRbE7CT65Pbu92Jspow7vZ6awKzta41vJtt1yxzGTI
AADUI+wCABjMMIy1gR0LxIB0Nh+Nh2BqKQ6F4aA4l14VrArd1+2R74QxlkcQ24Zr9RCMt/xJuW3j2WgOxLy0ce7w+S8CuWV9Tx5chWKfWS+7LI4/HMp9
oOxm63d70YO7xMFIMAIDTEPYBoJUyYjFFQiGFQ9WKhEKJT3XiY12umQ9XVysSrmNdYtkaupMhPV7WMkO22+OVx+eTx+eT15uY9/rk8fvl9frMdR7rOp
9PXsu8x+dPLHst9X2Wdu31rCE8LZQnLuEGAADIFsI+ADSxaCSicHWVItXV8SBenQzSlmlVVa3rIqGQbd4M5smQnljXEoJ3PDD7awJxYt7rSwnQtnp+eX1ee
Xx+ebw+ef1+W33r9l5LnZoQbqmfCNeMSgMAANgR9gG0GclL0MOJsG0GbGsoT11nqVNrWE8J7bFodJ8fm8frldcfiAdnvz/DfPo6Xx3rvD6/PP542E4P7n5zlJz7
owEAAFomwj6AFiMaCdeE7+oqS7BOBup4WU0Ir7LVN8N6cj511Ly6ep8+RM3lcsuXEw/TvkDN1BfIkTcQkM8fiE8D9joZg3kgUEuIj89zyTgAAACsCPsAGsw
wjJpR7eoqhauqFLLMp05D1uXqeFi3h3d7qN+XI+Juj1e+HEvgtgbvQE6tQdwW1jOs8ybW+wIBuT2MfAMAACA7CPuAA8Wi0QxhvDIx0p0I4FVV7LqirN0J
2sF6qqVMQW2OPb7ot3cydHxK0j4LaAnRLGU0fLfTk5GUfRk+XJ+78BAAAAp+K3XSALzDBuCd/WEW97WXLku6pmfSLAm9tUJ+4fT9xHHg2Hm/0YvIn
w7AvkyJ+Y+nJyzDLrOmvdTOHdHBH358iXw4g4AAAAsLcI+0Adkg90q64oV3VFhUKVFaquKFeookLVlfFl66h3zb3iNfeW28vi8/vqKeout7smjOfkyJsazAM58
uUEzRHwZD1rWXLenxOsCfP+AE8/BwAAAFowwj4cy4jFFKqqjIf0lLBeM58I74n5TPVi0eYL5rYHuCXDtz9geahbTs0l6YmQ7QsEMpRZ68W39ecE5fH5GCE
HAAAA2iDCPlokwzAUrq5S1e7d8dC9e7eqKspVXb67JpRXVsRH2GuZD1VVNt395S6X/DlBBXLzFMjNlT+Ya059OUHzQW81l6bn2MK5Nzmfckk7ry4DAA
AA0BwI+2g2sWhU1RXlqirfrerymml1+e7E/O6UdfFgnwz4TfVkdo/XK39ungLBXPlzcxXU58mfk9XUVn4pcI+apV8zVoyjNvDenKU
fbeqdifCesVu2wh8qLJyr/vg9ngUyMtXTuLjz81VIC9fgWDQDOTWcJ4pyHv9/ib4NgAAAACg9SDsO1A0ElaoslKhyorENDGdfuH89XBUvq66SUDg5Td7bXlVpl
oUqypvkQXK+nKBy8vIVyMtLmeYrkJunnHZLNC9fObl5CuTnKyc3X95AgMvcAQAAAKCRCPstRCQctZTzC20N7yrQqc1lTv3bN5XYnRtctId0SygO58fIc
y3wgL88ckXd7PE3aHwAAAAAABA3Qj7Wbbs0Yf1/pLFzfLEd68/IH8wGP/k5MqfG5Q/Jyh/MDdRnpuyLleBYFC+RHk8uOfJlxNkdB0AAAAAWhHCfpa53G5b0
Pcm3mceyI0/5T0QzJUvGEwrs4f1oPy5uZZpvJwRdQAAAABomwj7WTbk1J/osNE/MkM7AR0AAAAAsLcI+1mWW1iU7S4AAAAAAAByGF4cDAAAAAAOAw
hH0AAAAAAByGsA8AAAAAAgMMQ9gEAAAAAcBjCPgAAAAAAADkPYBwAAAADAYQj7AAAAAAAA4DGEfAAAAAACHIewDAAAAAAOAwhH0AAAA
