Time Series in Finance

Author: Sambhav Lamichhane

Advisors: Arkady Shemyakin & Dominick Bolden

Date: August 16, 2024

1. Abstract

This project applied quantitative mathematical techniques to analyze historical financial data with the goal of developing models to predict market behavior and automate trading decisions. Utilizing Python, we conducted time series analysis focused on key financial concepts such as mean reversion, cointegration, and linear regression. Initially, synthetic portfolios of futures contracts were created and tested using linear long-short models and cross-sectional mean-reversion strategies. Following successful simulations, the methodologies were extended to Exchange Traded Funds (ETFs), incorporating risk management and position sizing principles to enhance trading performance. The project culminated in the development of an automated trading system, driven by Application Programming Interfaces (APIs) to execute trades based on real-time data analysis. The results demonstrated that the models effectively identified profitable trading opportunities, while the automation improved efficiency and minimized human biases. This research lays the groundwork for further advancements in automated trading systems and provides a framework for future exploration of more complex financial instruments and strategies.

2. Introduction

In today's fast-paced financial world, the ability to accurately model and predict market behavior is a significant advantage. Financial markets are complex, influenced by countless factors that create patterns and trends over time. To decipher these patterns, quantitative approaches—especially time series analysis—have become essential tools in financial modeling and trading strategies. By examining historical data, time series analysis helps identify trends, cycles, and anomalies that guide informed decision-making.

This project was conducted as part of the Center for Applied Mathematics Industry Opportunity (CAMIO), a program dedicated to applying mathematical research to real-world problems. Among the 12 projects undertaken this year, this was the only one to involve an external advisor, reflecting its unique interdisciplinary approach. The project was developed and executed under the mentorship of Dominick Bolden, an engineer with a keen interest in finance. His expertise and enthusiasm played a crucial role in shaping the research, bridging the gap between mathematical theory and practical finance.

The goal of this project was to apply quantitative mathematical methods to historical financial data to create predictive models for market behavior. Using Python, we focused on time series analysis, incorporating key concepts like mean reversion, cointegration, and linear regression. Our work initially concentrated on futures contracts, where we constructed synthetic portfolios designed to test the effectiveness of our models. We employed linear long-short strategies and cross-sectional mean-reversion techniques to identify price discrepancies and generate trading signals.

Building on our initial successes with futures, we expanded the research to include Exchange Traded Funds (ETFs). This required integrating risk management strategies, such as position sizing and stop-loss mechanisms, to better navigate the complexities of the ETF market. This phase of the project demonstrated the versatility of our models and their applicability across different types of financial instruments.

A key aspect of this project was the development of an automated trading system. By automating market monitoring, data analysis, and trade execution, we aimed to reduce human bias and improve the consistency and efficiency of trading decisions. This automation was achieved through the use of Application Programming Interfaces (APIs), which allowed us to execute trades based on real-time data, ensuring that our trading decisions were grounded in rigorous, data-driven analysis.

This project contributes to the growing field of automated trading systems by offering a practical framework that can be built upon in future research. The models we developed provide a solid foundation for further exploration into more complex financial instruments and strategies, potentially leading to even more refined and effective trading systems.

3. Problem Statement and Research Goals

The primary challenge addressed by this research was the need for accurate and reliable models that can predict market behavior and guide automated trading strategies. Financial markets are inherently volatile, and traditional methods often fail to account for the complexities of time series data. This project sought to bridge that gap by applying quantitative mathematical techniques to analyze historical financial data and develop predictive models capable of informing trading decisions.

To address this problem, the research focused on two key goals:

i. **Developing Robust Predictive Models:** Leveraging time series analysis, mean reversion, and cointegration to create models that can accurately predict market movements.
ii. **Automating the Trading Process:** Implementing an automated trading system using Python and APIs to execute trades based on real-time data analysis.

4. Literature Review

To lay a strong foundation for our research, we extensively referred to *Algorithmic Trading: Winning Strategies and Their Rationale* by Ernest P. Chan. This book provided a comprehensive overview of the key concepts required for our project, particularly those related to mean reversion strategies. Chan's work is widely regarded in the field of quantitative finance, offering both theoretical insights and practical applications, which were crucial to our study.

The book was instrumental in helping us understand and implement mean reversion strategies, which are central to our time series analysis. Chan's detailed explanations, coupled with ready-to-use code examples, allowed us to quickly adapt these strategies to our specific needs. The clarity and depth of the material provided not only a thorough understanding of the concepts but also a significant head start in developing our models. This saved us substantial time and effort, allowing us to focus more on refining and expanding our strategies rather than starting from scratch.

