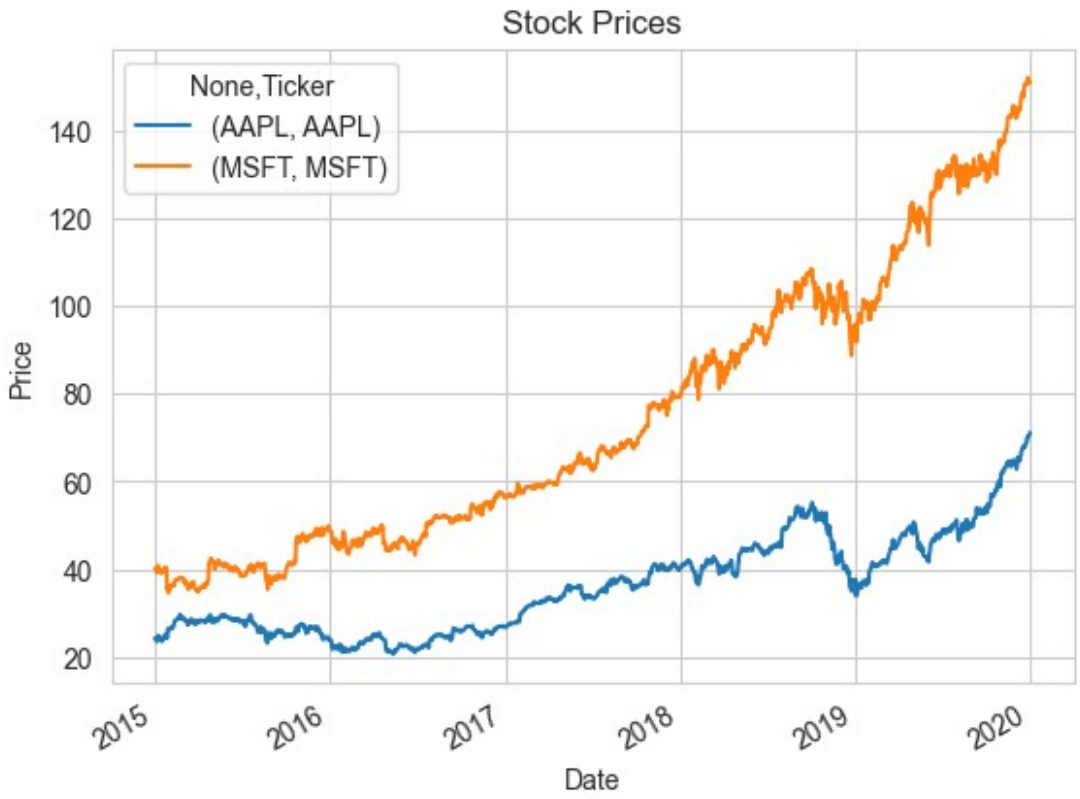
```
import numpy as np
import pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt
import seaborn as sns
from pykalman import KalmanFilter
from statsmodels.tsa.stattools import coint

sns.set_style("whitegrid")

# Step 1: Fetching the Data
stock1 = "AAPL"
stock2 = "MSFT"
start_date = "2015-01-01"
end_date = "2020-01-01"

# Fetch data
data1 = yf.download(stock1, start=start_date, end=end_date)["Adj
Close"]
data2 = yf.download(stock2, start=start_date, end=end_date)["Adj
Close"]

# Check if data was downloaded correctly
if data1 is None or data2 is None:
    raise ValueError("Stock data could not be fetched. Check the stock
symbols or the date range.")

# Merge data into a single DataFrame
prices = pd.concat([data1, data2], axis=1, keys=[stock1,
stock2]).dropna()

# Plot the raw prices
plt.figure(figsize=(12, 6))
prices.plot(title="Stock Prices")
plt.ylabel("Price")
plt.show()
```

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

```
<Figure size 1200x600 with 0 Axes>
```

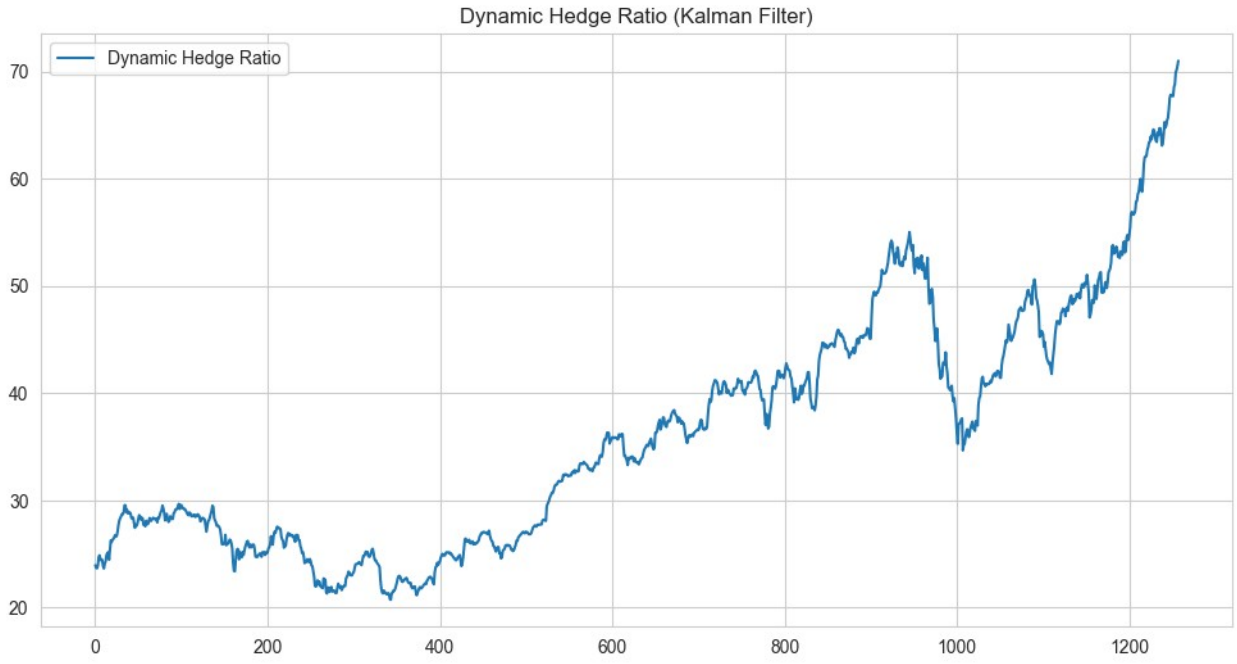



```
score, pvalue, _ = coint(prices[stock1], prices[stock2])
print(f"Cointegration test p-value: {pvalue:.4f}")
if pvalue < 0.05:
    print("The stocks are cointegrated.")
else:
    print("The stocks are not cointegrated.")

# Data checks
print(prices.head())
print(prices.isna().sum())
print(prices.shape)
print(prices.dtypes)
```

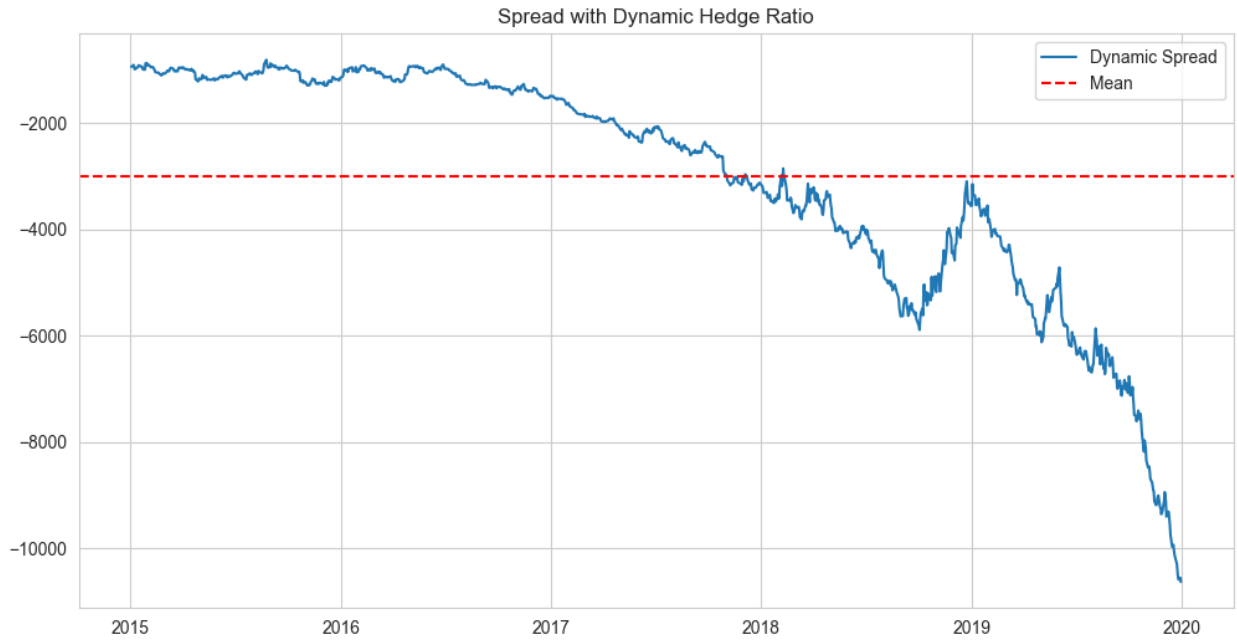
```
Cointegration test p-value: 0.5855
The stocks are not cointegrated.
                AAPL       MSFT
Ticker          AAPL       MSFT
Date
2015-01-02  24.347172  40.152489
2015-01-05  23.661263  39.783249
2015-01-06  23.663492  39.199329
2015-01-07  23.995312  39.697380
2015-01-08  24.917273  40.865196
      Ticker
AAPL  AAPL      0
```

```
MSFT  MSFT      0
dtype: int64
(1258, 2)
      Ticker
AAPL  AAPL      float64
MSFT  MSFT      float64
dtype: object

stock1_prices = pd.to_numeric(prices.loc[:, (stock1,
stock1)].squeeze(), errors='coerce')
stock2_prices = pd.to_numeric(prices.loc[:, (stock2,
stock2)].squeeze(), errors='coerce')

# Create a new DataFrame with just the prices we need
analysis_df = pd.DataFrame({
    stock1: stock1_prices,
    stock2: stock2_prices
})

# Now drop NaN values
analysis_df = analysis_df.dropna()
hedge_ratio = np.polyfit(analysis_df[stock2].values,
analysis_df[stock1].values, 1)[0]

# Calculate spread and z-score
spread = analysis_df[stock1] - hedge_ratio * analysis_df[stock2]
spread_zscore = (spread - spread.rolling(window=20).mean()) /
spread.rolling(window=20).std()

# Plot spread
plt.figure(figsize=(12, 6))
plt.plot(spread, label="Spread")
plt.axhline(spread.mean(), color='red', linestyle='--', label="Mean")
plt.legend()
plt.title("Spread between Stocks")
plt.show()
```

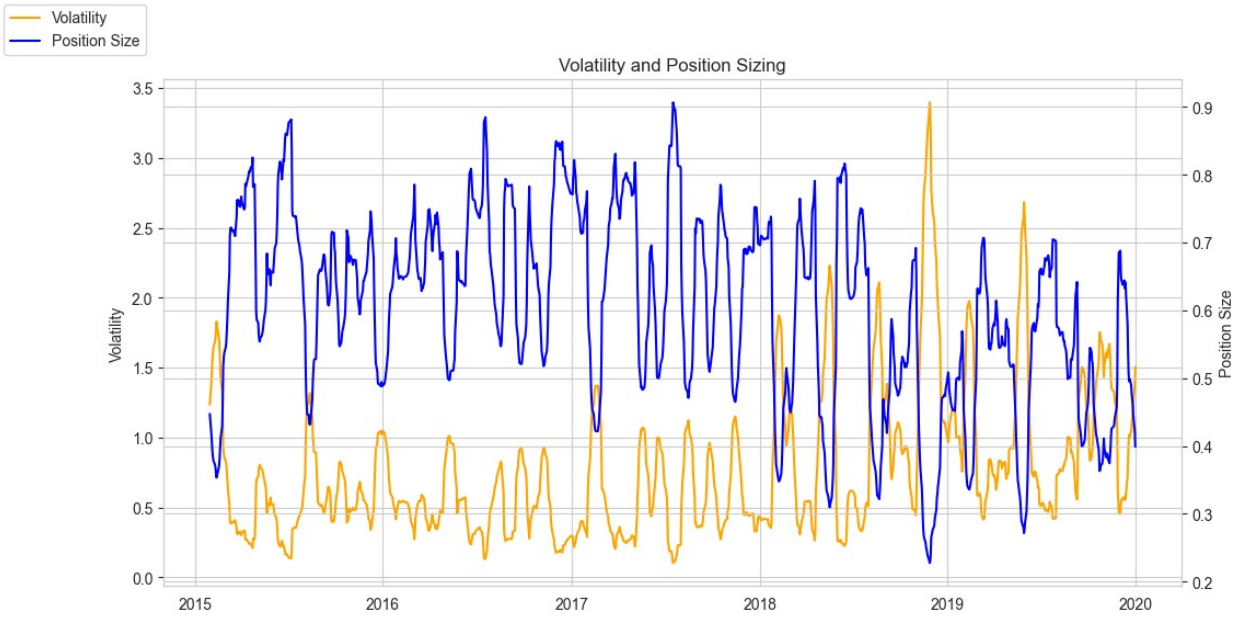

```
kf = KalmanFilter(initial_state_mean=0, n_dim_obs=2)
state_means, _ = kf.em(analysis_df.values,
n_iter=10).filter(analysis_df.values)
dynamic_hedge_ratio = state_means[:, 0]
spread_dynamic = analysis_df[stock1] - dynamic_hedge_ratio *
analysis_df[stock2]

# Plot dynamic hedge ratio
plt.figure(figsize=(12, 6))
plt.plot(dynamic_hedge_ratio, label="Dynamic Hedge Ratio")
plt.title("Dynamic Hedge Ratio (Kalman Filter)")
plt.legend()
plt.show()
```




```
plt.figure(figsize=(12, 6))
plt.plot(spread_dynamic, label="Dynamic Spread")
plt.axhline(spread_dynamic.mean(), color='red', linestyle='--',
label="Mean")
plt.legend()
plt.title("Spread with Dynamic Hedge Ratio")
plt.show()
```

