ASSURE-US SUMMER 2023

Pairs Trading

Lockheed Martin & Northrop Grumman

Faculty mentor: Dr. Doina Bein

Department of Computer Science

California State University Fullerton

BY: SOKHENG TEANG &
HAMID SUHA

Lockheed Martin vs Northrop Grumman

Lockheed Martin:

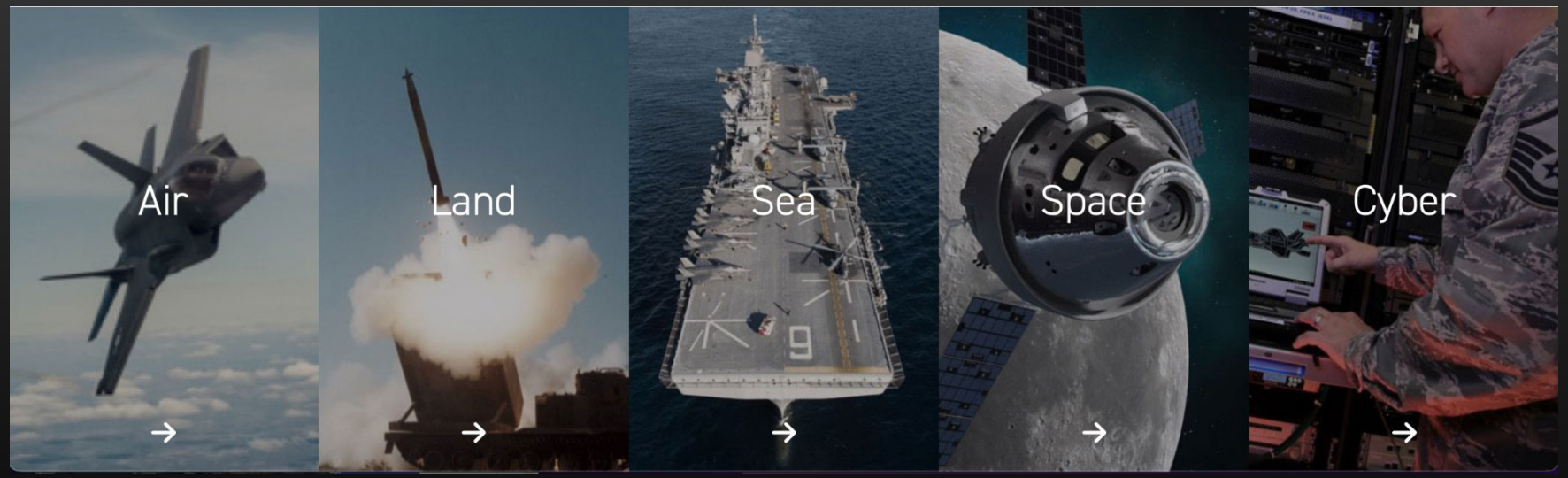
- Founded in 1995, headquartered in Bethesda, Maryland.
- Known for aerospace, defense, arms, security, and advanced technologies.
- Major projects include the F-35 Lightning II fighter jet, the Orion spacecraft, and various missile defense systems.
- In 2020, it was the world's largest defense contractor based on revenue.

Northrop Grumman:

- Established in 1939, headquartered in Falls Church, Virginia.
- Renowned for aerospace, defense, security, and advanced technology solutions.
- Noteworthy projects include the B-2 Spirit stealth bomber, the E-2D Advanced Hawkeye, and the James Webb Space Telescope.
- It is one of the world's top defense contractors, and a significant player in unmanned systems and cybersecurity.