AAByGsA8AAAAAAgMMQ9gEAAAAAcBjCPgAAAAAAADkPYBwAAAADAYbzZ7gAAAAAApDIMQ4YhxQxDhmTOyzJvJOtJMmKSofRtDOu8DMWMmr
aVqTxRpkRbhhLtGTV1k9ua5UrfNmZk3r8Mxdcp07bW/aRsm9KOrY61zNJnpdWtOVZlaDv5HSjjfqz7qKVcNe3GMtRRWp9Tz2ttxxj/TiWX5o4f1IR/ypyNsA8A
ANACGIY9bCR/UZZlPi3A1BFqkuWxWOZwFEsNJDIUs4Sl2gJQ2j7SQoZ9Gxn2X/prjitl3ykhI3Ub2eqlB5ZYrL5AmBIoUgNXhu/OXm7ftz0gpYSalO8luZ9k0E
kLiCnfoXX/Uvo5SDu+DOfA2n6sthBWWx9s37m1Dct2GUJZ8s9U6rlO+7OQ8j3by5N/NoF0LpcI+41A2AcAoBbJXzpjhmELW8kyI2ZZp5RAVss0+Ut78hfnWM
z+y7rZtiWMJPehlGUjpQ1reLPuK3XftiDYgHrJ9qR4oIoZqb+YWwJNyrJ5LLbvJ9HPmL0fsvUrpSxDGLCGGPt+04NnauhI7bNSz5+UsX5tbaQGMmvwzNhGhg
AHoHm4XJJLksvlkkuSO1HgMte55HbVrJe1viteP1lXcpntuV0uW9vmvmzbpLfpTuwzsYlZx9wuUVep5Zb+KrUPKW2m9s3af3dKffu8dT/WZWt79m2T34n1uN
K+E+t+bd+JvV23uc6Vvi9JbrerSf9sOB1hHwD2sVjMUDQRTGIxWeYNRZPrkuWx+LqoOZU5X1x1t5sg0jQ3nNtKYftu3Mti1BxuxvTaAyw2Qspa6lfVvdTO3WVt
fy3STrRi3rjQzraw3j1vYs4TDZ9/rqA62J9Rdstyv9F3lrmEn+wm0vSw8x7roCTK37s+7D/su7O8M2Sg0jQ3nNtKYftu3Mti1BxuxvTaAyw2Qspa6lfVvdTO3WVt
mOL7zv9+8y0r7rCnTXEWvdVE+zs58T2XVr6lRrAGtRexsBYf0C2/tmq2U96W7aQW+efu9q3Adoqwj7gELZ7z1QT8OoMO6nhKC2QWYNR3eEpGjPqbDs1oB
qJMmsorQl5hqKW/kQNS/3EfqKWPpvhNSUApwbpmJEaoOvYJpZpHyn9T26Xse2aPlkDdJQE6Whu6y+tiV9Ek790mr/gul22wGTdps6pMpe7Lfsyf+F31/wS7Hb
X/FLtTtlf6v7jyzXhKl5eE14yLae17bYfm3U7+zGkByRrP6whyJ34rd2dUlfW78ZtDz7W7z7T6Fzqd1BbME4dicpUP/U8ZQpf5vfkTumDrN9DyndQXzhKqQ8AgB
VhH2kMw1AkZigSNRSJxRSNxZejMUPhqH05Ek1MLfUybRefxsz64ZihaDSWst5SL1PbUXtoqrkk0n4Jq3kpp+yXlKaG1dTLV1PrpF76aR1tNFK2j39vUvI+tfhc
+uWZ1nvWaq1judctWcmop104i8sleVwuM7Al5z1uV4PK3a5EWSJUehpY7k6043G7avrgiodFlyuxj5RAGK9b0461D8m61v24atnOYw2v7poQmRpmk323Bk3rej
PkWtbVWd9lac/duPrW74iwBQAAWhrCfpaFozFVhaMKRWIKRw2FIjGFolZCF0ViiPD6ttsxb11VH7PVC0TrKI4aqU9q17SOxLSGybUgLO7WENnvQSQQ
fdx3bWtYnR7eSgdSVCGzJsOdJ2V8yzCUDqD10ZqqTsr05X/f2yfDqsRyHNfAmg6PHciw1x5AMzSnbJuqaIdzyXaQel8flkitRt2ZdzbEAAAAAe4Own2VzlqzV/L