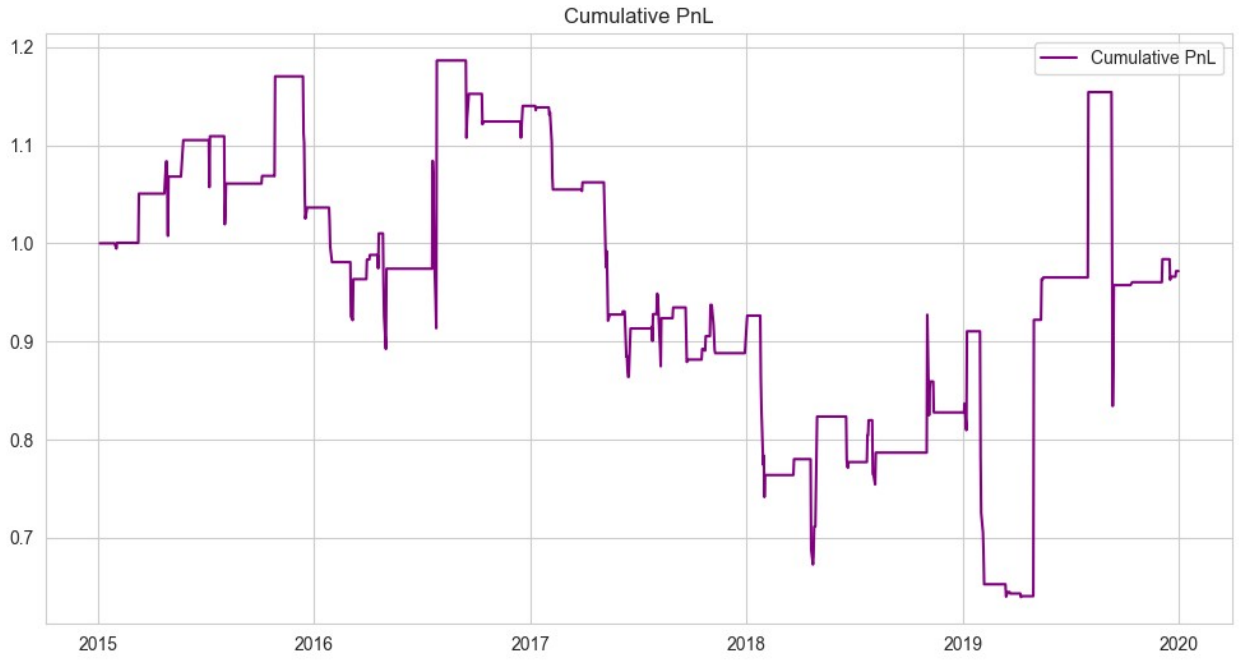


```
volatility = spread.rolling(window=20).std()
threshold = spread.std() * 2
position_size = 1 / (1 + volatility)

fig, ax1 = plt.subplots(figsize=(12, 6))
ax1.plot(volatility, label="Volatility", color="orange")
ax1.set_ylabel("Volatility")
ax2 = ax1.twinx()
ax2.plot(position_size, label="Position Size", color="blue")
ax2.set_ylabel("Position Size")
fig.legend(loc="upper left")
plt.title("Volatility and Position Sizing")
plt.show()
```




```
min_holding_period = 5  # days
signals = pd.DataFrame(index=spread.index)
signals['long'] = (spread_zscore < -2) & (spread.shift(1) != 1)
signals['short'] = (spread_zscore > 2) & (spread.shift(1) != -1)
signals['exit'] = ((spread_zscore.abs() < 0.5) &
                   (spread.shift(1) != 0) &
                   (spread.shift(min_holding_period).notna()))

# Calculate positions and returns
positions = np.where(signals['long'], 1, np.where(signals['short'], -
1, 0))
positions = pd.Series(positions, index=spread.index).fillna(0)
daily_returns = positions.shift(1) * spread.pct_change()
cumulative_pnl = (1 + daily_returns).cumprod()
```

```
plt.figure(figsize=(12, 6))
plt.plot(spread, label="Spread")
plt.axhline(threshold, color="red", linestyle="--", label="Stop-Loss
Threshold")
plt.axhline(-threshold, color="green", linestyle="--")
plt.scatter(signals.loc[signals['long']].index,
spread[signals['long']],
           color='green', label='Buy Signal', marker='^')
plt.scatter(signals.loc[signals['short']].index,
spread[signals['short']],
           color='red', label='Sell Signal', marker='v')
plt.legend()
plt.title("Trading Signals")
plt.show()

# Plot PnL
plt.figure(figsize=(12, 6))
plt.plot(cumulative_pnl, label="Cumulative PnL", color="purple")
plt.title("Cumulative PnL")
plt.legend()
plt.show()
```



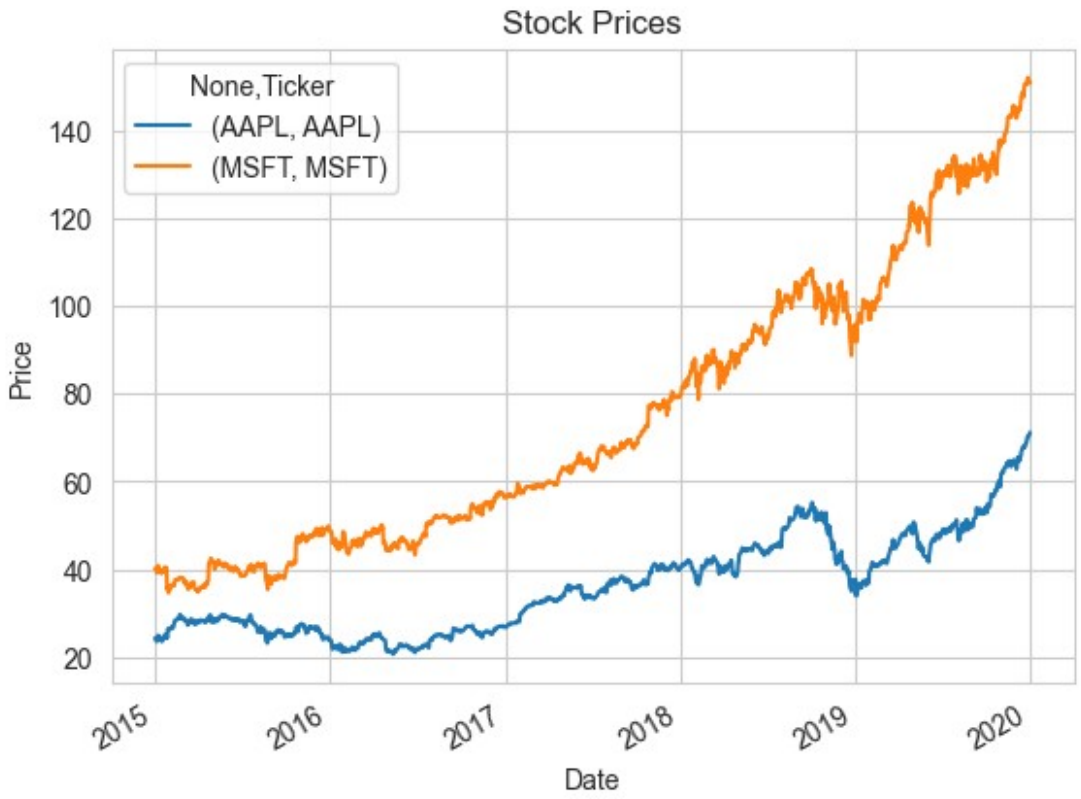

```
returns = daily_returns[~daily_returns.isna()]
total_trades = (positions.diff() != 0).sum()
win_trades = (returns > 0).sum()
loss_trades = (returns < 0).sum()
win_rate = win_trades / total_trades if total_trades > 0 else 0

# Calculate drawdown
cumulative = (1 + returns).cumprod()
rolling_max = cumulative.expanding().max()
drawdown = (cumulative - rolling_max) / rolling_max

# Calculate Sharpe ratio
sharpe_ratio = daily_returns.mean() / daily_returns.std() *
np.sqrt(252)

# Print performance metrics
print(f"\nPerformance Metrics:")
print(f"Total Trades: {total_trades}")
print(f"Win Rate: {win_rate:.2%}")
print(f"Max Drawdown: {drawdown.min():.2%}")
print(f"Sharpe Ratio: {sharpe_ratio:.4f}")
print(f"Annual Return: {(cumulative_pnl.iloc[-
1]**(252/len(cumulative_pnl))-1):.2%}")


Performance Metrics:
Total Trades: 123
Win Rate: 55.28%
Max Drawdown: -46.09%
```

```
Sharpe Ratio: 0.1494
Annual Return: -0.57%
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf
from pykalman import KalmanFilter
import seaborn as sns

sns.set_style("whitegrid")

# Step 1: Fetching the Data
stock1 = "AAPL"
stock2 = "MSFT"
start_date = "2015-01-01"
end_date = "2020-01-01"

# Fetch data
data1 = yf.download(stock1, start=start_date, end=end_date)["Adj
Close"]
data2 = yf.download(stock2, start=start_date, end=end_date)["Adj
Close"]

# Check if data was downloaded correctly
if data1 is None or data2 is None:
    raise ValueError("Stock data could not be fetched. Check the stock
symbols or the date range.")

# Merge data into a single DataFrame
prices = pd.concat([data1, data2], axis=1, keys=[stock1,
stock2]).dropna()

# Plot the raw prices
plt.figure(figsize=(12, 6))
prices.plot(title="Stock Prices")
plt.ylabel("Price")
plt.show()
```

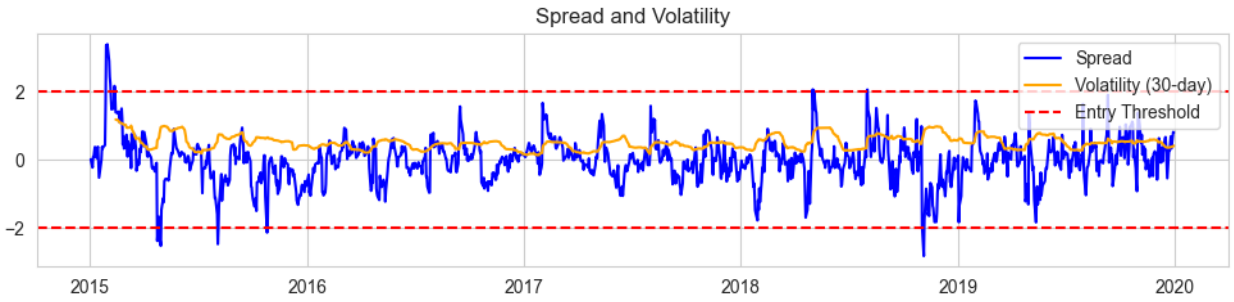
```
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
```

```
<Figure size 1200x600 with 0 Axes>
```




```
stock1_prices = pd.to_numeric(prices.loc[:, (stock1,
stock1)].squeeze(), errors='coerce')
stock2_prices = pd.to_numeric(prices.loc[:, (stock2,
stock2)].squeeze(), errors='coerce')

# Create a new DataFrame with just the prices we need
data = pd.DataFrame({
    stock1: stock1_prices,
    stock2: stock2_prices
})

# Now drop NaN values
data = data.dropna()

def kalman_filter(y, x):
    delta = 1e-5
    trans_cov = delta / (1 - delta) * np.eye(2)  # State transition
covariance
    obs_mat = np.expand_dims(np.vstack([x, np.ones(len(x))]).T,
axis=1)

    kf = KalmanFilter(n_dim_obs=1, n_dim_state=2,
                      transition_matrices=np.eye(2),
                      observation_matrices=obs_mat,
                      observation_covariance=1.0,
```

```
                      transition_covariance=trans_cov,
                      initial_state_mean=[0, 0],
                      initial_state_covariance=np.eye(2))

    state_means, _ = kf.filter(y)
    return state_means

hedge_ratios = kalman_filter(data["AAPL"].values, data["MSFT"].values)
data["Hedge_Ratio"] = hedge_ratios[:, 0]
data["Intercept"] = hedge_ratios[:, 1]
data["Spread"] = data["AAPL"] - (data["Hedge_Ratio"] * data["MSFT"] +
data["Intercept"])

rolling_window = 30
data["Volatility"] =
data["Spread"].rolling(window=rolling_window).std()

# Step 3: Define Entry and Exit Thresholds
entry_threshold = 2
exit_threshold = 0.5
data["Signal"] = 0
data.loc[data["Spread"] > entry_threshold * data["Volatility"],
"Signal"] = -1  # Short spread
data.loc[data["Spread"] < -entry_threshold * data["Volatility"],
"Signal"] = 1   # Long spread
data.loc[abs(data["Spread"]) < exit_threshold * data["Volatility"],
"Signal"] = 0  # Exit

data["Position"] = data["Signal"].ffill().fillna(0)

# Step 4: Backtesting
data["Returns"] = data["Position"].shift() * data["Spread"].diff()
data["Cumulative_Returns"] = data["Returns"].cumsum()

plt.figure(figsize=(12, 8))

# Plot spread and volatility
plt.subplot(3, 1, 1)
plt.plot(data.index, data["Spread"], label="Spread", color="blue")
plt.plot(data.index, data["Volatility"], label="Volatility (30-day)",
color="orange")
plt.axhline(entry_threshold, color="red", linestyle="--", label="Entry
Threshold")
plt.axhline(-entry_threshold, color="red", linestyle="--")
plt.title("Spread and Volatility")
plt.legend()

<matplotlib.legend.Legend at 0x1d01469fb30>
```

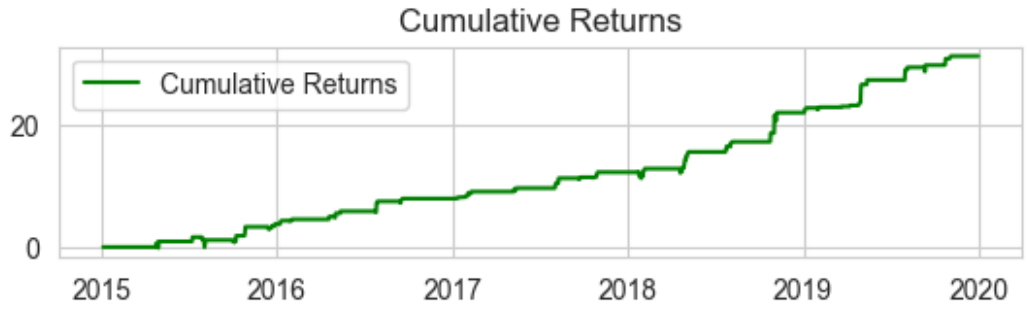



```
plt.subplot(3, 1, 2)
plt.plot(data.index, data["Signal"], label="Signal", color="purple")
plt.title("Trading Signals")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1d0143b4290>
```


```
plt.subplot(3, 1, 3)
plt.plot(data.index, data["Cumulative_Returns"], label="Cumulative
Returns", color="green")
plt.title("Cumulative Returns")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1d0146491c0>
```