Chan's work provided a crucial boost to our project, enabling us to effectively design and implement our trading models. The insights gained from this book were directly applied in our time series analysis, model development, and the eventual automation of our trading system. This resource was particularly valuable as it bridged the gap between academic theory and real-world trading, offering a practical framework that aligned perfectly with the goals of our research.

5. Concepts and Methodology

i. Time Series Analysis:

Time series analysis is the foundation of our research, focusing on the examination of data points collected at consistent intervals over time. In financial markets, time series data, such as daily stock prices or interest rates, provide critical insights into market behavior. By analyzing this data, we can identify underlying patterns, trends, and seasonal effects that are not immediately apparent. Time series models are essential for forecasting future market movements, allowing traders to make informed decisions based on past behavior. Techniques such as autoregressive models (AR), moving average models (MA), and their combination (ARMA/ARIMA) are commonly used in time series analysis to model and predict future values.

ii. Mean Reversion:

The concept of mean reversion is a cornerstone in financial modeling and trading. It is based on the idea that asset prices and returns fluctuate around a long-term mean or average. When prices deviate significantly from this mean, mean reversion theory suggests they will eventually return to their historical average. This tendency is especially evident in markets with strong cyclical behavior, such as commodities and interest rates. In trading, mean reversion strategies involve identifying these deviations and taking positions that anticipate a return to the mean, allowing traders to profit from the correction. For example, in pairs trading, a trader might buy an undervalued asset and sell an overvalued one, betting that their prices will converge. The speed at which this convergence happens is quantified by the half-life of mean reversion, a key metric calculated using the Ornstein-Uhlenbeck process. This calculation helps traders determine the optimal timing for entering and exiting trades.

iii. Stationarity and Tests for Stationarity:

Stationarity is a crucial property in time series analysis, indicating that the statistical properties of a series—such as its mean, variance, and autocorrelation—remain constant over time. Stationarity is important because many econometric models rely on the assumption that the underlying time series is stationary. Non-stationary data, where statistical properties change over time, can lead to misleading results and poor forecasts. To assess stationarity, we employed several tests, including the Augmented Dickey-Fuller (ADF) Test and the Phillips-Perron (PP) Test. The ADF test extends the basic Dickey-Fuller test by incorporating lagged difference terms to account for autocorrelation. $\Delta y(t) = \lambda y(t-1) + \mu + \beta t + \alpha_1 \Delta y(t-1) + \cdots + \alpha_k \Delta y(t-k) + \epsilon_t$

The PP test, on the other hand, adjusts for serial correlation and heteroskedasticity without adding lagged differences, providing a complementary robustness check. If a time series is found to be non-stationary, we use techniques like differencing—where the differences between consecutive data points are analyzed—to transform it into a stationary series, enabling accurate modeling and forecasting.

iv. Stationarity test that we ran:

	Date	Close/Last	Volume	Open	High	Low
0	2024-07-16	$0.64	31100.0	$0.60	$0.64	$0.60
1	2024-07-15	$0.61	56200.0	$0.64	$0.64	$0.61
2	2024-07-12	$0.62	15150.0	$0.62	$0.62	$0.61
3	2024-07-11	$0.64	31900.0	$0.62	$0.64	$0.62
4	2024-07-10	$0.62	22888.0	$0.64	$0.64	$0.62
...	...	...	...	...	...	...
2511	2014-07-23	$0.98	NaN	$0.98	$0.98	$0.98
2512	2014-07-22	$0.98	NaN	$0.98	$0.98	$0.98
2513	2014-07-21	$0.98	NaN	$0.98	$0.98	$0.98
2514	2014-07-18	$0.98	NaN	$0.98	$0.98	$0.98
2515	2014-07-17	$0.98	NaN	$0.98	$0.98	$0.98

[2516 rows x 6 columns]


```
ADF Statistic: -2.151500
p-value: 0.224317
Critical Values:
        1%: -3.433
        5%: -2.863
        10%: -2.567
```

This image presents an analysis of China Petroleum Futures, focusing on testing whether the price data is stationary over time using the Augmented Dickey-Fuller (ADF) test. The analysis includes a preview of the data, charts for open and close prices, and the results of the stationarity test.

Starting with the data preview, we see that the dataset spans from July 2024 to July 2014, comprising 2,516 rows of daily trading information. The columns include the date, closing price, trading volume, opening price, highest price, and lowest price for each day.

The first chart illustrates the open prices for China Petroleum Futures over this period. The prices show significant ups and downs, with a notable peak around 2015, a sharp decline leading into 2020, and then a gradual recovery through 2023 and 2024.

The second chart tracks the close prices, which follow a similar pattern. Like the open prices, the close prices also peaked in 2015, dropped sharply around 2020, and then recovered, although they remain volatile.

The key focus of this analysis is the ADF test, which assesses whether the time series (in this case, the price data) is stationary. Stationarity is important in time series analysis because it implies that the statistical properties like mean and variance do not change over time, making the data more predictable.