e/ynY3GsSbCGjJqc/jrln2uOR1u23rvR532jbxqdvcxpeynKme11Oz7HHXET4tZUoJn+mjdHWP/CVH7Wob2Uuuk2qWJfv9YpL9Hjd7WeK+uAzbmGV11Ek0mW
FfNdskR3zrGqkEAAAAA4EyE/Szze9zmvNftkt/rlt/rls/jlt/jViA5b5a75Pd65Pe45fe6ElPLNl63Ap7UbRLllnYzllv2EfB45PMmg7ebcAgAAAAArYjLMJx5wXZZWZ
mKiopUWlqqwsLCbHenVtWRqAwjHvq5dBcAAAAAWq+WlEMZ2c+ygNeT7S4AAAAAAABzGXX8VAAAAAAADQmhD2AQAAAABwGMI+AAAAAAAOQ
9gHAAAAAMBhCPsAAAAAADgMYR8AAAAAAIch7AMAAAAAA4DDebHcAbYNhGFKGjxFfWfOJxWTEC9PrxlekNrxXy2ltpu2invbq4nLVtbJxxbW1Vec+
6tGAY8n4nadVatDOGteHuvbbyG1qb8rIXKmuPzOWedt3Y9ukAe3W9efQSJtpUDsZ27Dtp+Ht1f33ogn6lda3hrVb79/XutrMtL7efw8a+e9D2uo9+WNCW18asW2t
3ap1u6b5+1lrn5uqP/uiT/Xtp7519TW+p+e9ub/bRn9/e/C9Nvrf+yboa6aqDfldoraNm2Pfzbb/Bvap1roN7FNj9p+piSY+Hw06Fw3+vur5mdLs+2vMPvfN+W7Y7+J7
sc8G9cOlbrfdmqEeMiHsZ9n2RxepbMkS2QOwEf9znQi/aWWjxIhOMO6TAG7tnbSys39ZthnfeWJbWv6CAAAAAB7yUXYbwzCfpaFN2xQ5fvvZ7sbzpY6+r2
Xyw0ZS6/zvzgaPVLS2NGQxLqGjPq3gDq1bllXm4290qEB5a5aytOWGzBfa1u1zde2zlqkxvVhr9tI618T9CW1nbTz0si26ulnWl8btP1etp9qT/4cN8m2tRU3/d+dhp
VnLo5vUt+/H1n4DmtbVeu/Y030Pe3JNo29IqzZ/wzUstts9SlTUcbtM1Zs8n3X+mdob/rUYvffsD8jTX4+GvBdNO6KyWyehwbsuyH7zFAnbV9NeR4a+vdub7v9v1I
uwn2VFP/mJgkccLrlc8b8EyY9ckjtDmcuVmNRT7nYnylPq1tWOuS5R7nanl2WoH580YL+Jv6i1tuNOL3dJ6XVT/sLX/4siAAAAALQtLT7s33ffbrjjju0adMm/e
AHP9Ddd9+tY489NtvdaI5/fspp3+/bHcDAAAAAOAgLfpp/E8++aSum667988GlGDp0qI444gjdf/9
ZtmAAQM0btw85t8RXtWw4ZTCb4cs8ckwr17RXz1v8/7Gw4sd0X/r1t2uZ7vBLGnBepxdWgdm7mnZhOrjw6DWq6DBQxUGfCzjg2fm79fQtMrzVDeXSzm57ovAQQlUwS4dDj
ZtmAAQM0btw4zZ07t85y+e+xB/J8eBbBG5Ad0Bw9hZxXgXJfZbHIPOJpjb5aTjgX/ZSNyMDN/f//qNFWKPc0ZyLLc2rzXuSzdaYA/xpxKtSN1/QuTy95Qy8tgdhWn9+Xbzh6AWUUdxdMWoiUoG1XRC18T9LKMbHP8q8Vn0v8wULdyKhUvXyajCxK4WmxEiHRZpaKbLqJ1CmI4RxJ2kU2rf3pq8Ao3HTMqZ78UUyrd2xxrqnr2oryRWGGR4tHkXbPOHc/Yf5HKM8aSF1Zrv+ZR