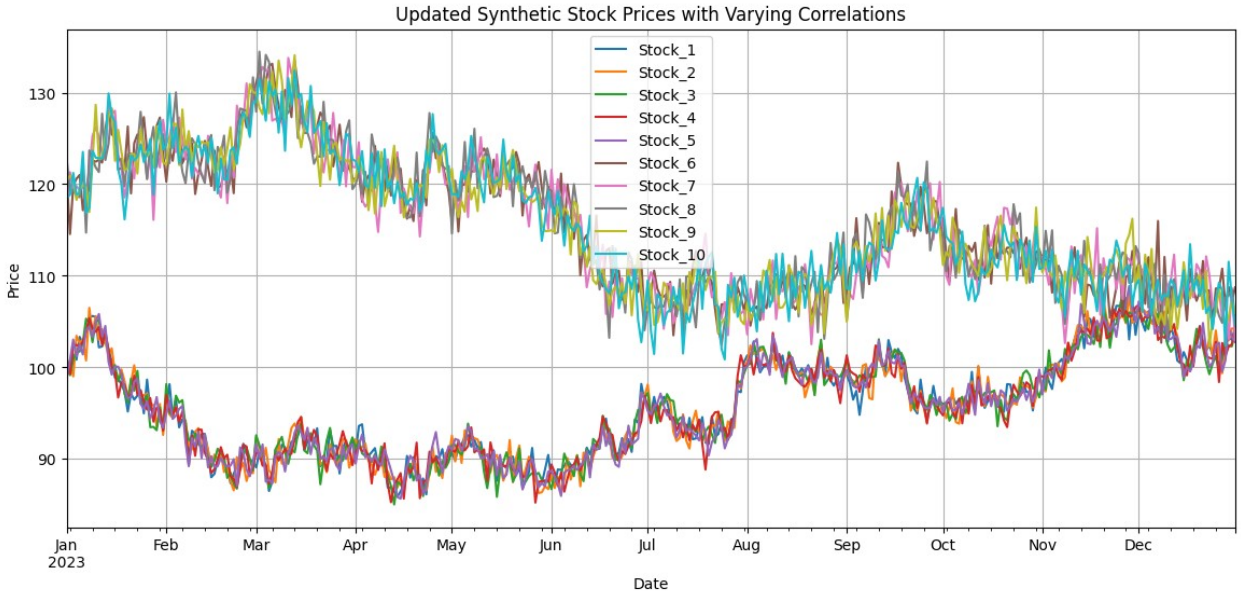

```
plt.tight_layout()
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```

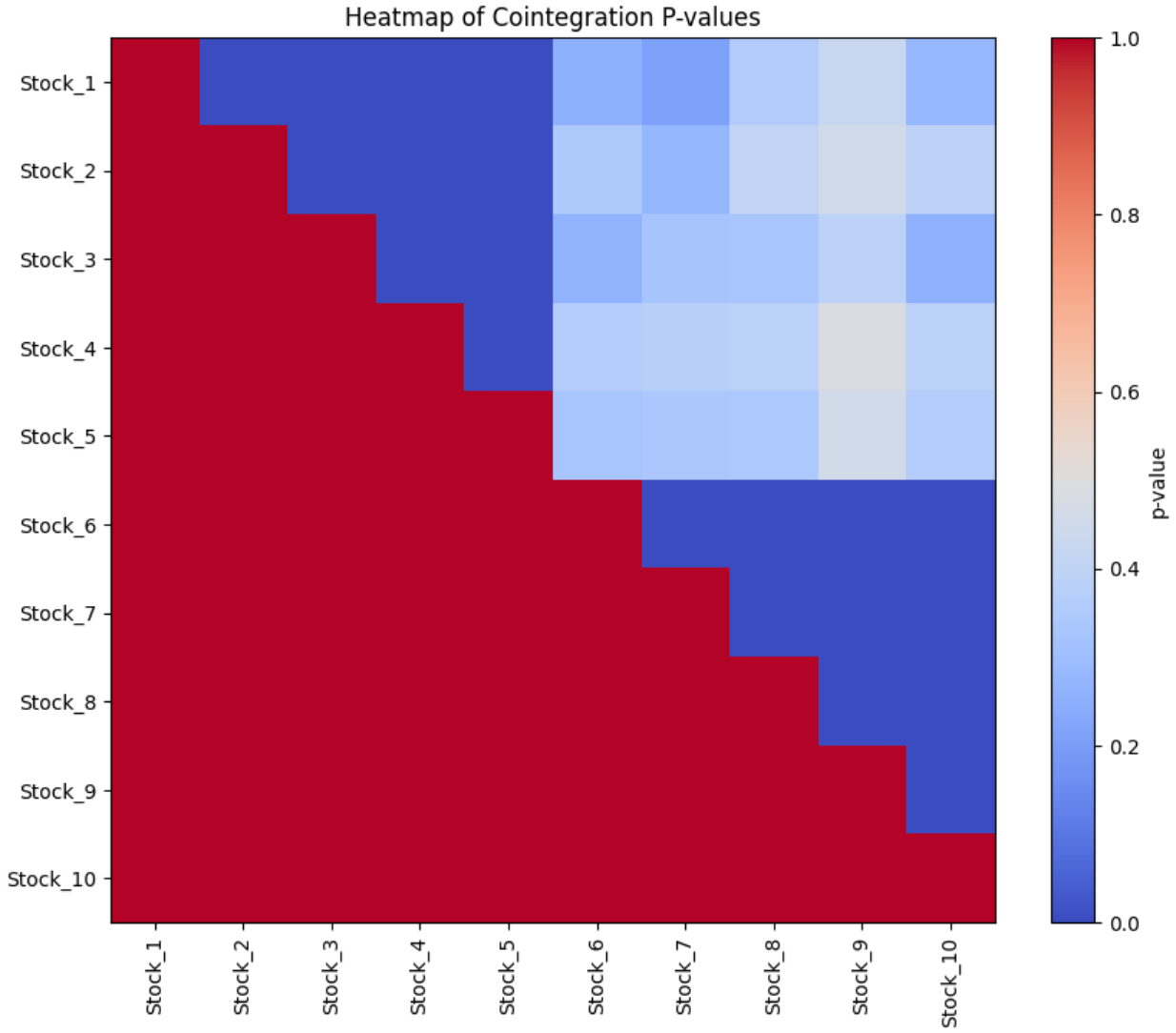
```
print("Final Cumulative Returns:", data["Cumulative_Returns"].iloc[-1])
```

```
Final Cumulative Returns: 31.356945023370518
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import coint
from pykalman import KalmanFilter

np.random.seed(42)
dates = pd.date_range('2023-01-01', '2023-12-31', freq='D')
num_stocks = 10

# Generate synthetic stock prices with grouped correlations
group_size = num_stocks // 2
group_1 = np.random.randn(len(dates)).cumsum() + 100
group_2 = np.random.randn(len(dates)).cumsum() + 120

prices = pd.DataFrame(
    {f'Stock_{i}': group_1 + np.random.normal(0, 1, len(dates)) for i
in range(1, group_size + 1)},
    index=dates
)

prices = pd.concat(
    [prices,
     pd.DataFrame(
         {f'Stock_{i}': group_2 + np.random.normal(0, 2, len(dates))
for i in range(group_size + 1, num_stocks + 1)},
         index=dates
     )],
    axis=1
)

# Plot updated synthetic stock prices
plt.figure(figsize=(14, 6))
prices.plot(ax=plt.gca())
plt.title('Updated Synthetic Stock Prices with Varying Correlations')
plt.xlabel('Date')
plt.ylabel('Price')
plt.grid()
plt.show()
```




```
def find_cointegrated_pairs(data):
    n = data.shape[1]
    pvalue_matrix = np.ones((n, n))
    pairs = []
    for i in range(n):
        for j in range(i + 1, n):
            _, pvalue, _ = coint(data.iloc[:, i], data.iloc[:, j])
            pvalue_matrix[i, j] = pvalue
            if pvalue < 0.05:
                pairs.append((data.columns[i], data.columns[j]))
    return pairs, pvalue_matrix

pairs, pvalues = find_cointegrated_pairs(prices)

plt.figure(figsize=(10, 8))
plt.imshow(pvalues, cmap='coolwarm', interpolation='none')
plt.colorbar(label='p-value')
plt.xticks(range(num_stocks), prices.columns, rotation=90)
plt.yticks(range(num_stocks), prices.columns)
plt.title('Heatmap of Cointegration P-values')
plt.show()
```

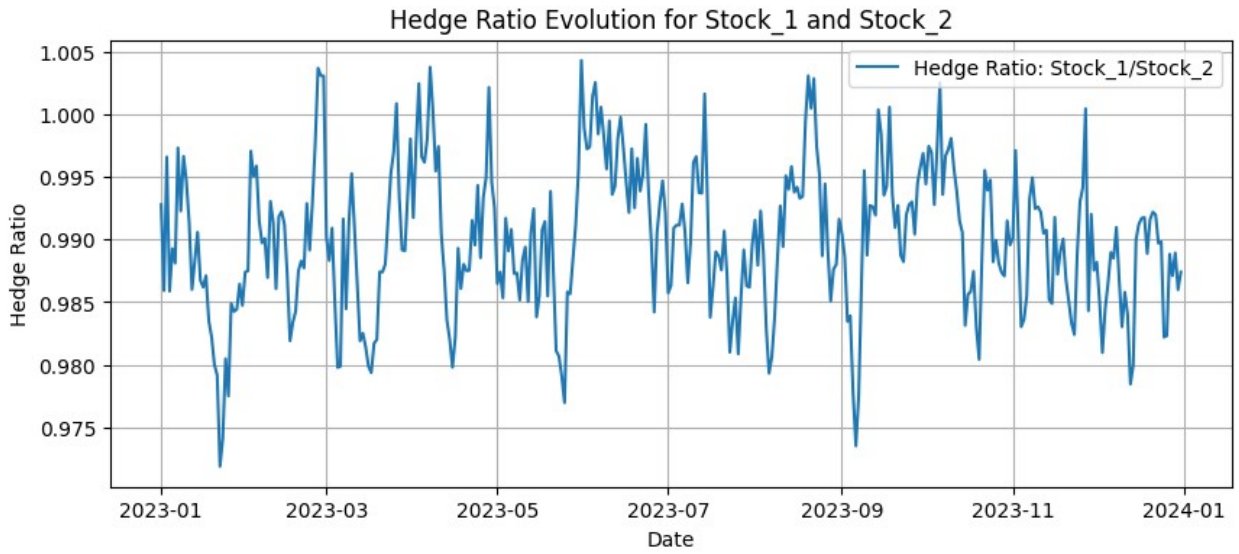


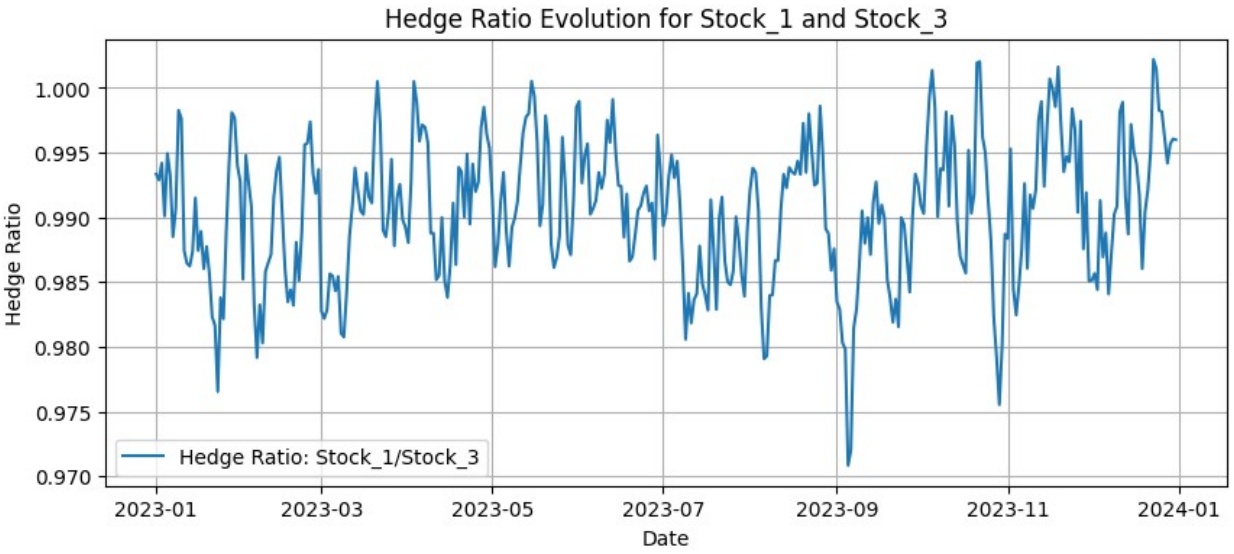