International

Lockheed Martin & Northrop Grumman



Tools and Library

- **Python:** The primary programming language used to conduct the analysis.
- **Alpha Vantage API:** Used to collect the daily adjusted stock prices.
- **Requests Library:** A Python library used for making HTTP requests to the Alpha Vantage API to retrieve data.
- **NumPy Library:** A Python library used for performing numerical computations, including the calculation of logarithmic returns, correlation, and other statistical measures.
- **Matplotlib Library:** A Python library used for creating static, animated, and interactive visualizations in Python. In this case, it's used to plot the stock prices, log returns, and residuals.
- **StatsModels Library:** A Python library used for conducting statistical tests and data exploration. In this context, it's used to perform the Augmented Dickey-Fuller test for cointegration.
- **Jupyter Notebook:** An open-source web application that allows creation and sharing of documents that contain live code, equations, visualizations, and narrative text.

How We Get Stock Price Datas From API

```python
# Global variable
api_key = ############

# Function to get the stock data from API
def get_stock_data(stock_ticker):
    response = requests.get(f'https://www.alphavantage.co/query?function=TIME_SERIES_DAILY_ADJUSTED&symbol={stock_ticker}&apikey={api_key}')
    data = response.json()
    return data
```

```python
# User inputs for two stock tickers
stock_ticker1 = input('Enter the first stock ticker (e.g. AAPL): ')
stock_ticker2 = input('Enter the second stock ticker (e.g. GOOGL): ')

# Get stock data
data1 = get_stock_data(stock_ticker1)
data2 = get_stock_data(stock_ticker2)
```

How We Extract data & Calculate

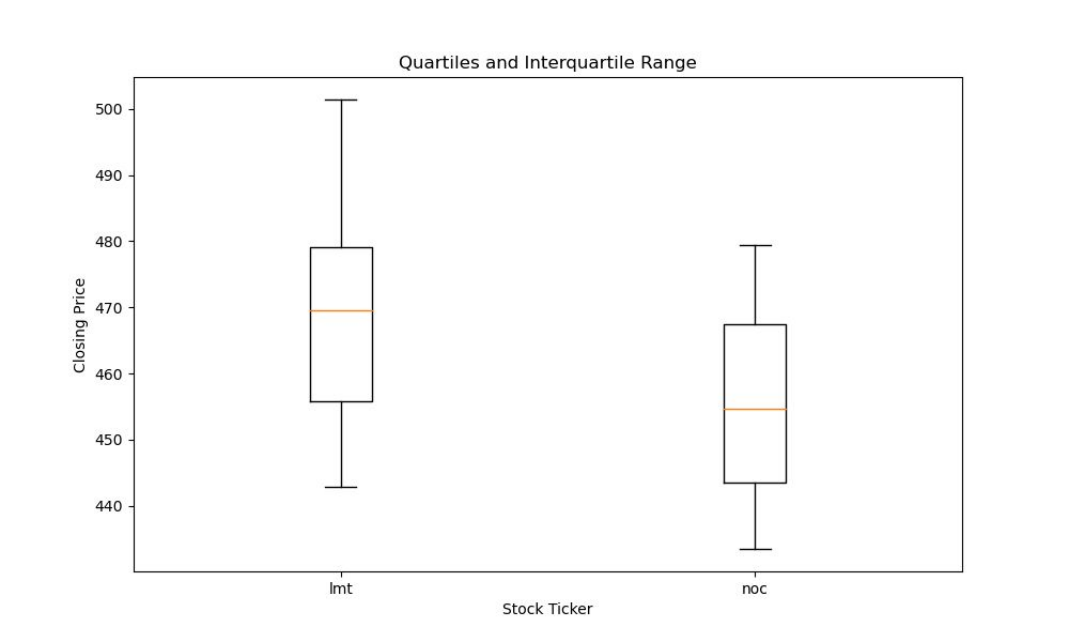
```python
# Function to extract the closing prices from the stock data
def extract_closing_prices(data):
    dates = sorted(data['Time Series (Daily)'].keys(), reverse=True)
    closing_prices = []
    for i, date in enumerate(dates):
        if i >= 120:
            break
        close_price = float(data['Time Series (Daily)'][date]['4. close'])
        closing_prices.append(close_price)
    return closing_prices

# Function to calculate and print statistics
def calc_statistics(closing_prices, stock_ticker):
    mean = np.mean(closing_prices)
    deviations = [(price - mean) for price in closing_prices]
    squared_deviations = [(deviation ** 2) for deviation in deviations]
    sum_squared_deviations = sum(squared_deviations)
    average_squared_deviation = sum_squared_deviations / (len(closing_prices) - 1)
    square_root_avg_squared_deviation = math.sqrt(average_squared_deviation)
    q1 = np.percentile(closing_prices, 25)
    q2 = np.percentile(closing_prices, 50)
    q3 = np.percentile(closing_prices, 75)
    iqr = q3 - q1

    print(f"Mean of {stock_ticker} closing prices: {mean}")
    print(f"Sum of squared deviations from the mean (SS): {sum_squared_deviations}")
    print(f"Average squared deviation from the mean: {average_squared_deviation}")
    print(f"Square root of average squared deviation: {square_root_avg_squared_deviation}")
    print(f"Q1: {q1}")
    print(f"Q2 (Median): {q2}")
    print(f"Q3: {q3}")
    print(f"Interquartile Range (IQR): {iqr}\n")

    return closing_prices
```

