

Assignment 3 Solution

Contributors:

- **Idhant Kadel**a 230481
- **Aaditya Rathi** 230006
- **Aryan Saini** 230220
- **Shagun** 230949
- **Shreyansh Kothari** 230988

```
!pip install arch
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
import statsmodels.api as sm
from arch import arch_model
```

```
from google.colab import drive
drive.mount('/content/drive')
```

Reading the data

```
data = pd.read_csv("/content/drive/MyDrive/EC0423 assignments/Assignment3_Data - Sheet1.csv")
print(data.head())
```

	Year	Jet Kerosene-FOB	Cargoes	Med.\$/MT	USD/AUD	AUD/USD
0	1/12/1994			AUD 151.50	AUD 0.69	AUD 1.45
1	2/12/1994			AUD 148.00	AUD 0.72	AUD 1.40
2	3/12/1994			AUD 146.50	AUD 0.71	AUD 1.41
3	4/12/1994			AUD 150.00	AUD 0.72	AUD 1.38
4	5/12/1994			AUD 153.00	AUD 0.72	AUD 1.39

Q1

Base Inputs

```
# Base scenario variables
flights_march = 700
avg_revenue_per_flight = 50_000      # AUD
delta_flights_vs_usd = -0.7
fixed_costs = 15_000_000             # AUD
jet_kerosene_per_flight = 13          # metric tonnes
other_variable_costs = 9_000         # AUD per flight
usd_per_aud_march = 0.77495
usd_per_aud_april = 0.76225
jet_kerosene_usd_per_mt = 510.75     # USD/metric tonne

# Convert jet kerosene cost to AUD/metric tonne
kerosene_price_aud = jet_kerosene_usd_per_mt / usd_per_aud_april
kerosene_cost_per_flight = kerosene_price_aud * jet_kerosene_per_flight
```

Scenario 1 : No change in number of flights

```
flights_april_no_change = flights_march
revenue_no_change = flights_april_no_change * avg_revenue_per_flight
variable_costs_no_change = flights_april_no_change * (kerosene_cost_per_flight + other_variable_costs)
total_costs_no_change = variable_costs_no_change + fixed_costs
earnings_no_change = revenue_no_change - total_costs_no_change
```

```
print(f"Earnings before taxes (no change in flights): AUD {earnings_no_change:,.2f}")
```

Earnings before taxes (no change in flights): AUD 7,602,492.62

Scenario 2 : Change in number of flights due to exchange rate

```
# Change in exchange rate return (%)
usd_return = (usd_per_aud_april / usd_per_aud_march) - 1
percent_change_flights = usd_return * delta_flights_vs_usd

# Expected flights
flights_april_change = round(flights_march * (1 + percent_change_flights))

revenue_change = flights_april_change * avg_revenue_per_flight
```

```

variable_costs_change = flights_april_change * (kerosene_cost_per_flight + other_variable_costs)
total_costs_change = variable_costs_change + fixed_costs
earnings_change = revenue_change - total_costs_change

print(f"Earnings before taxes (with change in flights): AUD {earnings_change:,.2f}")

```

Earnings before taxes (with change in flights): AUD 7,860,806.82

Q2

```

# Manual Log Returns using shift
data['Jet Kerosene log returns'] = np.log(data['Jet Kerosene-FOB Cargoes Med.$/MT'] / data['Jet Kerosene-FOB Cargoes Med.$/MT'].shift(1))
data['AUD/USD log returns'] = np.log(data['AUD/USD'] / data['AUD/USD'].shift(1))

# Remove NaNs
returns = data.dropna()

# Monthly Volatility and Mean
jet_vol_month = returns['Jet Kerosene log returns'].std()
fx_vol_month = returns['AUD/USD log returns'].std()

jet_mean_month = returns['Jet Kerosene log returns'].mean()
fx_mean_month = returns['AUD/USD log returns'].mean()

# Annualized Volatility and Mean
jet_vol_year = jet_vol_month * np.sqrt(12)
fx_vol_year = fx_vol_month * np.sqrt(12)

jet_mean_year = jet_mean_month * 12
fx_mean_year = fx_mean_month * 12

# Output
summary = pd.DataFrame({
    'Jet Kerosene return in USD': [jet_vol_month, jet_vol_year, jet_mean_month, jet_mean_year],
    'AUD/USD returns': [fx_vol_month, fx_vol_year, fx_mean_month, fx_mean_year]
}, index=['Volatility per month', 'Volatility per annum', 'Mean per month', 'Mean per annum'])

print(summary)

```

	Jet Kerosene return in USD	AUD/USD returns
Volatility per month	AUD 0.12	AUD 0.03
Volatility per annum	AUD 0.40	AUD 0.11
Mean per month	AUD 0.00	AUD -0.00
Mean per annum	AUD 0.04	AUD -0.00

Q3

Finding the Probability of Returns Being Lower Than -23%

The probability that jet kerosene returns in USD will be lower than -23% can be found using the standard normal distribution.