```
print("Identified Cointegrated Pairs:", pairs)
```

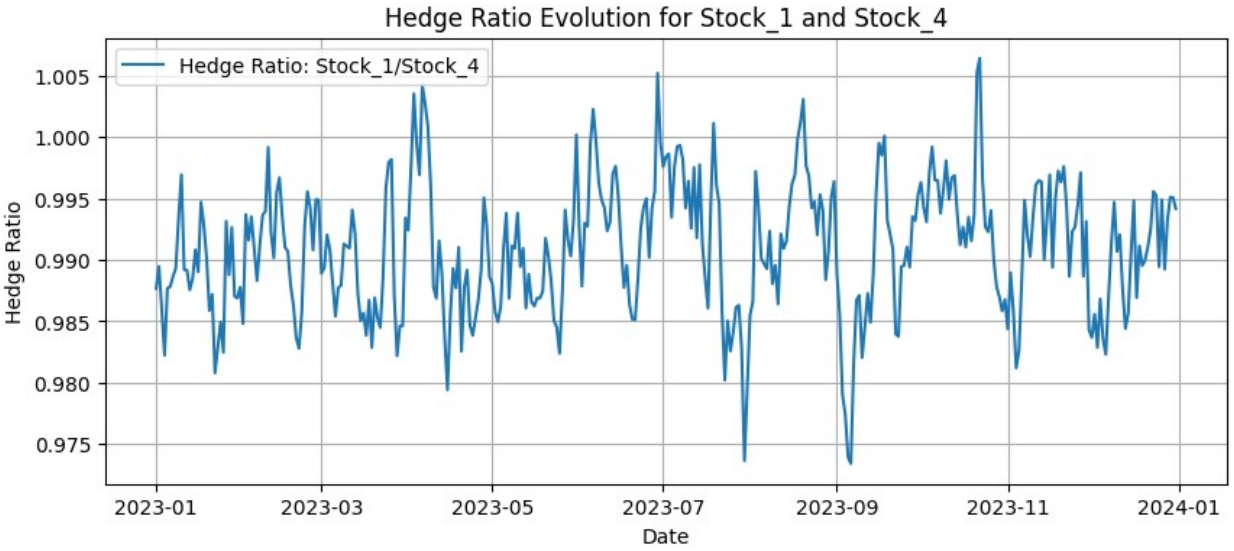
```
Identified Cointegrated Pairs: [('Stock_1', 'Stock_2'), ('Stock_1',
'Stock_3'), ('Stock_1', 'Stock_4'), ('Stock_1', 'Stock_5'),
('Stock_2', 'Stock_3'), ('Stock_2', 'Stock_4'), ('Stock_2',
'Stock_5'), ('Stock_3', 'Stock_4'), ('Stock_3', 'Stock_5'),
('Stock_4', 'Stock_5'), ('Stock_6', 'Stock_7'), ('Stock_6',
'Stock_8'), ('Stock_6', 'Stock_9'), ('Stock_6', 'Stock_10'),
('Stock_7', 'Stock_8'), ('Stock_7', 'Stock_9'), ('Stock_7',
'Stock_10'), ('Stock_8', 'Stock_9'), ('Stock_8', 'Stock_10'),
('Stock_9', 'Stock_10')]
```

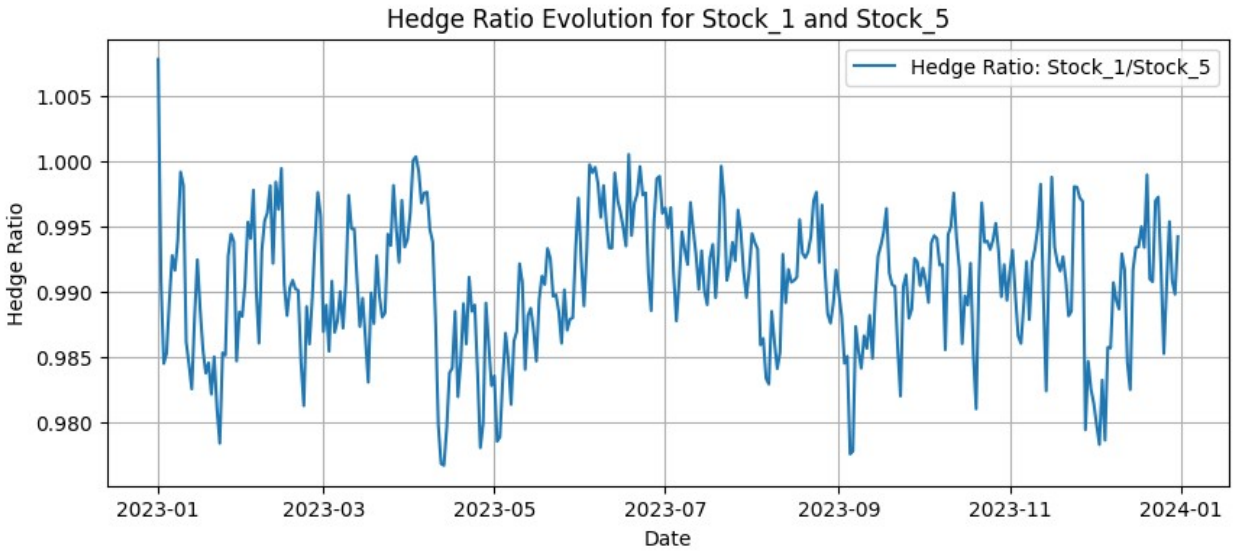
```
def kalman_filter_hedge_ratio(y, x):
    delta = 1e-5  # Process noise
    trans_cov = delta / (1 - delta) * np.eye(2)  # State transition
covariance
    obs_mat = np.expand_dims(np.vstack([x, np.ones_like(x)]).T,
```

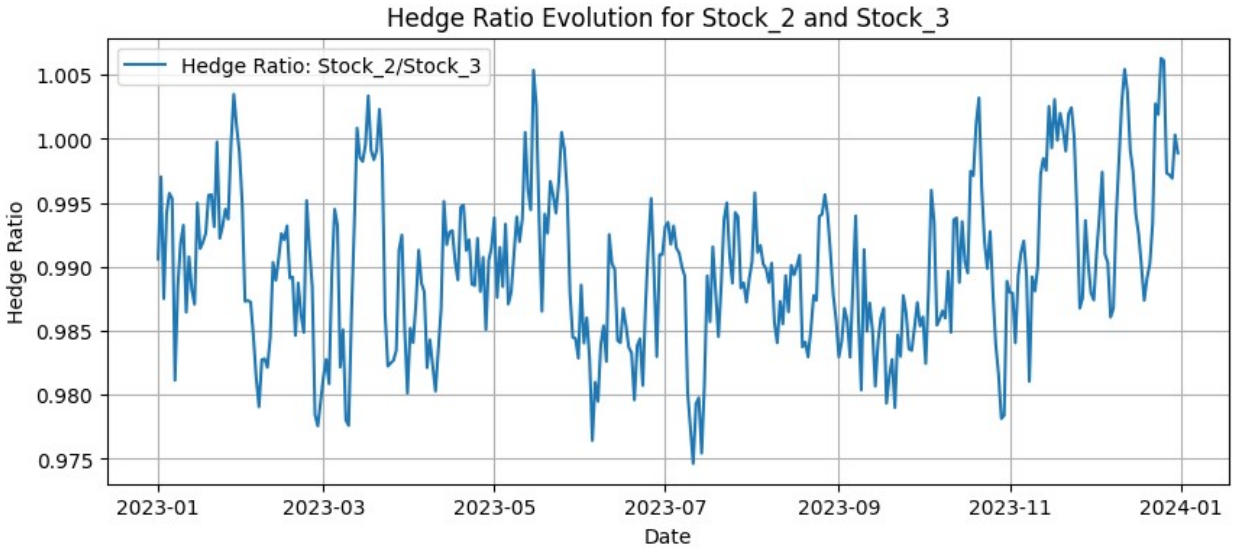
```
axis=1)

    kf = KalmanFilter(n_dim_obs=1, n_dim_state=2,
                      initial_state_mean=[0, 0],
                      initial_state_covariance=np.ones((2, 2)),
                      transition_matrices=np.eye(2),
                      observation_matrices=obs_mat,
                      observation_covariance=1.0,
                      transition_covariance=trans_cov)

    state_means, _ = kf.filter(y)
    return state_means[:, 0]  # Hedge ratio

hedge_ratios = {}

for pair in pairs:
    stock_1, stock_2 = pair
    y = prices[stock_1]
    x = prices[stock_2]
    hedge_ratios[pair] = kalman_filter_hedge_ratio(y, x)