The ADF test results show an ADF statistic of -2.151500, with a p-value of 0.224317. These values suggest that the time series is non-stationary, meaning that its statistical properties change over time. Specifically, because the ADF statistic is not sufficiently negative compared to the critical values, and the p-value is above the threshold of 0.05, we cannot reject the null hypothesis that the series has a unit root. In simpler terms, the data is not stable, and its behavior changes over time, which can make forecasting more challenging.

In summary, the analysis reveals that the price data for China Petroleum Futures is non-stationary, indicating that its trends and volatility vary over the years. This finding is essential for anyone looking to model or predict future prices using this data, as non-stationary data often requires special handling in time series analysis.

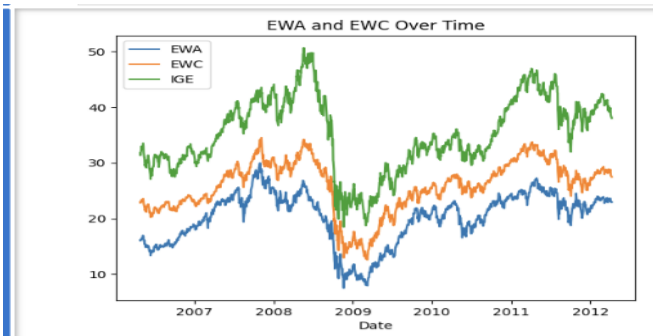
v. Cointegration:
Cointegration occurs when a linear combination of two or more non-stationary time series is stationary, indicating a long-term equilibrium relationship between the series despite short-term deviations. This is particularly valuable in pairs trading, where traders exploit temporary divergences between closely related assets. To test for cointegration, we used both the Engle-Granger Two-Step Method and the Johansen Test. The Engle-Granger method involves regressing one series on another and testing the residuals for stationarity. The Johansen test, a more advanced technique, assesses multiple time series simultaneously and determines the number of cointegrating vectors, providing a more comprehensive analysis when dealing with multiple assets.

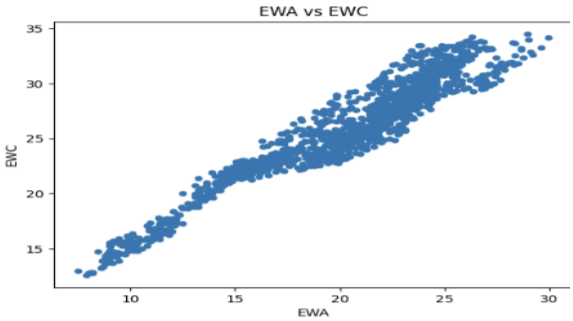
vi. Cointegration Test Results: A pair trading strategy was evaluated using two ETFs, EWA and EWC, to assess their cointegration. The ADF test showed a statistic of -2.

This image presents an analysis of a pair trading strategy involving two ETFs: EWA. The analysis includes multiple charts, a regression analysis, and key statistical results to evaluate the relationship between these two ETFs and the performance of the trading strategy.

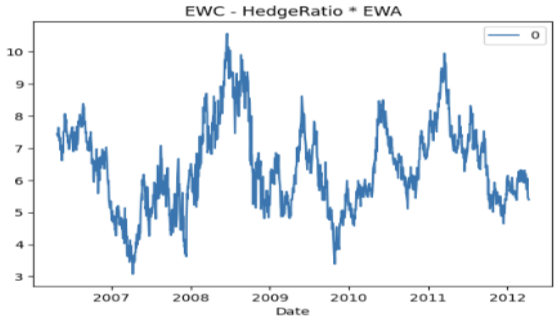
At the top, the first chart compares the closing prices of EWA (in blue) and EWC (in orange) from 2015 to 2024. Both ETFs show a similar overall trend, reflecting a high correlation, which is crucial for pair trading. However, there are periods where the two ETFs diverge, creating potential trading opportunities.

The second chart is a scatter plot that directly compares the closing prices of EWA and EWC on a daily basis. The strong linear relationship visible in the plot further confirms that these two






```
Intercept    6.411331
EWA          0.962429
dtype: float64
hedgeRatio=0.962429
```




```
t-statistic=-3.063528
pvalue=0.095866
Critical Values: [-3.90376106 -3.34020915 -3.04728056]
[19.98321869  3.98276124]
[[13.4294 15.4943 19.9349]
 [ 2.7055  3.8415  6.6349]]
[16.00045745  3.98276124]
[[12.2971 14.2639 18.52  ]
 [ 2.7055  3.8415  6.6349]]
[34.42862022 17.53171895  4.47102054]
[[27.0669 29.7961 35.4628]
 [13.4294 15.4943 19.9349]
 [ 2.7055  3.8415  6.6349]]
[16.89690127 13.06069841  4.47102054]
[[18.8928 21.1314 25.865 ]
 [12.2971 14.2639 18.52  ]
 [ 2.7055  3.8415  6.6349]]
[0.01121626 0.00868086 0.00298021]
[[ 0.7599635  -0.11204898  0.0789828 ]
 [-1.04602749 -0.5796762   0.26467204]
 [ 0.22330592  0.53159644 -0.09515547]]
Intercept   -0.115768
ylag        -0.030586
dtype: float64
halflife=22.662578 days
```


```
APR=0.125739 Sharpe=1.402653
```