Z-Score Formula:

$$Z = \frac{\text{Threshold Return} - \mu}{\sigma}$$

where:

- Threshold Return: -0.23 (or -23%)
- μ : mean monthly log return (use your calculated mean)
- σ : monthly volatility (standard deviation of log returns)

Cumulative Probability:

$$P(\text{return} \leq -23\%) = \Phi(Z)$$

Where $\Phi(Z)$ is the cumulative standard normal distribution evaluated at Z.

```

# Threshold return
threshold = -0.23

# Calculate Z-score
z_score = (threshold - jet_mean_month) / jet_vol_month

# Calculate cumulative probability
probability = norm.cdf(z_score)

# Print results
print(f"Z-score : {z_score:.4f}")
print(f"Probability : {probability:.2%}")

```

Z-score : -2.0285
Probability : 2.13%

Q4

Empirical Probability

Empirical Probability is calculated using the numerical definition of probability:

$$\text{Empirical Probability} = \frac{\text{Number of observations where return} < -23\%}{\text{Total number of observations}}$$

If the empirical probability is larger than the probability estimated from the normal distribution, this suggests that:

- Normality assumptions underestimate extreme events, especially over longer risk horizons.
- In reality, financial returns may exhibit fat tails and time-varying volatility, violating normal distribution assumptions.

```
# Calculating the empirical probability using booleans
empirical_prob = (data['Jet Kerosene log returns'] < threshold).mean()


# Print results
print(f"Empirical Probability: {empirical_prob:.2%}")
```

 Empirical Probability: 2.74%

Q5

```
# Calculate correlation between log returns in Jet Kerosene Prices and FX Rate
combined_returns = pd.concat([returns['Jet Kerosene log returns'], returns['AUD/USD log returns']], axis=1).dropna()
correlation = combined_returns.corr().iloc[0, 1]

print(f"Correlation between Jet Kerosene returns and AUD/USD returns: {correlation:.4f}")
```

 Correlation between Jet Kerosene returns and AUD/USD returns: -0.3077

Q6

For simulating prices, we are using Cholesky decomposition in numpy to simulate the two correlated returns.

```
# PARAMETERS

# Financial parameters
n_paths = 1000
n_months = 12
n_iterations = 10


# Jet Fuel and FX parameters
vol_jet = 0.1152 # monthly volatility for jet fuel
vol_fx = 0.0316 # monthly volatility for AUD/USD
corr = 0.30 # Correlation

# Flight & Cost parameters
fixed_costs = 15_000_000 # AUD
revenue_per_flight = 50_000 # AUD
jet_kerosene_per_flight = 13 # metric tonnes
other_variable_costs_per_flight = 9_000 # AUD
starting_flights = 700 # Number of flights to start
delta = -0.70

# Initial Prices (April 2021)
initial_jet_price = 510.75 # USD per MT
initial_fx = 0.76225 # USD per AUD
```

```
# CHOLESKY DECOMPOSITION FOR CORRELATED NORMALS

cov_matrix = np.array([
    [vol_jet**2, corr * vol_jet * vol_fx],
    [corr * vol_jet * vol_fx, vol_fx**2]
])
chol = np.linalg.cholesky(cov_matrix)
chol
```