    # Plot hedge ratio evolution
    plt.figure(figsize=(10, 4))
    plt.plot(dates, hedge_ratios[pair], label=f'Hedge Ratio:
{stock_1}/{stock_2}')
    plt.title(f'Hedge Ratio Evolution for {stock_1} and {stock_2}')
    plt.xlabel('Date')
    plt.ylabel('Hedge Ratio')
    plt.legend()
    plt.grid()
    plt.show()
```

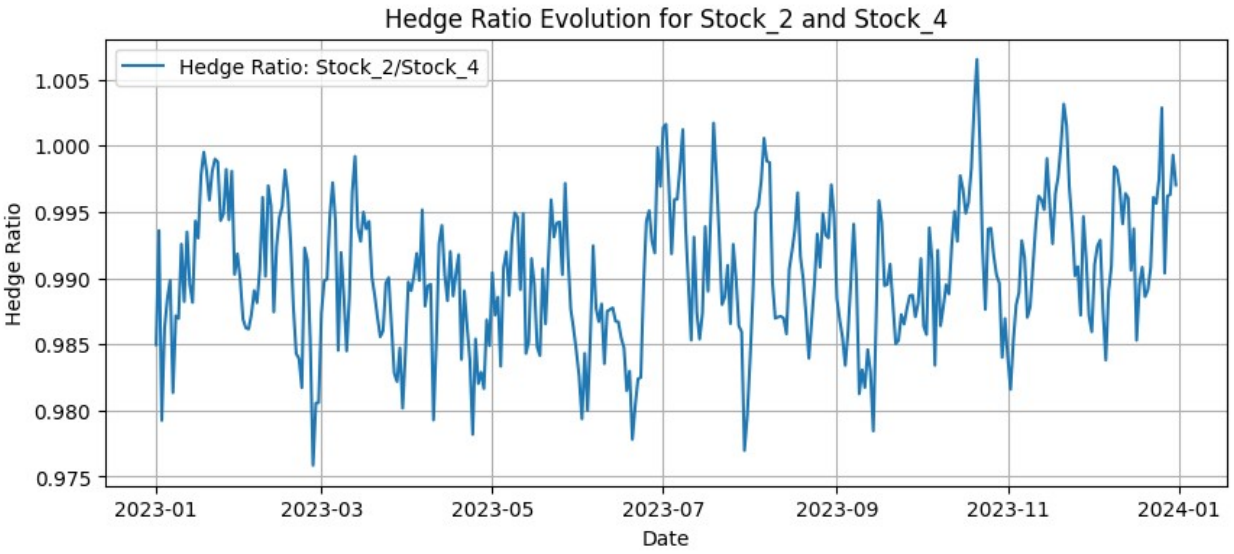


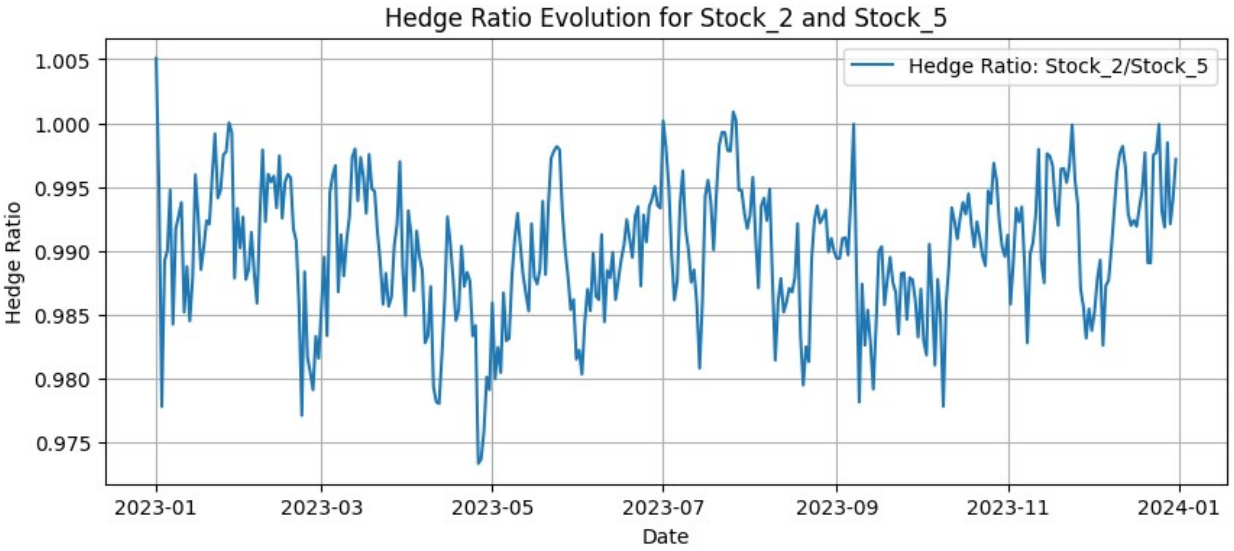

Hedge Ratio Evolution for Stock_1 and Stock_3
Hedge Ratio
1.000
0.995
0.990
0.985
0.980
0.975
0.970
Hedge Ratio: Stock_1/Stock_3
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Date

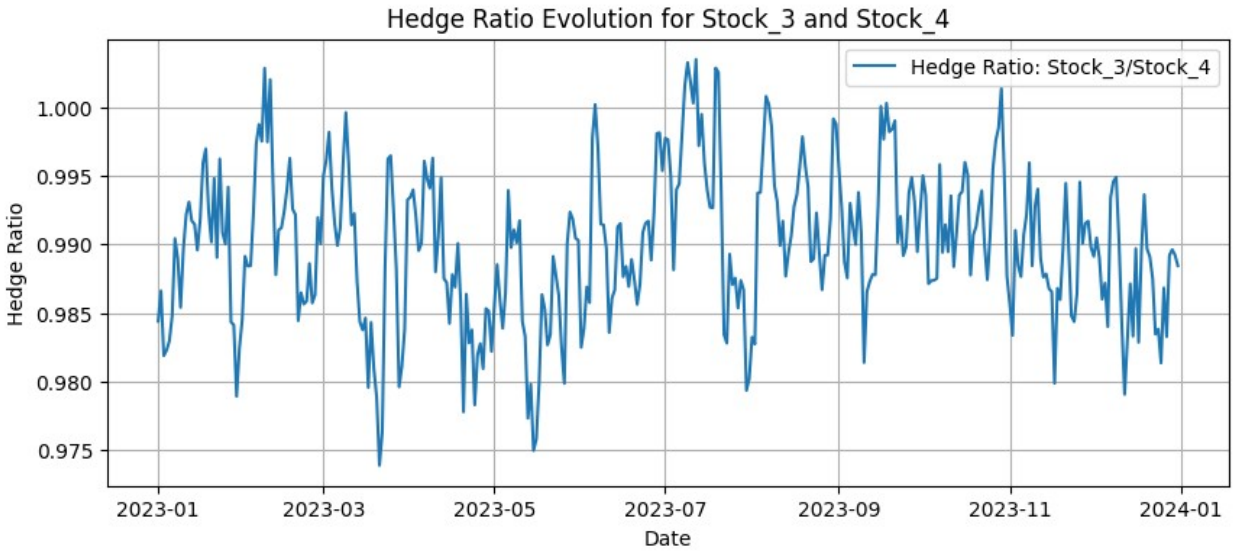

Hedge Ratio Evolution for Stock_1 and Stock_4
Hedge Ratio
1.005
1.000
0.995
0.990
0.985
0.980
0.975
Hedge Ratio: Stock_1/Stock_4
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Date

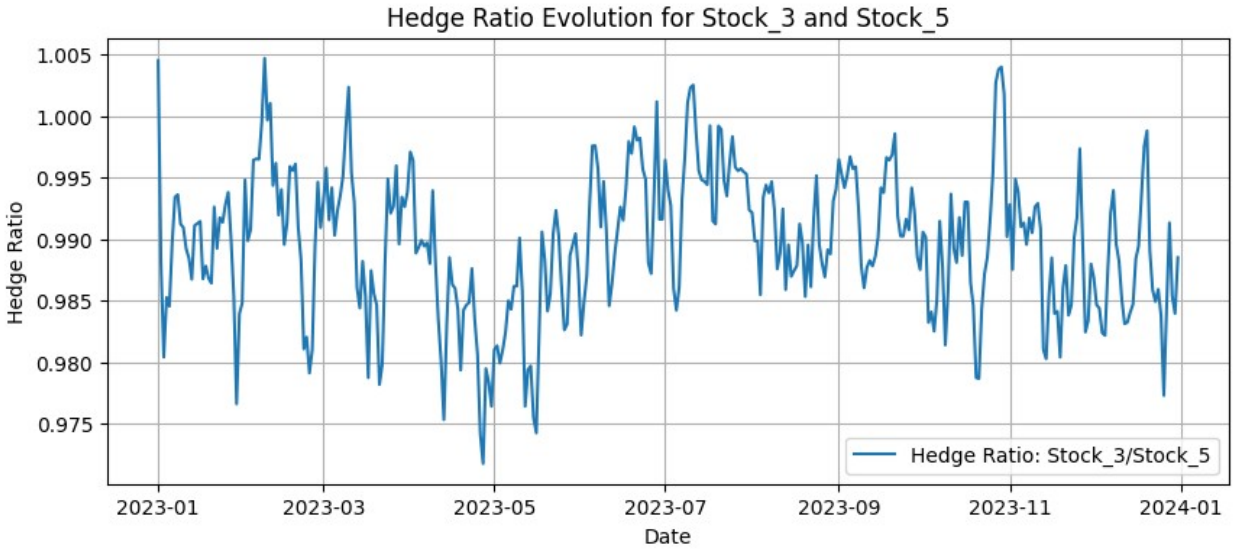

Hedge Ratio Evolution for Stock_1 and Stock_5
Hedge Ratio: Stock_1/Stock_5
Hedge Ratio
1.005
1.000
0.995
0.990
0.985
0.980
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Date

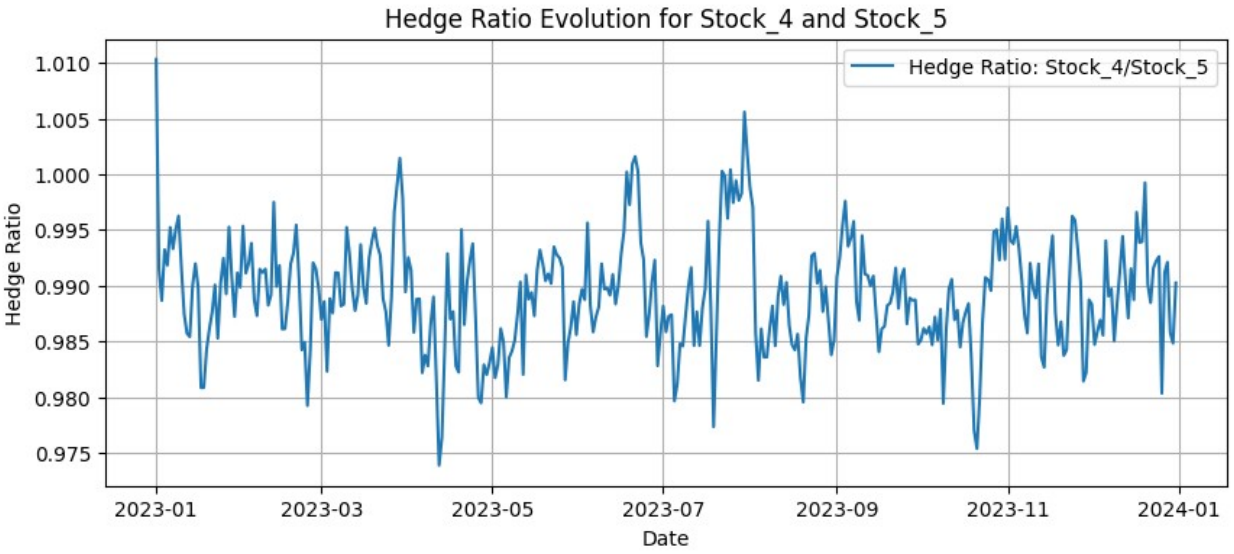

Hedge Ratio Evolution for Stock_2 and Stock_3
Hedge Ratio: Stock_2/Stock_3
Hedge Ratio
1.005
1.000
0.995
0.990
0.985
0.980
0.975
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Date

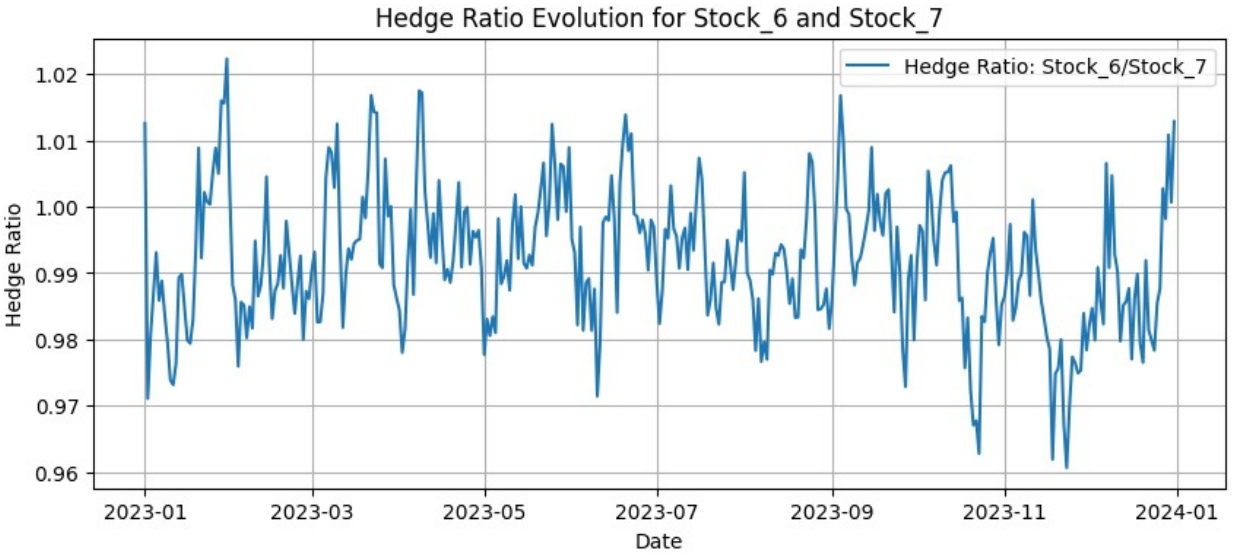

Hedge Ratio Evolution for Stock_2 and Stock_4
Hedge Ratio: Stock_2/Stock_4
Hedge Ratio
1.005
1.000
0.995
0.990
0.985
0.980
0.975
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Date

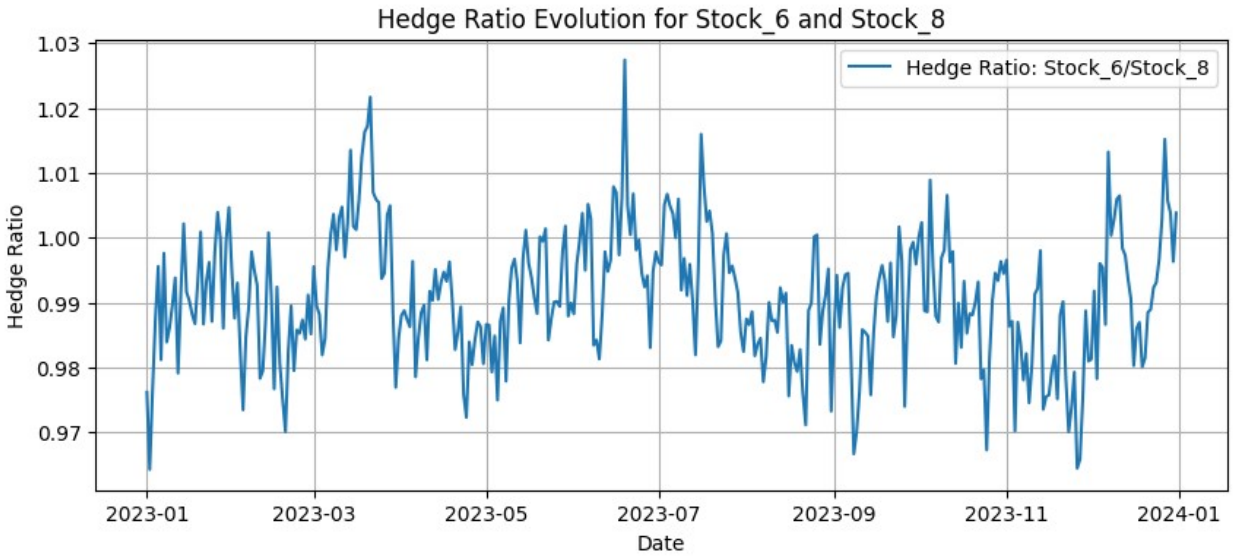

Hedge Ratio Evolution for Stock_2 and Stock_5
Hedge Ratio: Stock_2/Stock_5
Hedge Ratio
1.005
1.000
0.995
0.990
0.985
0.980
0.975
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Date

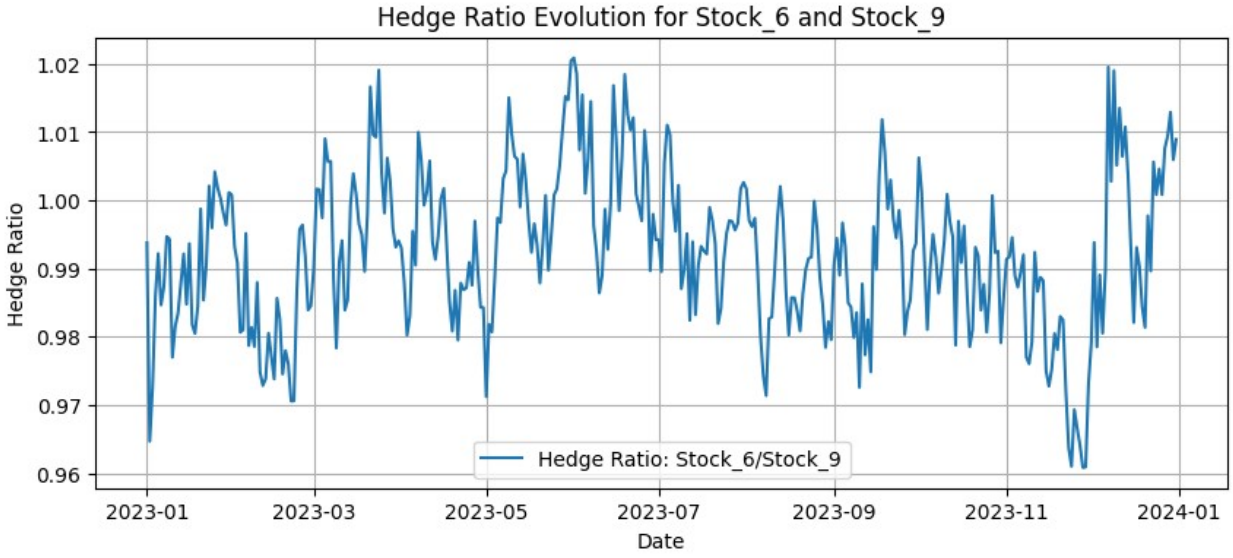

Hedge Ratio Evolution for Stock_3 and Stock_4
Hedge Ratio: Stock_3/Stock_4
Hedge Ratio
1.000
0.995
0.990
0.985
0.980
0.975
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Date

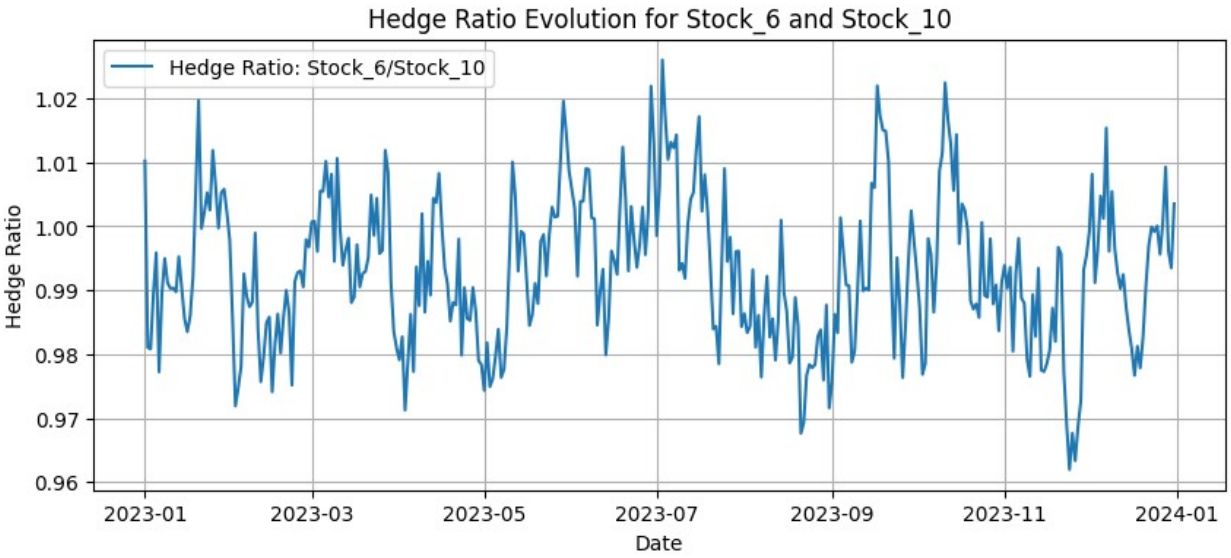

Hedge Ratio Evolution for Stock_3 and Stock_5
Hedge Ratio: Stock_3/Stock_5
Hedge Ratio
1.005
1.000
0.995
0.990
0.985
0.980
0.975
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Date

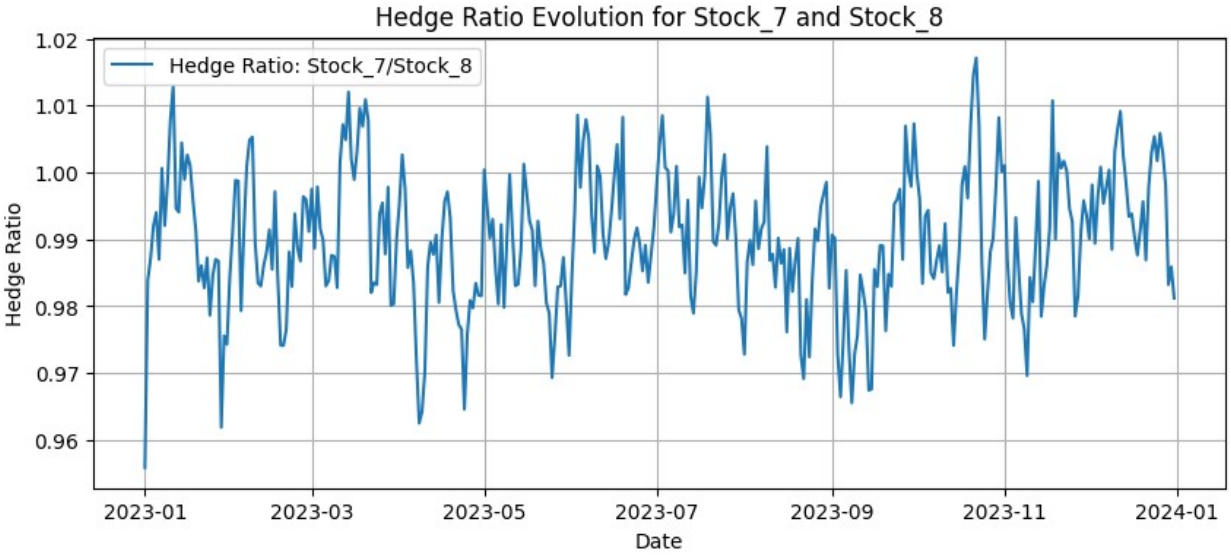