2Rq+ZJJpqBUIPxGqCkgtMKGZjZ7CwH+BRWcUeyUX6YYjhxMXyMVzWq5H+U0v/Oq9cXZ7Fzx9yd9ZvCDVBKNWL0HJkjNwjHcZd5H9zSSq2rKHjlUx3Tp2+hWJ2YE7LLF+fa1IiCeMHT/qlMs2WHu8mpD4IBpKEt2X9EZGtwk2CvVDrGkUCA/7BtpBqx5T+LWrAhNbaoLvzWbxS/hlZXaHFZ8jqThs/1+6LOO3Nt3GqlCcR4X6o8Y9pl0FK0lO7Tg2Jq3/5q8A5HuI1FPLUrWHaiXGzpLSwaTrzd2JeufDHHN0/vdiqDwEpV0uoN2NYvT7hi7IuqvVbf7ssU6kTVfAXW6I33pwq4wJRTOBgLqbm4YOvc9O1D8fgn8T0KrQXxuT0bzXJPO+A/lqRRVzKzhTA7/zCqKVCM2QGPSbJzxd8Jm6wkR9DuRbV5sY7Zq3Pf6/Hg1Ek9xE8SkMwKcG9m3mcv4hV6FiJI2uQPHvnGE//eYnnjqcxGGj+dEF1eTRdThHHYfpu0h+hT8b2Dd9B3yP8H4TUBmSUO8k0cwfTeUXHs6x2AkdA7Mq2EGPD7HpVSZYYsM7I/AmDL9PNFb1W6p5IrMbe69YmP1HSV2Prqfn3ckU/XaXAPY6eqsS2oVkEyKQ3kPzvQ1I8d5Jr8cSxD+I0bsMvnBcOTW9mhMu8ypJfbtK3UhA3RrFw9N5dpcZqjWJ6RYZhEqE3YeSzn9ZEvMR8xg3xOnrKAMWcdMJFo+JsrC4vv4VWaNsf35ht97SoG/xd0n+Qqf7qH+4NoHQgZGa4Gxfw3rRlxwCkNeQqQNd7T4i8a2E0s94w9K3ifuBJdK8EF3BtGQ2BeSbqvgl2Ysex1hKOQfD81zw5gq4j4W0BWyT2XGTB1xTe/rE9oAI0OAtWe1kyM+F4I5Tj2sVxsSBb7BncZB3/jN/9zBFZNVIIYN3nCT3RZCQ2RPfmc51nIHHRMGZ4AqOGQWdkTxmDZ1hMlD+eF8F2Nd8zT2UV4XkwTHQMY1/yfnkeL8KVM4aj9dGL7cClOhp1OnRKhJ4pzLNYwr2OkVUhCm6hfwSK8JV6Y/k0jN1kGi/nHEgZBhuNVFQkLdYxtyJxm9x1RqM1RQfvq/lKQvkxaaCVcPj0JJKhHMmKRx4AYn3XHNW5DrJRqM5/4LaG5bR9jj5hMhv2DlhSgQ3wbIRNWRPWO9ZSKOqnwc5U2e8hJcKVjWrbZJWgCXwAaCnKbYjEEr6iEPqKv57mcdM+f2bLsBt2bx2bLXY4XVB3aL9rkU8RsPeYl6xtGXwVZW1kW1+5rxrd4BK6bSkqRNWjU2Oq6yc9bPybGbzvr7m5Kh3c9VLbx2TBmLpLEnwP7Xg2dQgL+oAD+gAAAAAAAAAAAAIDDEPYBA
AAAAHAYwn4LEQgENHPmTAUCzOyT2pzRzUiBQeK8tV6cu9aLc9e6cL5ap5Z3+3hz7gD4AAAAAANoqRvYBAAAAAHAYwj4AAAAAAAA5D2
AcAAAAAwGEI+wAAAAAAOAxhvw5z587VfHJJfpmKeDj4jcsf2TtJJbXYnB5KH+Dvf2t1f+UpVRIFTrZZ8rkZ/nR0m7RrOWS0tk6fbVBzknTaZxrp6/Bx/
VWlqacZ/V1dU67LDD5HK5tGbNmnr7+NFHH2nkyJEKBoaPab7/9NHv2bF3mTU1a6ObZGUAAAAAAAA5D2A

fv357duCtWGs8Z6na6s83qeWev4Zo6/+WNue5i8VikqTrrrtOZ5xxhgYPHqz58+fL5XLp6aeflsTPuMZqjeerrf98k1rneUu1Nz/j2t5/7+yBKVOmaPHixXrjjTfUvXt3
s7xLly6S4v8D07VrV7N8y5Ytaf9bs2vXLp188snKz8/Xs88+K5/PZ6577bXXtHLlStulHJI0ZMgQnXvuuVq4cKGWLFliXtYWDAbN/af+78+WLVskpf9PlWEYev
jhhzVx4kT5/f49+h5ao5Z67hqitvPr9XpVXFzc4HZaq+Y+d5kMGzZMkvTf//5XXcXFe/33rq1preds165deu+99/TB/2/v/l3a1+Iwjp8LYnTQOtWig+JSFBzEoS7+QF