 array([[0.1152, 0.],
 [0.00948, 0.03014448]])

```
# SIMULATE 10 ITERATIONS OF 1000 PATHS EACH

summary_list = []

for iteration in range(n_iterations):
    total_earnings = []
```

```

for path in range(n_paths):
    jet_price = initial_jet_price
    fx = initial_fx
    flights = starting_flights

    monthly_earnings = []
    prev_fx_ret = -0.0768 # Returns of March 2021

    # Storing values of first path only
    if iteration == 0 and path == 0:
        jet_price_path = []
        fx_path = []
        flights_path = []
        earnings_path = []

    for month in range(n_months):
        shocks = np.random.normal(size=2)
        log_returns = chol @ shocks
        jet_ret, fx_ret = log_returns

        # Update prices
        jet_price *= np.exp(jet_ret)
        fx *= np.exp(fx_ret)

        # Update flights based on previous month's fx return
        flights = int(round(flights * (1 + delta * prev_fx_ret)))
        prev_fx_ret = fx_ret

        # Revenue
        revenue = flights * revenue_per_flight

        # Variable costs
        jet_cost_per_flight = (jet_price / fx) * jet_kerosene_per_flight
        variable_costs = flights * (jet_cost_per_flight + other_variable_costs_per_flight)
        total_costs = fixed_costs + variable_costs

        earnings = revenue - total_costs
        monthly_earnings.append(earnings)

    # Store prices and earnings for the first path only
    if iteration == 0 and path == 0:
        jet_price_path.append(jet_price)
        fx_path.append(fx)
        flights_path.append(flights)
        earnings_path.append(earnings)

    # Total earnings for this path over 12 months
    total_earnings.append(sum(monthly_earnings))

# SUMMARY STATISTICS FOR THIS ITERATION

total_earnings = np.array(total_earnings)

summary = {