ETFs move closely together, justifying the use of a pair trading strategy. A regression analysis is performed on this data, resulting in an intercept of -7.473607 and a slope (hedge ratio) of 1.688872. This hedge ratio indicates how much EWC should be bought or shorted relative to EWA in the pair trading strategy.

The third chart shows the spread between EWC and the hedge-ratio-adjusted EWA (EWC - 1.688872 * EWA) over time. Ideally, this spread should oscillate around zero, allowing the strategy to capitalize on mean reversion. However, the spread shows periods of divergence and convergence, with a clear downward trend since 2021, suggesting that the relationship may be weakening.

The ADF (Augmented Dickey-Fuller) test results are shown below this chart. The ADF test evaluates whether the spread is stationary, which is essential for a successful pair trading strategy. The ADF statistic is -2.446510, with a p-value of 0.303155. Since the p-value is higher than 0.05 and the ADF statistic is not sufficiently negative compared to the critical values, the spread is likely non-stationary. This indicates that the relationship between EWA and EWC may not be stable over time, which can undermine the effectiveness of the strategy.

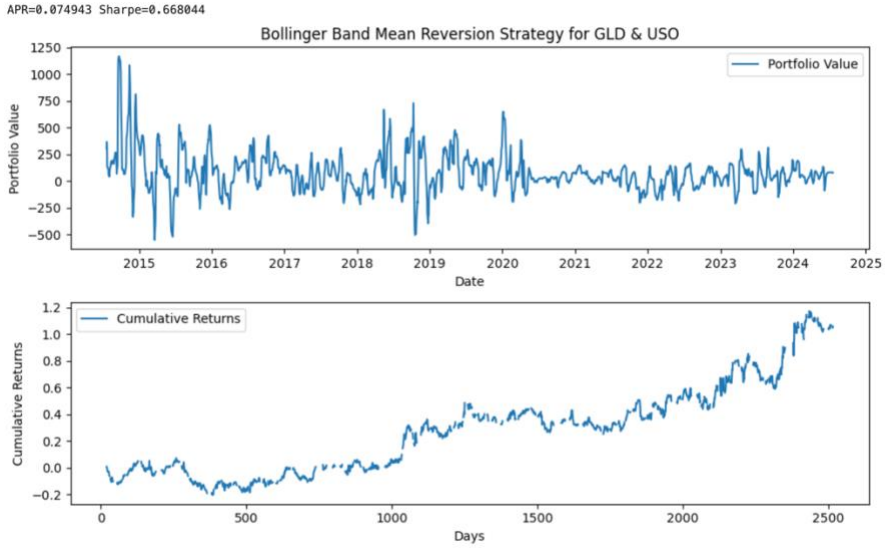
Finally, the last chart tracks the cumulative returns of the pair trading strategy from 2015 to 2024. The cumulative returns initially increase but then steadily decline, particularly from 2018 onward. By 2024, the strategy's returns have nearly eroded to zero, suggesting that it has been underperforming for several years.

The analysis concludes with key performance metrics: an annualized percentage return (APR) of 7.05% and a Sharpe ratio of 0.535122. While these figures suggest moderate returns relative to risk, the declining cumulative returns highlight significant challenges with the strategy's long-term viability.

In summary, this analysis reveals that although EWA and EWC are closely correlated, the non-stationarity of the spread and the declining cumulative returns suggest that the pair trading strategy may not be as effective as initially expected, particularly in recent years.

vi. Bollinger Band Mean Reversion Strategy

This chart illustrates the performance of a Bollinger Band Mean Reversion Strategy applied to a portfolio consisting of GLD (SPDR Gold Shares) and USO (United States Oil Fund) from 2014 to 2024. The analysis is presented in two sections: the portfolio value over time and the cumulative returns.



Top Panel: Portfolio Value Over Time

The portfolio value experiences significant fluctuations throughout the period, with sharp gains and losses visible in several instances. This volatility is typical of a mean reversion strategy, where the approach involves buying when prices are perceived to be low (below the lower Bollinger Band) and selling when they are high (above the upper Bollinger Band). The strategy assumes that prices will revert to their average levels, but this can lead to considerable swings when markets trend strongly without reverting quickly.