x1r7i5WBRE/AN0rIujoiDOgnRxqoud3CRgVETEou5iBQVFfb7DvfY2WL1WvaU5fb8gS5om4TyETz4pPXFdMzMzY4z5u4BLMjU1NWZ3d9cMDw+XPB5BE
NTMClVrfTOmcvP7Cmrc/5vd23e7urry6xzHMR0dHebq6soYY6hxJQhqXtVc34wJbm6FflzjSvqHf5V5fX3V9PS0WlpadHZ2VvTzSCSipaWl/LrHx8d3kzvkcjn19
fVpcHBQ9/f37/ZzeXkpz/PySzqdljFG29vbur6+/vD8VlZW1NTUpMfHx/y6ZDJZdHKHvb09GWPkeV5JYxBUlZ5doc8mL+rs7PStm5qasn7yonJlV8zOzo6MMbq
8vPxwm1KuO6k6Ji8KemYvLy++69jzPCUSCUWjUXme55u91xZBz6xQtdU3qfLzK0SN8ytXdrlcTo7j+CYOe3p6Ujgc1tra2ofnR43zC3pe1VjfpODnVuinNY5m/
xOJREKhUEiZTMb36oOHh4f8NslkUqFQSKlUSp7nKR6P+17bcHd3p1gspu7ubp2fn/v28/z8XPS42Wz2SzO6397eqrm5WfF4XJ7nKZVKqbGxsejrNiYmJhSLxb
4/GAFT6dlJ0vHxsVzX1ejoqIaGhuS6ru97b68lmpub08nJiTY2NqritUTlym5/f1/Ly8tyXVcXFxfa2tpSS0uLxsbGPj2/r153b3n29vZqfHxcruvq+Pj4l0erMtiSWSHbZ
yu2KbNqq29S5ecnUeM+Us77k9nZWbW2tiqdTuv09FSTk5MKh8O6ubn58PyocX625FXI9vom2ZXbT2sczf4njDFFl83Nzfw2r6+vWlhYUCQSkeM4GhgY8D15e
XsaU2zJZrNFj1tKw3h4eKj+/n45jqNIJKLFxcV3T4Rub29VX1+v9fX17wxDIAUhu7a2tqL7LpTJZNTT06Pa2lq1t7drdXX1O8MRKOXK7uDgQLFYTKFQSHV1
dYpGo1pYWPjSr1tfue6KHbutre03hqji2JJZIdtvhmzJrBrrmxSM/KhxxZXz/uTp6Unz8/MKh8NqaGjQyMiIjo6O/vMcqXH/siWvQrbXN8me3H6jxv31z4AAAAAA
AABLMBs/AAAAAACWodkHAAAAAMAyNPsAAAAAAFiGZh8AAAAAAMvQ7AMAAAAAAYBmafQAAAAAALEOzDwAAAACAZWj2AQAAAACwD
M0+AAAAAACWodkHAAAAAMAyNPsAAAAAAFiGZh8AAAAAAMv8AVw0EdUmDTEuAAAAAElFTkSuQmCC",
"text/plain": [
"<Figure size 1200x800 with 1 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
}
],
"source": [

```
"if r > 0:\n",
"    # Creating a VAR model for prediction using the VECM\n",
"    model = VAR(commodity_data)\n",
"    vecm_result = model.fit(r)\n",
"    \n",
"    # Summary of the VECM model\n",
"    print(vecm_result.summary())\n",
"    \n",
"    # Forecasting using the VECM model\n",
"    # Forecasting 24 steps ahead\n",
"    forecast = vecm_result.forecast(commodity_data.values[-vecm_result.k_ar:], steps=24)\n",
"    \n",
"    # Convert forecast to DataFrame for plotting\n",