    "Maximum": np.max(total_earnings),
    "95th Percentile": np.percentile(total_earnings, 95),
    "Median": np.percentile(total_earnings, 50),
    "5th Percentile": np.percentile(total_earnings, 5),
    "Minimum": np.min(total_earnings),
}

summary["EaR (95% Confidence)"] = summary["Median"] - summary["5th Percentile"]
summary_list.append(summary)

# AVERAGE ACROSS 10 ITERATIONS

summary_df = pd.DataFrame(summary_list)
avg_summary = summary_df.mean().to_frame(name="Average Earnings over 12 months")
pd.options.display.float_format = 'AUD {:.2f}'.format

# DISPLAY RESULTS

months = list(range(1, len(jet_price_path) + 1))

fig, axes = plt.subplots(2, 2, figsize=(14, 10), sharex=True)

# Top Left: Flights
axes[0, 0].plot(months, flights_path, marker='^', color='orange')
axes[0, 0].set_title("Number of Flights (First Path)")
axes[0, 0].set_ylabel("Flights")
axes[0, 0].grid()

# Top Right: Earnings
axes[0, 1].plot(months, earnings_path, marker='s', color='red')

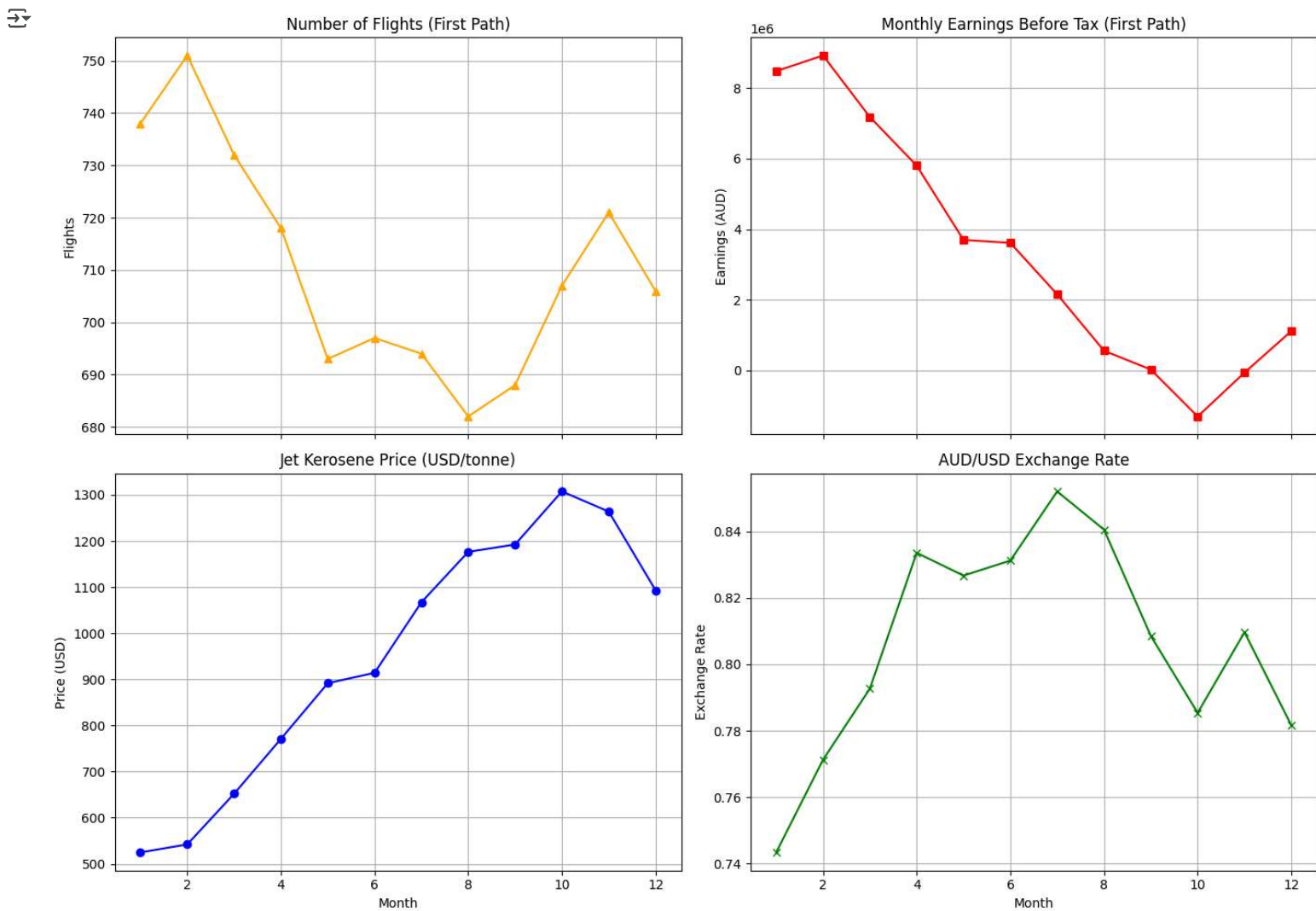
```

```
axes[0, 1].set_title("Monthly Earnings Before Tax (First Path)")
axes[0, 1].set_ylabel("Earnings (AUD)")
axes[0, 1].grid()

# Bottom Left: Jet Kerosene Price
axes[1, 0].plot(months, jet_price_path, marker='o', color='blue')
axes[1, 0].set_title("Jet Kerosene Price (USD/tonne)")
axes[1, 0].set_xlabel("Month")
axes[1, 0].set_ylabel("Price (USD)")
axes[1, 0].grid()

# Bottom Right: AUD/USD Exchange Rate
axes[1, 1].plot(months, fx_path, marker='x', color='green')
axes[1, 1].set_title("AUD/USD Exchange Rate")
axes[1, 1].set_xlabel("Month")
axes[1, 1].set_ylabel("Exchange Rate")
axes[1, 1].grid()

plt.tight_layout()
plt.show()
```



```
print(f"\nAverage Summary Statistics across {n_iterations} iterations of {n_paths} paths:\n")
print(avg_summary)
```