Bottom Panel: Cumulative Returns

The cumulative returns chart shows that the strategy initially faced losses but gradually improved over time. Starting around 2018, there is a noticeable upward trend, indicating that the strategy became more successful, ultimately achieving positive overall returns by 2024. This suggests that, despite early challenges, the mean reversion strategy was profitable in the long run.

Bollinger Bands Explained

Bollinger Bands are a technical analysis tool consisting of three lines: a moving average (middle band) and two bands above and below it, representing standard deviations from the mean. The strategy buys when prices fall below the lower band and sells when they rise above the upper band, based on the expectation that prices will revert to the mean.

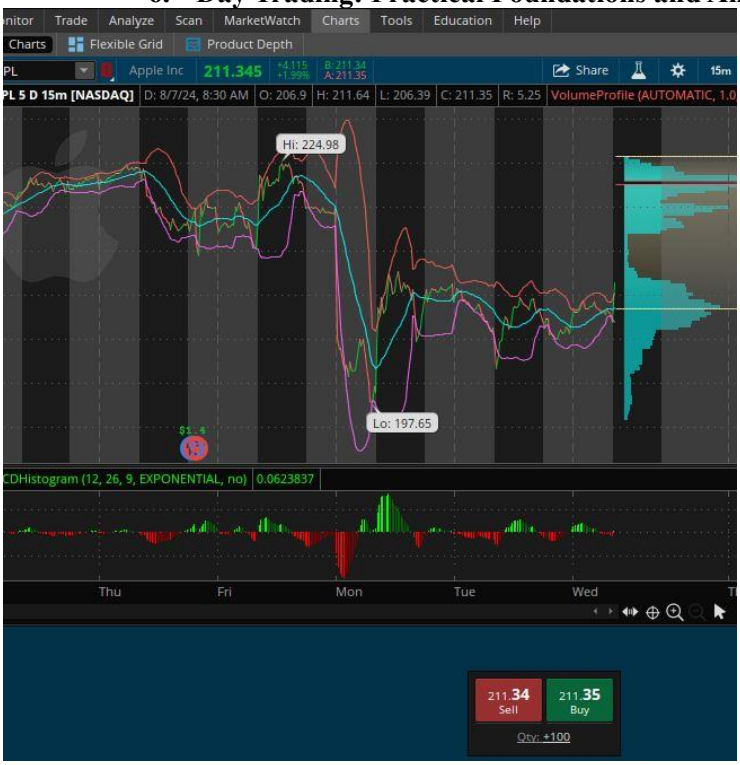
Performance Metrics

The strategy delivered an Annualized Percentage Return (APR) of 7.49%, indicating a modest but consistent return over time. The Sharpe Ratio of 0.668 reflects a reasonable balance between risk and return, suggesting that the strategy managed to achieve profits while managing risk effectively.

Summary

In summary, this Bollinger Band Mean Reversion Strategy for GLD and USO shows that while the strategy experienced significant volatility, it ultimately generated positive returns, particularly in the later years. This highlights the potential for mean reversion strategies to be profitable over the long term, even if they encounter short-term challenges along the way.

6. Day Trading: Practical Foundations and Analysis



To gain a deeper understanding of real-time market dynamics, we engaged in day trading as part of our research before transitioning to full automation using an API. This hands-on approach allowed us to apply theoretical concepts, refine our strategies, and evaluate the effectiveness of various technical indicators in a live trading environment. The experience gained was instrumental in bridging the gap between manual trading and automated systems.

Case Study: Apple Inc. (AAPL) Stock Analysis

One notable example from our day trading sessions involved analyzing Apple Inc. (AAPL) on the Thinkorswim (TOS) platform. Our analysis combined several technical indicators, leading to informed trading decisions.

- **Initial Observation:** The price of Apple spiked above the upper Bollinger Band, which typically signals overbought conditions and could indicate a good opportunity for short selling.
- **Technical Indicators:**

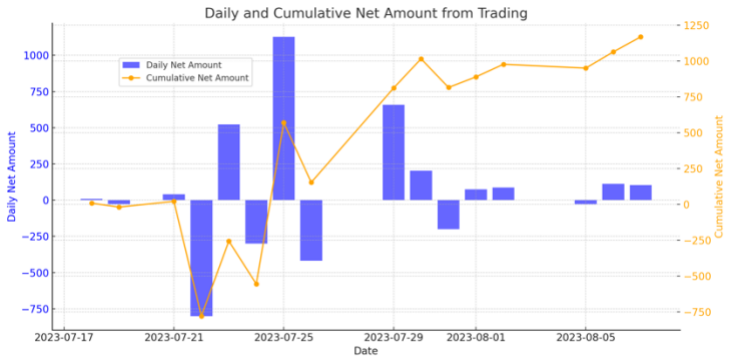
MACD Histogram: The histogram was trending upward with green bars, indicating strengthening bullish momentum.