Hedge Ratio Evolution for Stock_4 and Stock_5
Hedge Ratio: Stock_4/Stock_5
Hedge Ratio
1.010
1.005
1.000
0.995
0.990
0.985
0.980
0.975
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Date

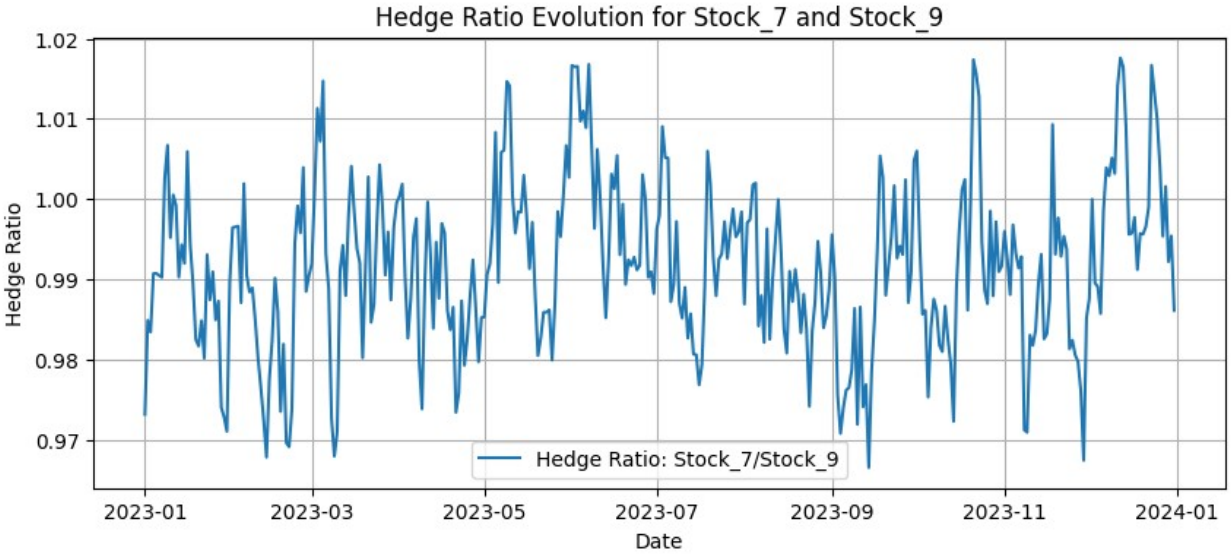

Hedge Ratio Evolution for Stock_6 and Stock_7
Hedge Ratio: Stock_6/Stock_7
Hedge Ratio
1.02
1.01
1.00
0.99
0.98
0.97
0.96
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Date

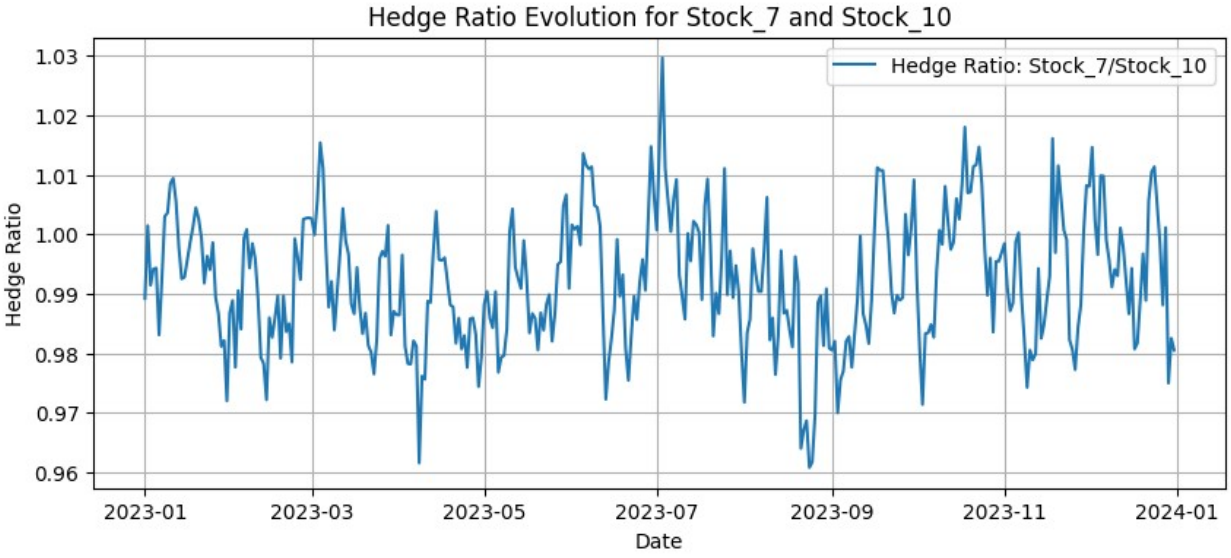

Hedge Ratio Evolution for Stock_6 and Stock_8
Hedge Ratio: Stock_6/Stock_8
Hedge Ratio
1.03
1.02
1.01
1.00
0.99
0.98
0.97
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Date

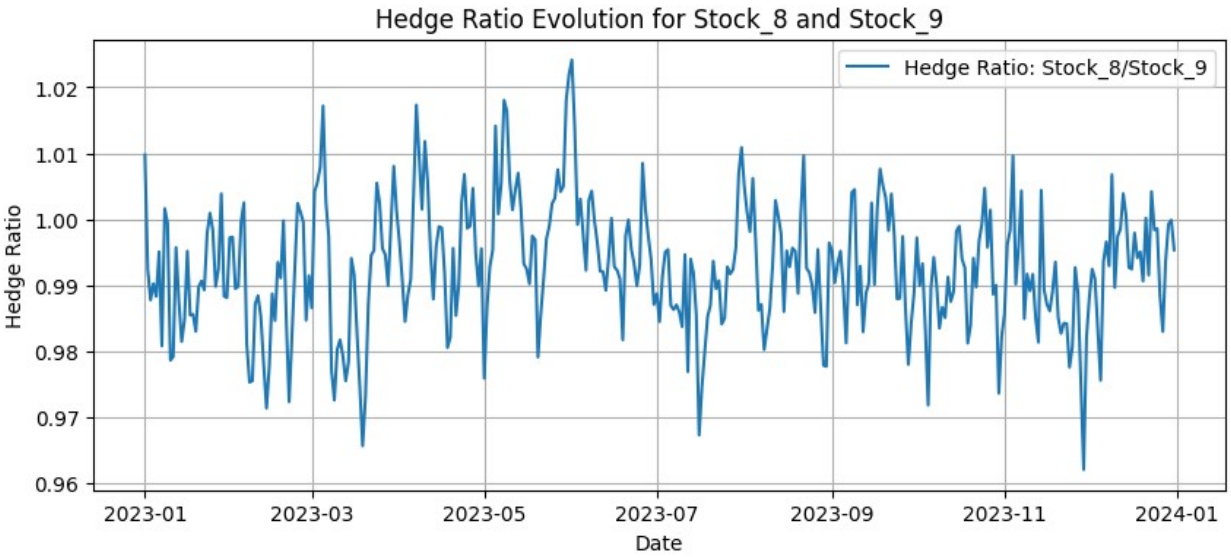

Hedge Ratio Evolution for Stock_6 and Stock_9
Hedge Ratio: Stock_6/Stock_9
Hedge Ratio
1.02
1.01
1.00
0.99
0.98
0.97
0.96
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Date

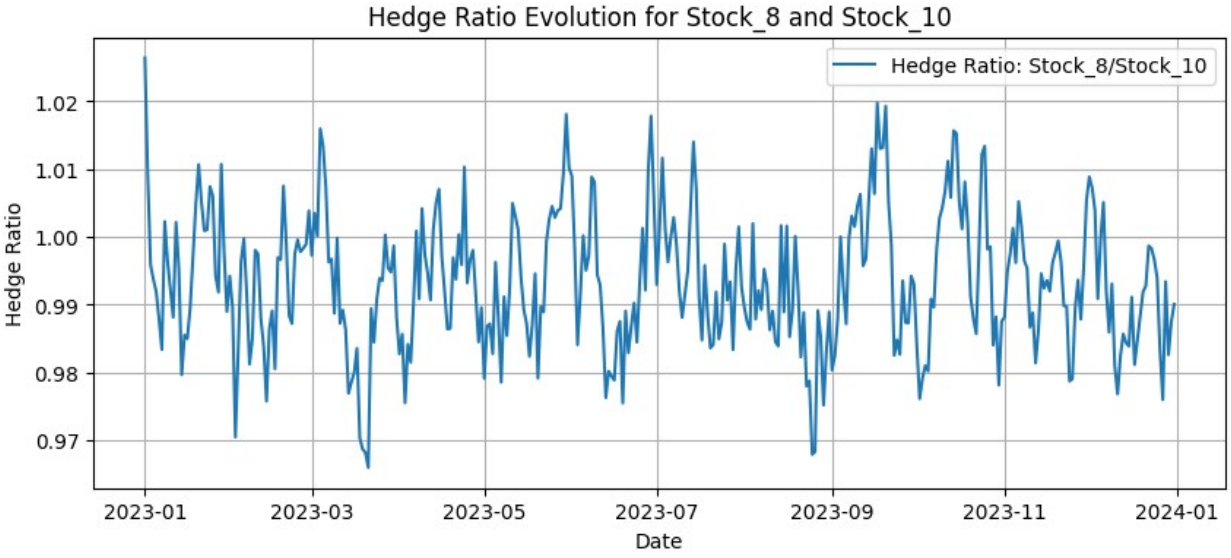

Hedge Ratio Evolution for Stock_6 and Stock_10
Hedge Ratio: Stock_6/Stock_10
Hedge Ratio
1.02
1.01
1.00
0.99
0.98
0.97
0.96
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Date

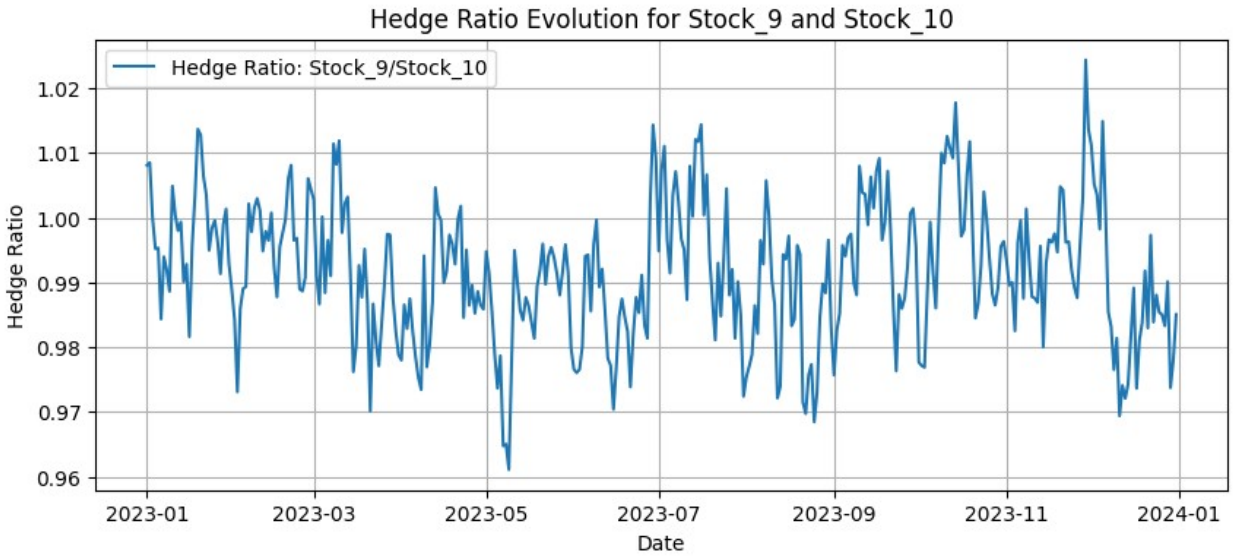

Hedge Ratio Evolution for Stock_7 and Stock_8
Hedge Ratio: Stock_7/Stock_8
Hedge Ratio
1.02
1.01
1.00
0.99
0.98
0.97
0.96
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Date


Hedge Ratio Evolution for Stock_7 and Stock_9
Hedge Ratio
Date
Hedge Ratio: Stock_7/Stock_9
1.02
1.01
1.00
0.99
0.98
0.97
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01


Hedge Ratio Evolution for Stock_7 and Stock_10
Hedge Ratio
Date
Hedge Ratio: Stock_7/Stock_10
1.03
1.02
1.01
1.00
0.99
0.98
0.97
0.96
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

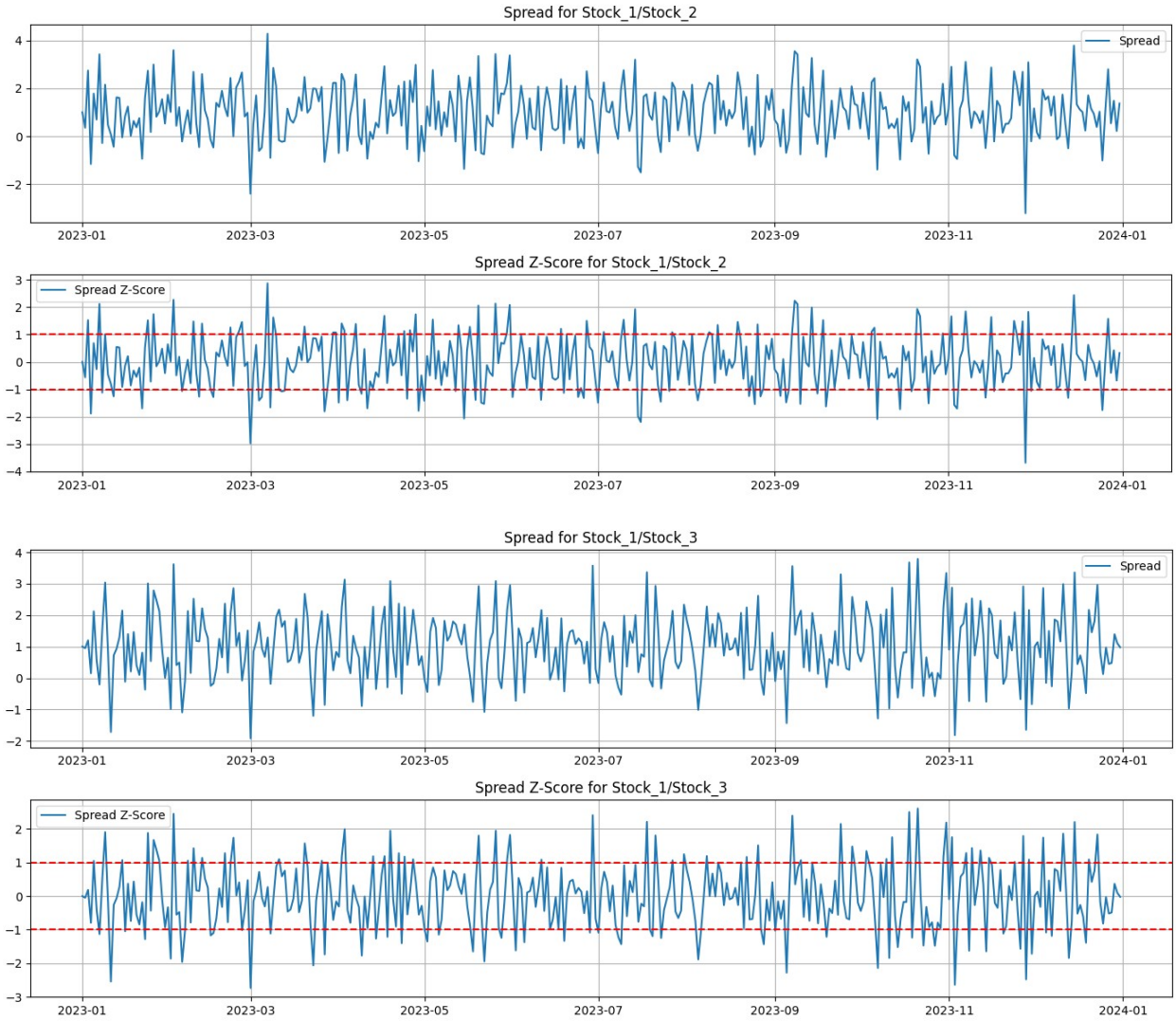

Hedge Ratio Evolution for Stock_8 and Stock_9
Hedge Ratio: Stock_8/Stock_9
Hedge Ratio
1.02
1.01
1.00
0.99
0.98
0.97
0.96
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Date

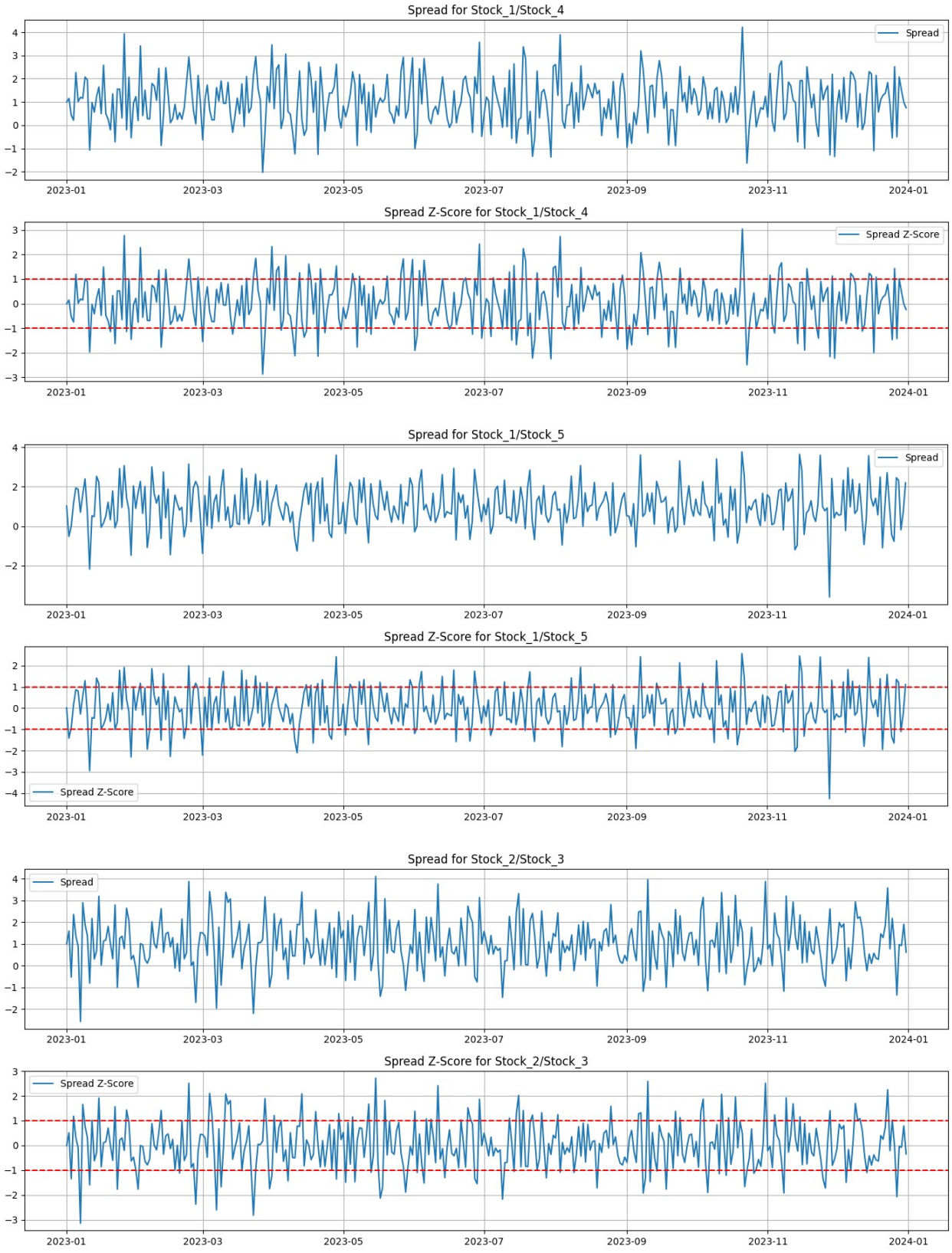

Hedge Ratio Evolution for Stock_8 and Stock_10
Hedge Ratio: Stock_8/Stock_10
Hedge Ratio
1.02
1.01
1.00
0.99
0.98
0.97
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Date

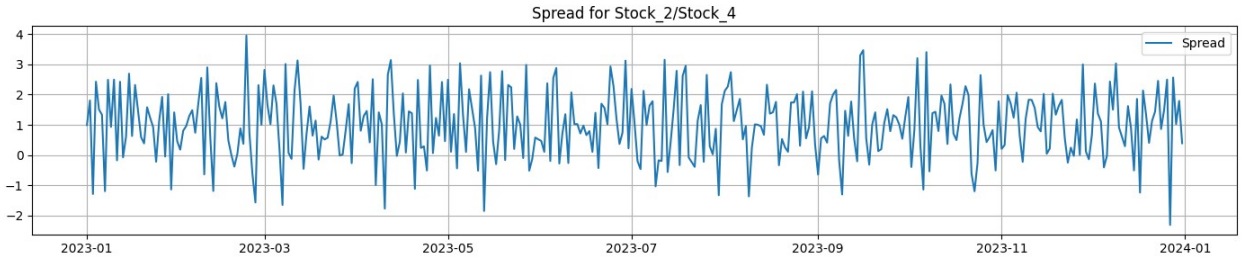


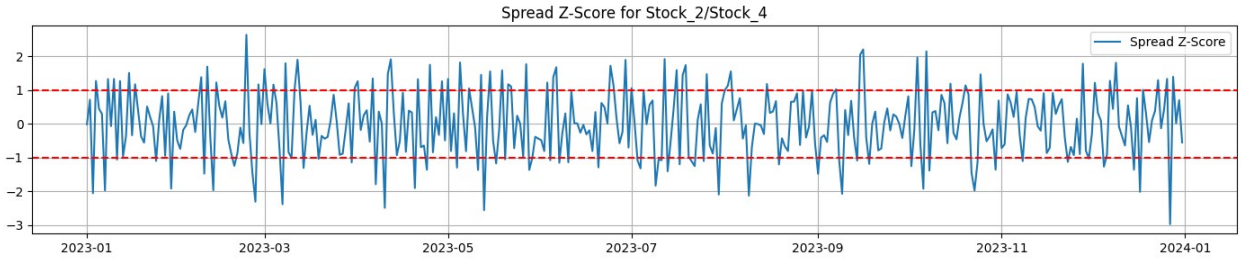