Average Summary Statistics across 10 iterations of 1000 paths:

Average Earnings over 12 months	
Maximum	AUD 170,420,737.77
95th Percentile	AUD 138,208,079.17
Median	AUD 104,390,740.68
5th Percentile	AUD 63,253,103.84
Minimum	AUD 11,336,844.94
EaR (95% Confidence)	AUD 41,137,636.84

$$h_{jet} = \frac{\text{Var}(\text{Jet Price})}{\text{Cov}(\text{Earnings, Jet Price})}$$

This hedge ratio shows how much exposure in jet kerosene futures contracts you need to hedge the variance of earnings due to jet fuel price fluctuations (assuming no hedge on the currency side).

$$h_{FX} = \frac{\text{Var}(\text{FX Rate})}{\text{Cov}(\text{Earnings, FX Rate})}$$

This hedge ratio shows how much exposure in FX Rate futures contracts you need to hedge the variance of earnings due to FX Rate fluctuations (assuming no hedge on the Jet Fuels Returns).

```
# Simulate log returns for May 2021
mean = [0, 0]
log_returns = np.random.multivariate_normal(mean, cov_matrix, size=n_paths)

# Convert log returns to prices
jet_may_prices = initial_jet_price * np.exp(log_returns[:, 0])
fx_may_rates = initial_fx * np.exp(log_returns[:, 1])

# Revenue and cost parameters
starting_flights = 700
revenue_per_flight = 50000
jet_kerosene_per_flight = 13
other_variable_costs_per_flight = 9000
fixed_costs = 15_000_000
delta = -0.70
prev_fx_ret = -0.0768 # previous month's return in AUD/USD

# Calculate number of flights for May (single value since it's deterministic)
flights_may = int(round(starting_flights * (1 + delta * prev_fx_ret)))

# Calculate earnings for May for each path
revenues = flights_may * revenue_per_flight
jet_cost_per_flight = (jet_may_prices / fx_may_rates) * jet_kerosene_per_flight
variable_costs = flights_may * (jet_cost_per_flight + other_variable_costs_per_flight)
total_costs = fixed_costs + variable_costs
earnings_may = revenues - total_costs

# Calculate correlation between earnings and jet kerosene prices
correlation_earnings_jet = np.corrcoef(earnings_may, jet_may_prices)[0, 1]
print(f"a) Correlation between earnings and jet kerosene prices (May 2020): {correlation_earnings_jet:.4f}")

# Calculate correlation between earnings and AUD/USD exchange rates
correlation_earnings_fx = np.corrcoef(earnings_may, fx_may_rates)[0, 1]
print(f"b) Correlation between earnings and AUD/USD exchange rate (May 2020): {correlation_earnings_fx:.4f}")

# Calculate covariance and variance
cov_earnings_jet = np.cov(earnings_may, jet_may_prices)[0, 1]
var_jet = np.var(jet_may_prices)

# Minimum variance hedge ratio
hedge_ratio_jet = cov_earnings_jet / var_jet
print(f"c) Minimum variance hedge ratio for jet kerosene: {hedge_ratio_jet:.4f}")

# Calculate covariance and variance
cov_earnings_FX = np.cov(earnings_may, fx_may_rates)[0, 1]
var_FX = np.var(fx_may_rates)

# Minimum variance hedge ratio
hedge_ratio_FX = cov_earnings_FX / var_FX
print(f"d) Minimum variance hedge ratio for FX Rate: {hedge_ratio_FX:.4f}")
```

- a) Correlation between earnings and jet kerosene prices (May 2020): -0.9589
 b) Correlation between earnings and AUD/USD exchange rate (May 2020): -0.0477
 c) Minimum variance hedge ratio for jet kerosene: -11412.2293
 d) Minimum variance hedge ratio for FX Rate: -1351212.5988

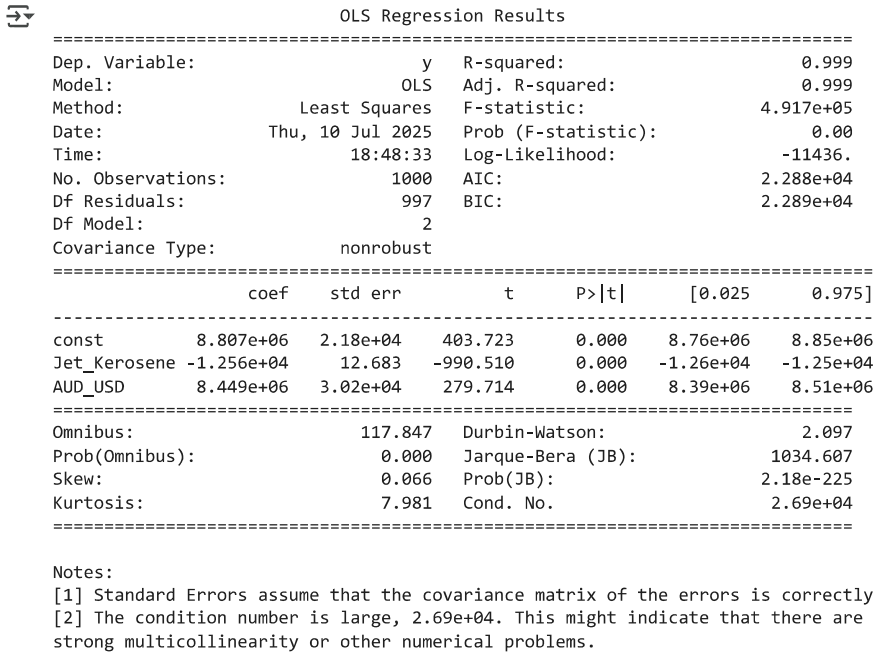
```
# Prepare the independent variables
X = pd.DataFrame({
    'Jet_Kerosene': jet_may_prices,
    'AUD_USD': fx_may_rates
})

# Add a constant for the intercept
X = sm.add_constant(X)

# Dependent variable
y = earnings_may

# Build and fit the regression model
model = sm.OLS(y, X).fit()
```

```
# Display the regression summary
print(model.summary())
```