Middle Moving Average: The moving average was solidly upward, suggesting increased trading volume after a recent market dip.

- **Trading Decision:** Despite the overbought signal from the Bollinger Bands, the upward trends in the MACD and volume profile led us to take a contrarian approach and buy. The reasoning was that the momentum indicated by the MACD and moving average would likely sustain the price increase.
- **Outcome:** Within five minutes of executing the buy order, the price of Apple rose, confirming our decision to prioritize the MACD and volume profile trends over the Bollinger Bands signal alone.

Result

Day trading provided practical insights into the application of technical analysis. By analyzing indicators like Bollinger Bands and MACD, we refined our understanding of market behavior, which directly informed the development of our automated trading systems. The success of these strategies



during manual trading sessions emphasized the importance of using multiple indicators for better decision-making.

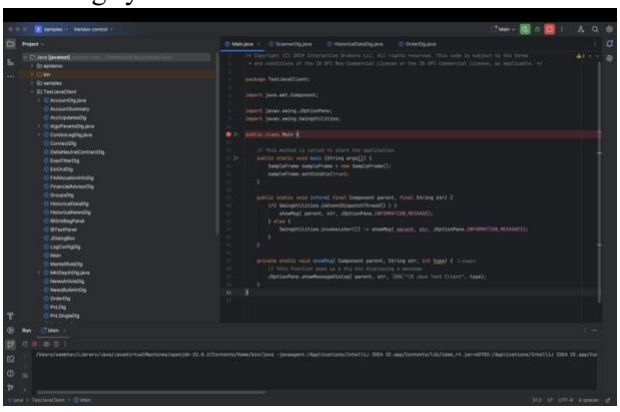
The accompanying graph shows the daily and cumulative net amounts from trading. The fluctuations in daily net amounts (blue bars) reflect the challenges of day trading, while the overall upward trend in the cumulative net amount (orange line) highlights the effectiveness of our approach.

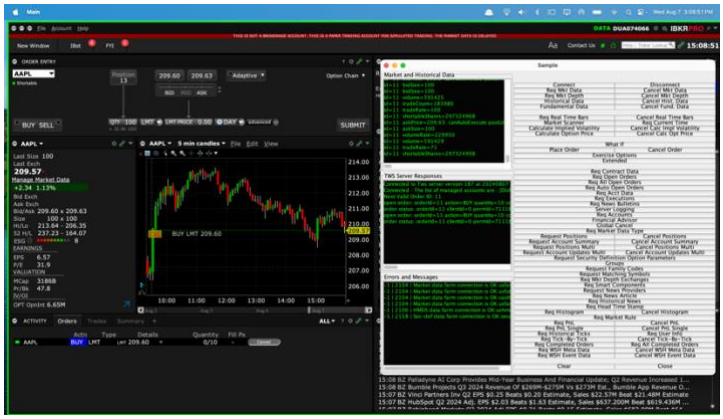
7. API and Automation

In the realm of automated trading, Application Programming Interfaces (APIs) are indispensable tools that enable seamless communication between different software applications. Acting as a bridge between platforms, an API facilitates the transfer of data and execution of commands, making it essential for developing an automated trading system where speed and precision are critical.

The Need for Automation:
Automating the trading process offers several key advantages. Firstly, it eliminates human biases, removing the emotional and psychological factors that can cloud judgment and lead to inconsistent decisions. Secondly, automation allows for the rapid execution of trades, capitalizing on fleeting market opportunities that human traders might miss. Finally, automation levels the playing field, giving individual traders access to the same tools and speed as large financial institutions, enabling them to compete more effectively.

Implementation Using Interactive Brokers API:
For this project, we utilized the API provided by Interactive Brokers (IBKR), a leading brokerage firm known for its advanced trading platforms. After extensive trials and iterations, we successfully developed a software solution that interfaces with IBKR's platform. This accomplishment represents a significant breakthrough in our research, as one of the primary objectives was to create a functional automated trading system.





The images illustrate two critical components of our setup:

- **Code Development:** The first image shows the code we developed using the Interactive Brokers API package. This code serves as the backbone of our automated trading system, managing everything from data retrieval to trade execution.
- **Trading Platform and API Interface:** The second image displays Interactive Brokers' trading platform. The trading station (in black) is where trades are executed, either manually or

automatically, while the API interface (in white) enables the automation of these processes based on real-time data analysis.

Result:
Achieving this level of automation marks a significant milestone in our research. While this is only the first step towards a fully integrated system, it demonstrates the feasibility and potential of automated trading through APIs. As we continue to refine our models and software, this breakthrough will serve as the foundation for future advancements in our automated trading strategies.