"    forecast_df = pd.DataFrame(forecast, index=pd.date_range(start=commodity['date'].iloc[-1], periods=24, freq='M'), columns=commodity_data.columns)\n",
"    \n",
"    # Plotting the forecast\n",
"    plt.figure(figsize=(12, 8))\n",
"    for col in forecast_df.columns:\n",
"        plt.plot(forecast_df.index, forecast_df[col], label=col)\n",
"    plt.legend()\n",
"    plt.title('VECM Forecast')\n",
"    plt.show()\n",
"\n",
"else:\n",
"    # If no co-integration exists, proceed with Unrestricted VAR Analysis\n",
"    model = VAR(commodity_data)\n",
"    var_result = model.fit(maxlags=10, ic='aic')\n",
"    \n",
"    # Summary of the VAR model\n",
"    print(var_result.summary())\n",
"    \n",
"    # Granger causality test\n",
"    causality_results = {}\n",
"    for col in commodity_data.columns:\n",
"        causality_results[col] = var_result.test_causality(causing=col, caused=commodity_data.columns.difference([col]), kind='f').summary()\n",
"    print(causality_results)\n",
"    \n",
"    # Forecasting using the VAR model\n",
"    forecast = var_result.forecast(commodity_data.values[-var_result.k_ar:], steps=24)\n",
"    \n",
"    # Convert forecast to DataFrame for plotting\n",
```

```
    "    forecast_df = pd.DataFrame(forecast, index=pd.date_range(start=commodity['date'].iloc[-1], periods=24, freq='M'), columns=commodity_data.columns)\n",
    "    \n",
    "    # Plotting the forecast\n",
    "    plt.figure(figsize=(12, 8))\n",
    "    for col in forecast_df.columns:\n",
    "        plt.plot(forecast_df.index, forecast_df[col], label=col)\n",
    "    plt.legend()\n",
    "    plt.title('VAR Forecast')\n",
    "    plt.show()"
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3 (ipykernel)",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.11.5"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```