Learnings

- Jet fuel prices have a very strong negative impact on earnings, meaning higher fuel prices significantly reduce profits.
- The exchange rate has almost no meaningful impact on earnings in this scenario.
- The fuel hedge ratio is large but practical, reflecting significant exposure.
- The FX hedge ratio is large in magnitude but unjustified by the weak correlation. Hedging FX in this case would introduce unnecessary complexity and cost.
- Jet Kerosene: For every \$1 increase in jet fuel price, earnings fall by ~\$12,590.
- AUD/USD: There is a positive earnings impact, but given the very low correlation, the practical hedging benefit is limited.

Most Efficient Hedge:

- Hedge Jet Kerosene prices using fuel futures or options.
- This risk has both a strong empirical correlation (-0.96) and a large economic impact (-12,590 AUD per \$1 fuel price change).
- Use monthly jet fuel futures contracts to offset the price fluctuations.

```
# Calculate log returns (expressed in % for interpretability)
jet_returns = 100 * np.log(data['Jet Kerosene-FOB Cargoes Med.$/MT'] / data['Jet Kerosene-FOB Cargoes Med.$/MT'].shift(1)).dropna()
fx_returns = 100 * np.log(data['AUD/USD'] / data['AUD/USD'].shift(1)).dropna()

# Fit GARCH(1,1) on Jet Kerosene Returns
jet_model = arch_model(jet_returns, vol='Garch', p=1, q=1)
jet_fit = jet_model.fit(dispatch='off')

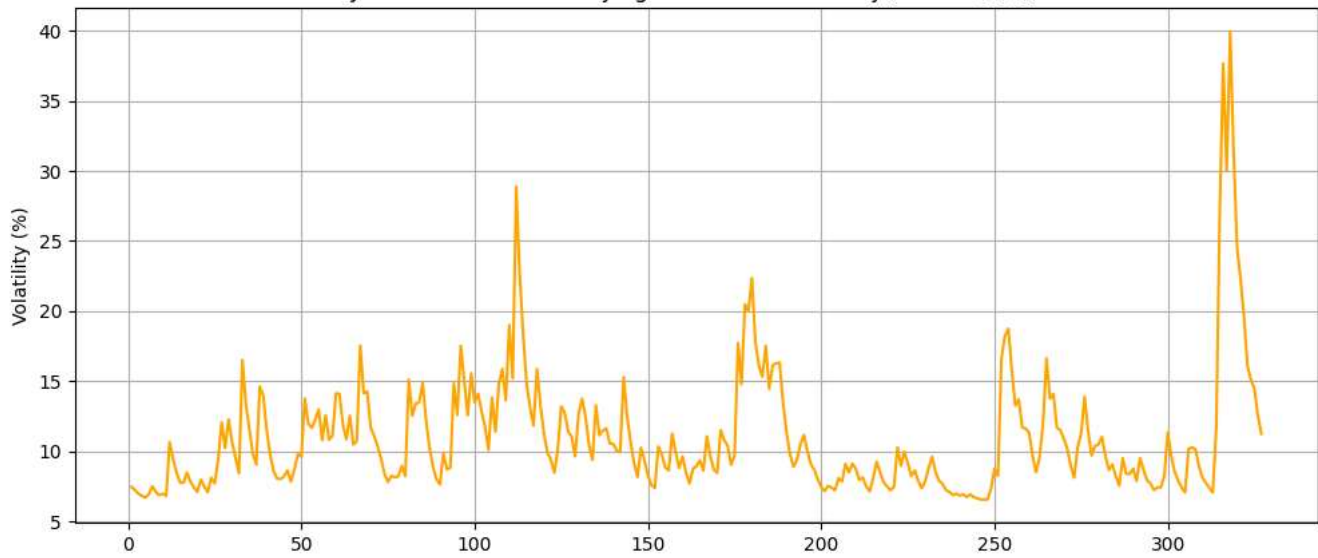
# Fit GARCH(1,1) on AUD/USD Returns
fx_model = arch_model(fx_returns, vol='Garch', p=1, q=1)
fx_fit = fx_model.fit(dispatch='off')

# Plot Conditional Volatility
plt.figure(figsize=(12, 5))
plt.plot(jet_fit.conditional_volatility, color='orange')
plt.title("Jet Kerosene: Time-Varying Conditional Volatility (GARCH(1,1))")
plt.ylabel("Volatility (%)")
plt.grid()
plt.show()

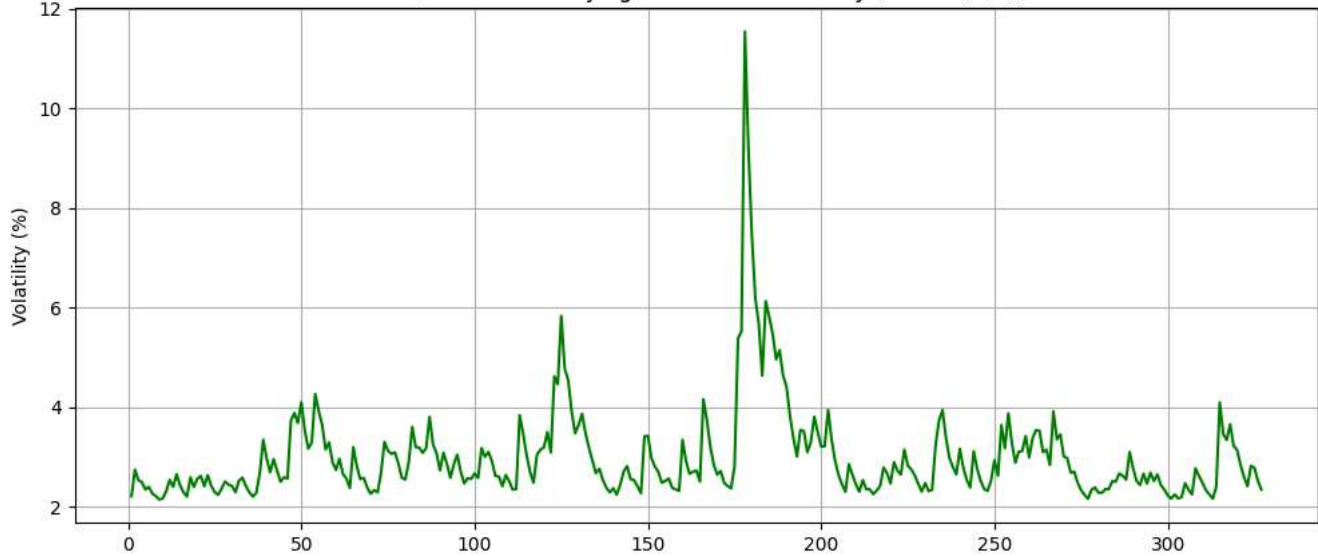
plt.figure(figsize=(12, 5))
plt.plot(fx_fit.conditional_volatility, color='green')
plt.title("AUD/USD: Time-Varying Conditional Volatility (GARCH(1,1))")
plt.ylabel("Volatility (%)")
plt.grid()
plt.show()
```