8. Conclusion

This research applied quantitative mathematical techniques to historical financial data to develop predictive models and automate trading decisions. By leveraging time series analysis, mean reversion, and cointegration, we constructed robust models that offered valuable insights into market behavior. Initially, these models were tested on synthetic portfolios of futures contracts and were later extended to Exchange Traded Funds (ETFs), demonstrating their versatility across different financial instruments. The development and implementation of an automated trading system using Interactive Brokers' API marked a significant milestone, enabling us to execute trades based on real-time data analysis. This automation enhanced trading efficiency and minimized human biases, leading to more consistent and objective outcomes. Overall, the research established a solid foundation for advancing automated trading systems, highlighting the potential of combining rigorous quantitative methods with modern technological tools.

9. Future Directions

The success of this project paves the way for several promising avenues for future research and development. One potential direction is to expand the scope of the models to include more complex financial instruments, such as options or derivatives, to further enhance the robustness of trading strategies. Additionally, integrating machine learning algorithms with the existing models could provide deeper insights and improve predictive accuracy. Exploring alternative data sources, such as sentiment analysis from news or social media, could also offer new perspectives on market trends. Continuous refinement of the automated trading system, with a focus on optimizing execution speed and reducing latency, is another important direction. Finally, back testing the models under different market conditions, including periods of high volatility or economic downturns, would provide a more comprehensive evaluation of their performance and resilience. These future directions will build on the achievements of this research and contribute to the broader field of quantitative finance and automated trading.

10. Bibliography

a. Chan, E. P. (2013). *Algorithmic Trading: Winning Strategies and Their Rationale*. Wiley.
b. Hacking The Markets. "Interactive Brokers TWS API Tutorial." Accessed August 2, 2024.

11. Appendix

Stationary Test Codes:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller

# Function to read the file and handle potential errors
def read_data(file_path):
    try:
        # Load the data with the correct delimiter
        df = pd.read_table(file_path, delimiter='\t', header=0)
        return df
    except pd.errors.EmptyDataError:
        raise ValueError(f"No data found in file: {file_path}")
    except Exception as e:
        raise ValueError(f"An error occurred while reading the file: {e}")

# Load the data
file_path = 'China_Petroleum.txt'
df = read_data(file_path)

# Ensure the 'Date' column is parsed as datetime if it exists
if 'Date' in df.columns:
    df['Date'] = pd.to_datetime(df['Date'])

# Print the DataFrame to see its structure
print(df)

# Check if 'Open' and 'Close/Last' columns exist for analysis
```

```
if 'Open' not in df.columns:
    raise ValueError("The 'Open' column does not exist in the data.")
if 'Close/Last' not in df.columns:
    raise ValueError("The 'Close/Last' column does not exist in the data.")

# Extract 'Open' and 'Close/Last' columns for analysis
open_prices = df['Open'].str.replace('$', '', regex=False).astype(float)
close_prices = df['Close/Last'].str.replace('$', '', regex=False).astype(float)

# Determine x-axis (index or date)
if 'Date' in df.columns:
    x_axis = df['Date']
    x_label = 'Date'
else:
    x_axis = df.index
    x_label = 'Index (row number)'

# Plot the 'Open' prices
plt.figure()
plt.plot(x_axis, open_prices)
plt.title('Open Prices')
plt.xlabel(x_label)
plt.ylabel('Open Price')
plt.show()

# Plot the 'Close/Last' prices
plt.figure()
plt.plot(x_axis, close_prices)
plt.title('Close Prices')
plt.xlabel(x_label)
```

```
plt.ylabel('Close Price')
plt.show()

# Perform ADF Test for mean reversion on 'Close/Last' prices
results = adfuller(close_prices, maxlag=1, regression='c', autolag=None)
print('ADF Statistic: %f' % results[0])
print('p-value: %f' % results[1])
print('Critical Values:')
for key, value in results[4].items():
    print('\t%s: %.3f' % (key, value))
```

Bollinger Band Mean Reversion Strategy code:

```
# Bollinger Band Mean Reversion Strategy

import numpy as np
import pandas as pd
import statsmodels.formula.api as sm
import statsmodels.tsa.stattools as ts
import matplotlib.pyplot as plt  # Import matplotlib for plotting

# Read the data from the two files
df1 = pd.read_csv('GLD.txt', delimiter='\t')  # Assuming GLD.txt is tab-delimited
df2 = pd.read_csv('USO.txt', delimiter='\t')  # Assuming USO.txt is also tab-delimited

# Ensure the date columns are in datetime format and set them as indices
df1['Date'] = pd.to_datetime(df1['Date'], format='%m/%d/%y').dt.date
df2['Date'] = pd.to_datetime(df2['Date'], format='%m/%d/%y').dt.date