```python
def backtest_portfolio(prices, pairs, hedge_ratios):
    portfolio_pnl = pd.DataFrame(index=prices.index)

    for pair in pairs:
        stock_1, stock_2 = pair
        hedge_ratio = hedge_ratios[pair]
        spread = prices[stock_1] - hedge_ratio * prices[stock_2]
        spread_zscore = (spread - spread.mean()) / spread.std()

        # Trading signal: long/short the spread
        positions = -np.sign(spread_zscore)
        daily_pnl = positions.shift(1) * spread.diff()
        portfolio_pnl[f'{stock_1}/{stock_2}'] = daily_pnl

        # Plot spread and z-score
        plt.figure(figsize=(14, 6))
        plt.subplot(2, 1, 1)
        plt.plot(spread, label='Spread')
        plt.title(f'Spread for {stock_1}/{stock_2}')
        plt.legend()
        plt.grid()

        plt.subplot(2, 1, 2)
        plt.plot(spread_zscore, label='Spread Z-Score')
        plt.axhline(1, color='r', linestyle='--')
        plt.axhline(-1, color='r', linestyle='--')
        plt.title(f'Spread Z-Score for {stock_1}/{stock_2}')
        plt.legend()
        plt.grid()

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

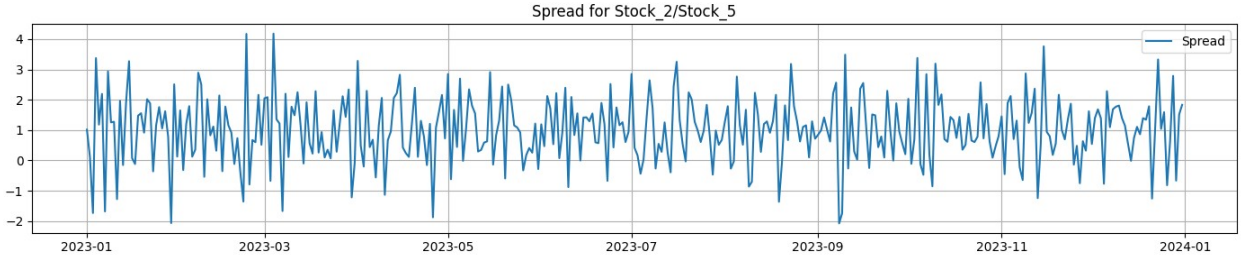
```
    # Calculate cumulative PnL
    portfolio_pnl['Total'] = portfolio_pnl.sum(axis=1)
    portfolio_pnl['Cumulative'] = portfolio_pnl['Total'].cumsum()

    # Plot cumulative PnL
    plt.figure(figsize=(12, 6))
    portfolio_pnl['Cumulative'].plot()
    plt.title('Portfolio Cumulative PnL')
    plt.xlabel('Date')
    plt.ylabel('Cumulative PnL')
    plt.grid()
    plt.show()

    return portfolio_pnl

portfolio_pnl = backtest_portfolio(prices, pairs, hedge_ratios)
```

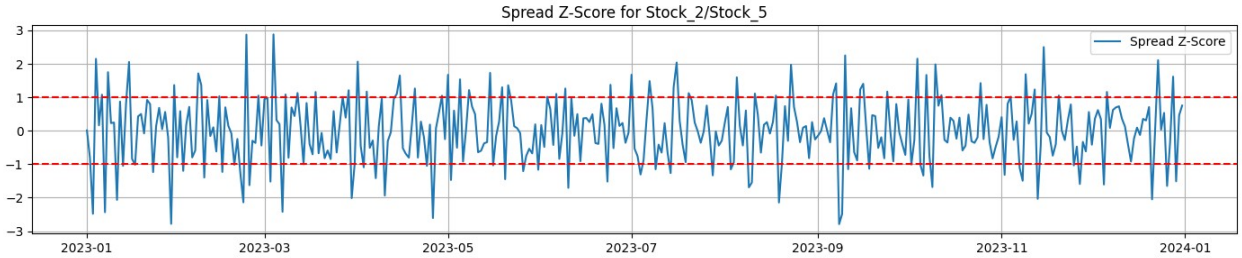


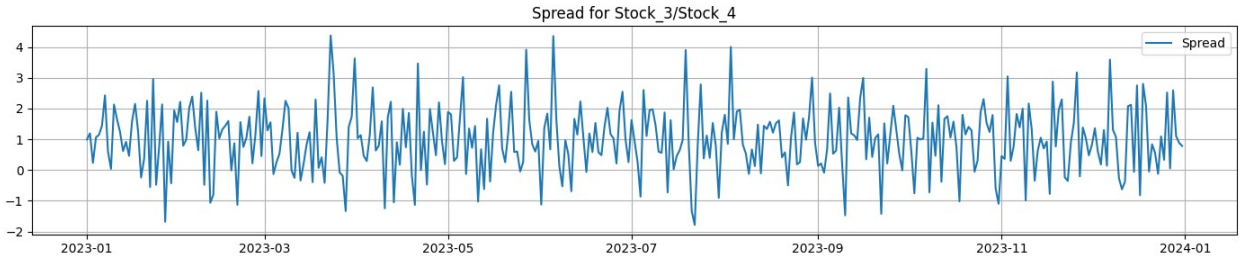

Spread for Stock_1/Stock_4
Spread
Spread Z-Score for Stock_1/Stock_4
Spread Z-Score
Spread for Stock_1/Stock_5
Spread
Spread Z-Score for Stock_1/Stock_5
Spread Z-Score
Spread for Stock_2/Stock_3
Spread
Spread Z-Score for Stock_2/Stock_3
Spread Z-Score
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