Jet Kerosene: Time-Varying Conditional Volatility (GARCH(1,1))



AUD/USD: Time-Varying Conditional Volatility (GARCH(1,1))



```
print(jet_fit.summary())
print(fx_fit.summary())
```



Constant Mean - GARCH Model Results

```
=====
Dep. Variable:    Jet Kerosene-FOB Cargoes Med.$/MT    R-squared:                0.000
Mean Model:      Constant Mean                        Adj. R-squared:            0.000
Vol Model:       GARCH                                Log-Likelihood:          -1228.13
Distribution:     Normal                              AIC:                     2464.27
Method:          Maximum Likelihood                  BIC:                     2479.43

Date:            Thu, Jul 10 2025                    No. Observations:        327
Time:            18:48:33                             Df Residuals:            326
                                           Df Model:                 1
```

Mean Model

```
=====
              coef    std err          t      P>|t|  95.0% Conf. Int.
-----
mu           0.5913    0.507       1.167    0.243 [ -0.401,  1.584]
Volatility Model
=====
              coef    std err          t      P>|t|  95.0% Conf. Int.
-----
omega       16.7492    9.339       1.794  7.289e-02 [ -1.554,  35.053]
alpha[1]     0.3070    0.123       2.489  1.281e-02 [ 6.527e-02,  0.549]
beta[1]       0.5916    0.133       4.448  8.655e-06 [  0.331,  0.852]
=====
```