# Extract 'Close/Last' columns and rename them to 'GLD' and 'USO'
df1 = df1[['Date', 'Close/Last']].rename(columns={'Close/Last': 'GLD'})
```

```
df2 = df2[['Date', 'Close/Last']].rename(columns={'Close/Last': 'USO'})

# Remove the dollar sign and convert to float
df1['GLD'] = df1['GLD'].replace('[\$,]', '', regex=True).astype(float)
df2['USO'] = df2['USO'].replace('[\$,]', '', regex=True).astype(float)

# Merge the two dataframes on the Date column
df = pd.merge(df1, df2, on='Date')
df.set_index('Date', inplace=True)

# Handle missing values
df.dropna(inplace=True)

lookback = 20
hedgeRatio = np.full(df.shape[0], np.nan)
for t in np.arange(lookback, len(hedgeRatio)):
    regress_results = sm.ols(formula="USO ~ GLD", data=df[(t-lookback):t]).fit()  # Note this can deal with NaN in top row
    hedgeRatio[t-1] = regress_results.params[1]

yport = np.sum(ts.add_constant(-hedgeRatio)[:, [1, 0]] * df[['GLD', 'USO']], axis=1)

# Plot the yport series with better labeling
plt.figure(figsize=(10, 6))
plt.subplot(2, 1, 1)
plt.plot(df.index, yport, label='Portfolio Value')
plt.title('Bollinger Band Mean Reversion Strategy for GLD & USO')
plt.xlabel('Date')
plt.ylabel('Portfolio Value')
plt.legend()
```

```
# Bollinger band strategy
entryZscore = 1
exitZscore = 0

MA = yport.rolling(lookback).mean()
MSTD = yport.rolling(lookback).std()
zScore = (yport - MA) / MSTD

longsEntry = zScore < -entryZscore
longsExit = zScore > -entryZscore

shortsEntry = zScore > entryZscore
shortsExit = zScore < exitZscore

numUnitsLong = np.zeros(longsEntry.shape)
numUnitsLong[:] = np.nan

numUnitsShort = np.zeros(shortsEntry.shape)
numUnitsShort[:] = np.nan

numUnitsLong[0] = 0
numUnitsLong[longsEntry] = 1
numUnitsLong[longsExit] = 0
numUnitsLong = pd.DataFrame(numUnitsLong)
numUnitsLong.fillna(method='ffill', inplace=True)

numUnitsShort[0] = 0
numUnitsShort[shortsEntry] = -1
numUnitsShort[shortsExit] = 0
```

```
numUnitsShort = pd.DataFrame(numUnitsShort)

numUnitsShort.fillna(method='ffill', inplace=True)


numUnits = numUnitsLong + numUnitsShort

positions = pd.DataFrame(np.tile(numUnits.values, [1, 2]) * ts.add_constant(-hedgeRatio)[:, [1, 0]] *
df[['GLD', 'USO']].values)

pnl = np.sum((positions.shift().values) * (df[['GLD', 'USO']].pct_change().values), axis=1)  # daily P&L
of the strategy

ret = pnl / np.sum(np.abs(positions.shift()), axis=1)

cumulative_returns = np.cumprod(1 + ret) - 1


# Plot the cumulative returns with better labeling

plt.subplot(2, 1, 2)

plt.plot(cumulative_returns, label='Cumulative Returns')

plt.xlabel('Days')

plt.ylabel('Cumulative Returns')

plt.legend()


# Print APR and Sharpe Ratio

APR = np.prod(1 + ret) ** (252 / len(ret)) - 1

Sharpe = np.sqrt(252) * np.mean(ret) / np.std(ret)

print('APR=%f Sharpe=%f' % (APR, Sharpe))


plt.tight_layout()

plt.show()  # Show the plot
```

Code for the API:

```
package TestJavaClient;

import java.awt.Component;
```

```
import javax.swing.JOptionPane;
import javax.swing.SwingUtilities;

public class Main {

   // This method is called to start the application
   public static void main (String args[]) {
      SampleFrame sampleFrame = new SampleFrame();
      sampleFrame.setVisible(true);
   }

   public static void inform( final Component parent, final String str) {
      if( SwingUtilities.isEventDispatchThread() ) {
         showMsg( parent, str, JOptionPane.INFORMATION_MESSAGE);
      } else {
         SwingUtilities.invokeLater(() -> showMsg( parent, str,
JOptionPane.INFORMATION_MESSAGE));
      }
   }

   private static void showMsg( Component parent, String str, int type) {
      // this function pops up a dlg box displaying a message
      JOptionPane.showMessageDialog( parent, str, "IB Java Test Client", type);
   }
}
```