Statistic Comparison Output

Mean of lmt closing prices: 468.6396
Sum of squared deviations from the mean (SS): 18738.287584000005
Average squared deviation from the mean: 189.27563216161622
Square root of average squared deviation: 13.757748077415004
Q1: 455.6975
Q2 (Median): 469.53999999999996
Q3: 479.10749999999996
Interquartile Range (IQR): 23.409999999999968

Mean of noc closing prices: 456.23710000000005
Sum of squared deviations from the mean (SS): 14877.15745900001
Average squared deviation from the mean: 150.27431776767685
Square root of average squared deviation: 12.258642574432
Q1: 445.27250000000004
Q2 (Median): 455.36
Q3: 467.4125
Interquartile Range (IQR): 22.139999999999986

How We Plot Data



Lockheed Martin:
Q1: 455.6975
Q2 (Median): 469.53999999999996
Q3: 479.10749999999996
Interquartile Range (IQR): 23.409999999999968

Northrop Grumman:
Q1: 445.27250000000004
Q2 (Median): 455.36
Q3: 467.4125
Interquartile Range (IQR): 22.139999999999986

How We Plot Data

```
# Function to plot the closing prices of both stocks
def plot_prices(closing_prices1, closing_prices2, stock_ticker1, stock_ticker2):
    plt.figure(figsize=(10, 6))
    plt.plot(closing_prices1, color='blue', label=stock_ticker1)
    plt.plot(closing_prices2, color='red', label=stock_ticker2)
    plt.title('Closing Prices')
    plt.xlabel('Days')
    plt.ylabel('Price')
    plt.legend()
    plt.show()
```

Graph Representation Of Closing Price

Cointegration Test

From Dr Bein's slide
"Three main methods can perform cointegration test: Engle-Granger Two-Step Method, Johansen Test, and Maximum Eigenvalue test"
For our team, we perform a simple cointegration test Engle-Granger Two-Step Method