Covariance estimator: robust

Constant Mean - GARCH Model Results

```
=====
Dep. Variable:    AUD/USD    R-squared:                0.000
Mean Model:      Constant Mean    Adj. R-squared:            0.000
Vol Model:       GARCH            Log-Likelihood:          -811.561
Distribution:     Normal          AIC:                     1631.12
Method:          Maximum Likelihood    BIC:                     1646.28

Date:            Thu, Jul 10 2025    No. Observations:        327
Time:            18:48:33             Df Residuals:            326
                                           Df Model:                 1
```

Mean Model

```
=====
              coef    std err          t      P>|t|  95.0% Conf. Int.
-----
```


mu	-0.1112	0.153	-0.726	0.468	[-0.411, 0.189]	
Volatility Model						
=====						
	coef	std err	t	P> t	95.0% Conf. Int.	

omega	1.5631	0.771	2.028	4.258e-02	[5.231e-02, 3.074]	
alpha[1]	0.2205	0.123	1.793	7.305e-02	[-2.060e-02, 0.462]	
beta[1]	0.6195	0.151	4.115	3.877e-05	[0.324, 0.915]	
=====						
Covariance estimator: robust						

▼ Jet Kerosene Price Volatility

Mean Model (Constant Mean)

Mean return (μ) = 0.59%, statistically insignificant ($p \approx 0.24$).

Indicates no significant upward or downward drift in prices.

Volatility Model (GARCH(1,1)):

- ω (base volatility): 16.75
- α_1 (impact of recent shocks): 0.31
- β_1 (impact of past volatility): 0.59

Volatility persistence ($\alpha + \beta$): ~ 0.90 , showing strong volatility clustering and persistence.

Plot Interpretation:

Frequent volatility spikes, especially between periods 28-320, indicating large shocks in jet kerosene markets.

Volatility clustering: high volatility periods tend to follow one another.

AUD/USD Exchange Rate Volatility

Mean Model (Constant Mean)

Mean return (μ) = -0.11%, also statistically insignificant ($p \approx 0.47$).

Exchange rate movements exhibit no strong trend.

Volatility Model (GARCH(1,1))

- ω (base volatility): 1.56
- α_1 (impact of recent shocks): 0.22
- β_1 (impact of past volatility): 0.62

Volatility persistence ($\alpha + \beta$): ~ 0.84 , indicating some clustering but lower persistence compared to jet kerosene.

Plot Interpretation Generally stable volatility with occasional spikes, most notably around period 180–190.

Exchange rate volatility is lower and less persistent compared to jet fuel prices.

```

# Step 1: Standardize residuals from GARCH fits
jet_resid_std = jet_fit.resid / jet_fit.conditional_volatility
fx_resid_std = fx_fit.resid / fx_fit.conditional_volatility

# Step 2: Create a DataFrame of standardized residuals
std_resid = pd.DataFrame({
    'Jet': jet_resid_std,
    'FX': fx_resid_std
}).dropna()

# Step 3: Calculate rolling window correlation (as DCC proxy)
window = 30
rolling_corr = std_resid['Jet'].rolling(window).corr(std_resid['FX'])

# Step 4: Calculate rolling covariance and variance for hedge ratio
rolling_cov = std_resid['Jet'].rolling(window).cov(std_resid['FX'])
rolling_var_jet = std_resid['Jet'].rolling(window).var()

# Hedge ratio: cov(Jet, FX) / var(Jet)
hedge_ratio = rolling_cov / rolling_var_jet

# Step 5: Plot conditional correlation
plt.figure(figsize=(12, 5))
plt.plot(rolling_corr, color='blue')
plt.title('Rolling Conditional Correlation (Proxy for DCC)')
plt.ylabel('Conditional Correlation')
plt.grid()
plt.show()

# Step 6: Plot time-varying hedge ratio
plt.figure(figsize=(12, 5))
plt.plot(hedge_ratio, color='red')
plt.title('Time-Varying Hedge Ratio: Jet Kerosene Hedged with AUD/USD')
  
```

```
plt.ylabel('Hedge Ratio')
plt.grid()
plt.show()
```



Rolling Conditional Correlation (Proxy for DCC)



Time-Varying Hedge Ratio: Jet Kerosene Hedged with AUD/USD