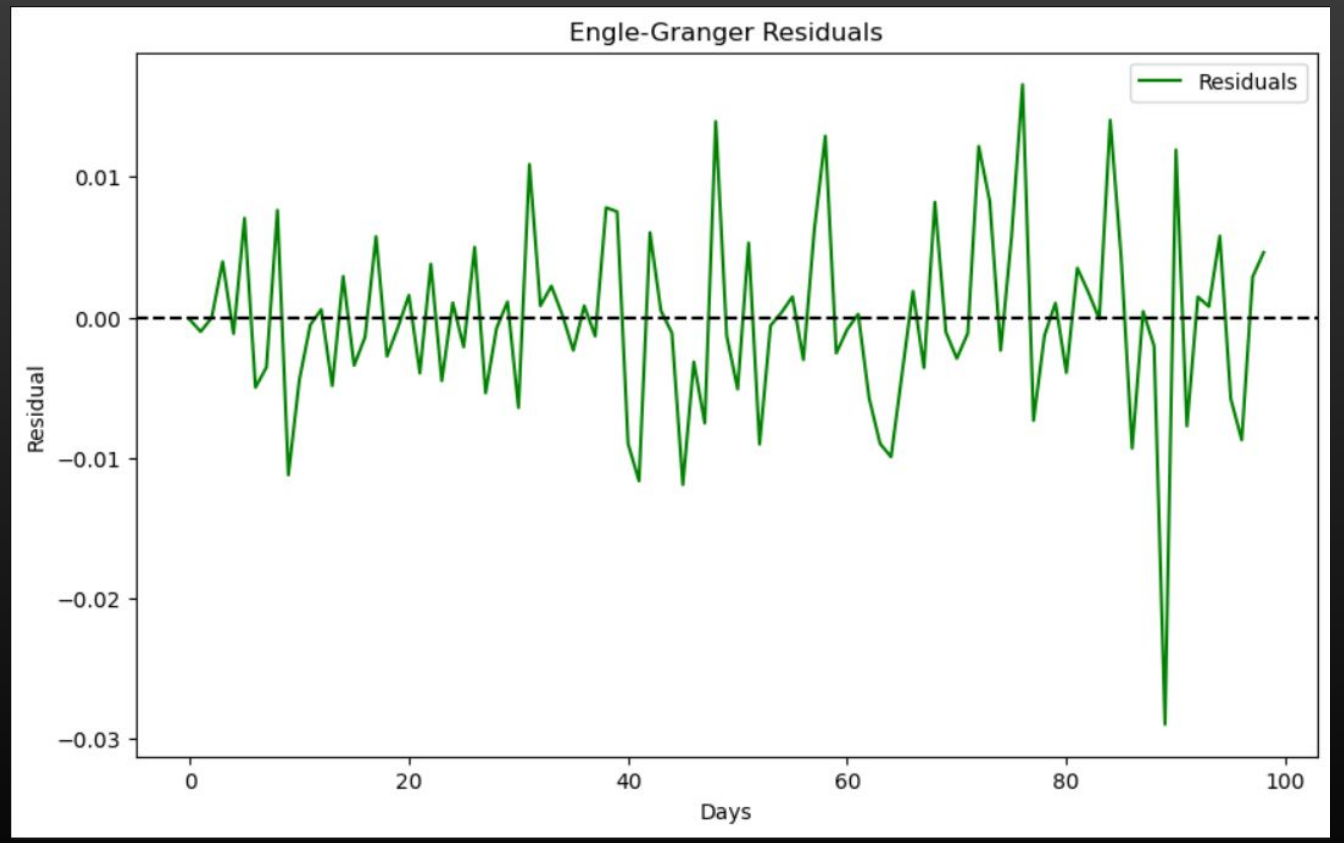
The p-value is: 2.6498936269547676e-20 which is significantly smaller than significance level (0.05) reject the null hypothesis and conclude that the series is stationary.
We found the stocks are cointegrated.

```python
# Function to perform cointegration test and print result
def cointegration_test(log_returns1, log_returns2):
    residuals = log_returns2 - log_returns1
    adf_result = sm.tsa.stattools.adfuller(residuals)
    p_value = adf_result[1]
    is_cointegrated = p_value < 0.05
    print("The stocks are cointegrated.") if is_cointegrated else print("The stocks are not cointegrated.")
    return residuals
```

How We Plot Cointegration Test

```python
# Function to plot the residuals of cointegration test
def plot_residuals(residuals):
    plt.figure(figsize=(10, 6))
    plt.plot(residuals, color='green', label='Residuals')
    plt.axhline(0, color='black', linestyle='--')
    plt.title('Engle-Granger Residuals')
    plt.xlabel('Days')
    plt.ylabel('Residual')
    plt.legend()
    plt.show()
```

How We Plot Data



THANKS

Presented By: SOKHENG TEANG & HAMID SUHA