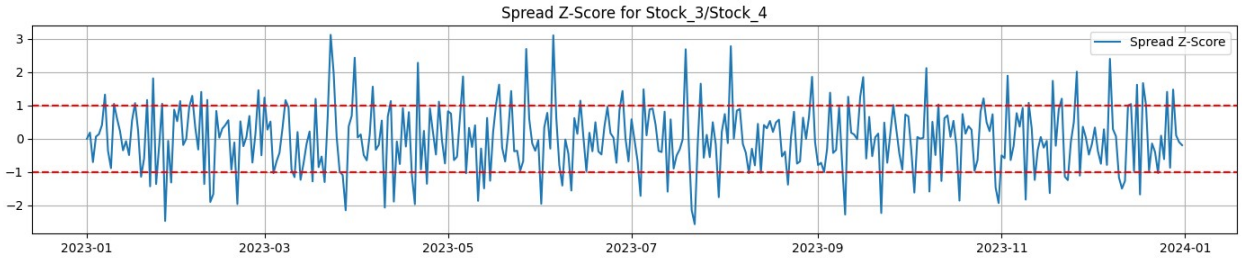

Spread for Stock_2/Stock_4
Spread
4
3
2
1
0
−1
−2
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

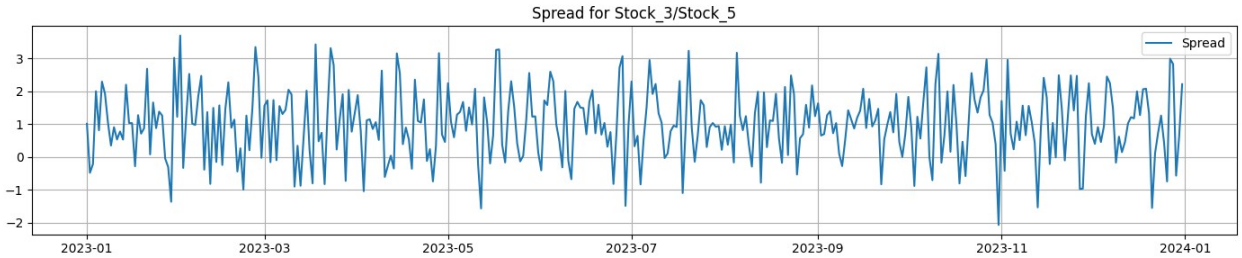

Spread Z-Score for Stock_2/Stock_4
Spread Z-Score
2
1
0
−1
−2
−3
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

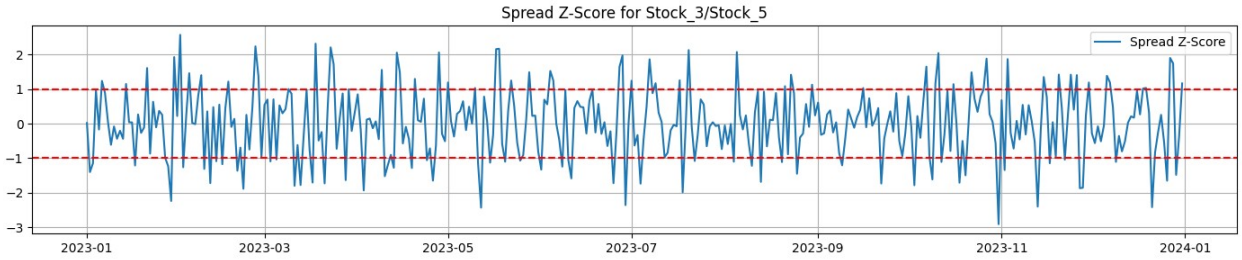

Spread for Stock_2/Stock_5
Spread
4
3
2
1
0
−1
−2
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

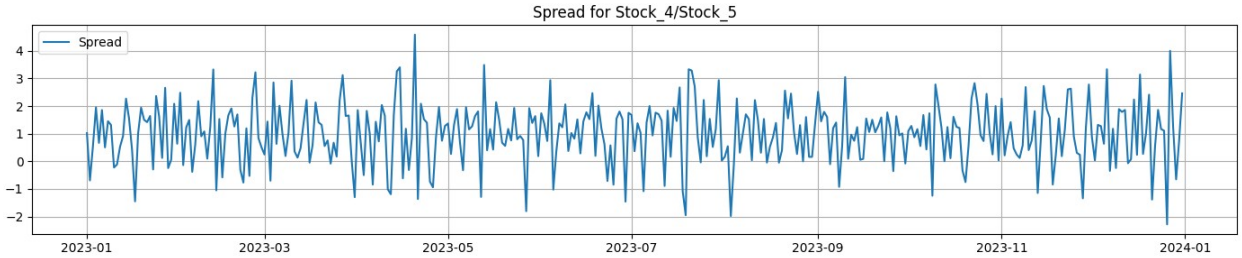

Spread Z-Score for Stock_2/Stock_5
Spread Z-Score
3
2
1
0
−1
−2
−3
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

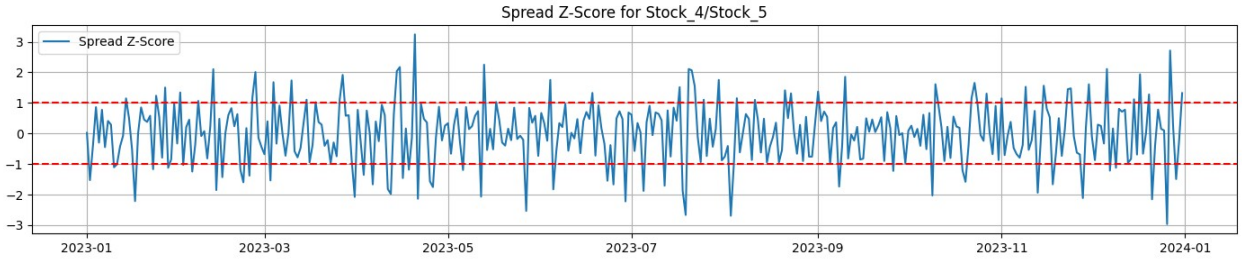

Spread for Stock_3/Stock_4
Spread
4
3
2
1
0
−1
−2
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

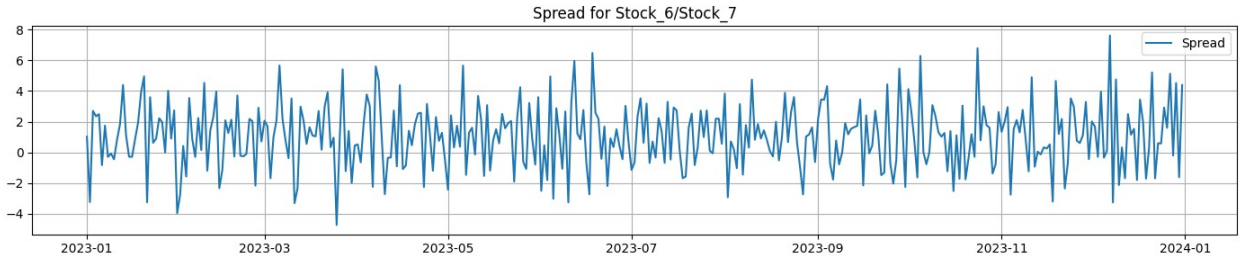

Spread Z-Score for Stock_3/Stock_4
Spread Z-Score
3
2
1
0
−1
−2
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

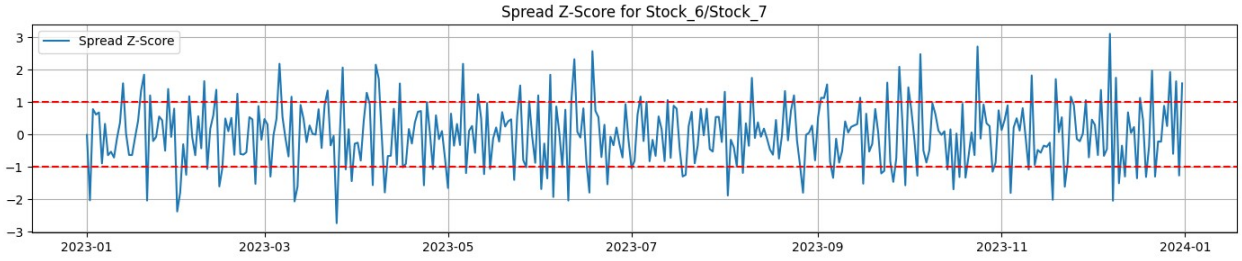

Spread for Stock_3/Stock_5
Spread
3
2
1
0
−1
−2
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

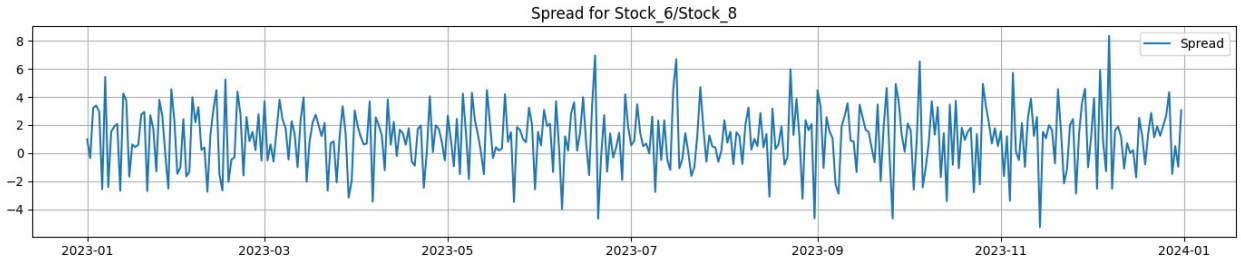

Spread Z-Score for Stock_3/Stock_5
Spread Z-Score
2
1
0
−1
−2
−3
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

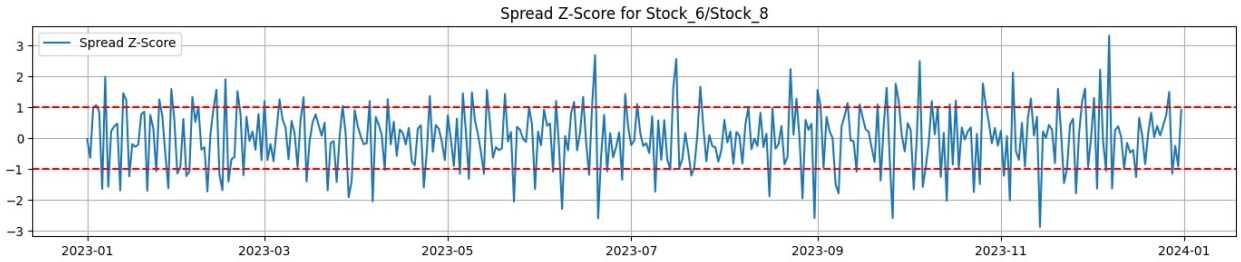

Spread for Stock_4/Stock_5
Spread
4
3
2
1
0
−1
−2
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

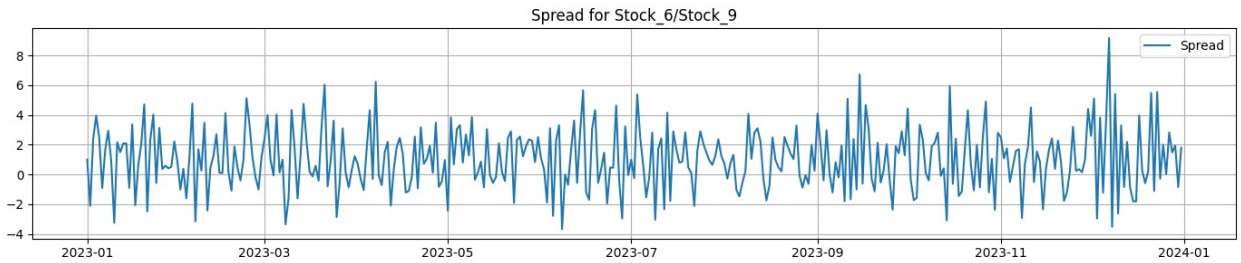

Spread Z-Score for Stock_4/Stock_5
Spread Z-Score
3
2
1
0
−1
−2
−3
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

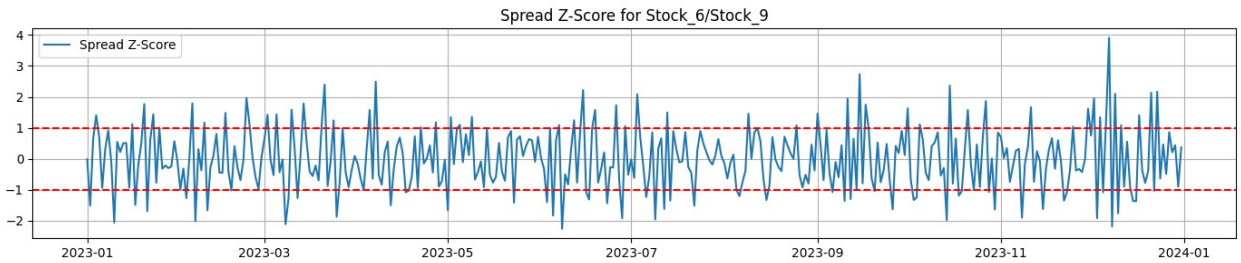

Spread for Stock_6/Stock_7
Spread
8
6
4
2
0
−2
−4
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

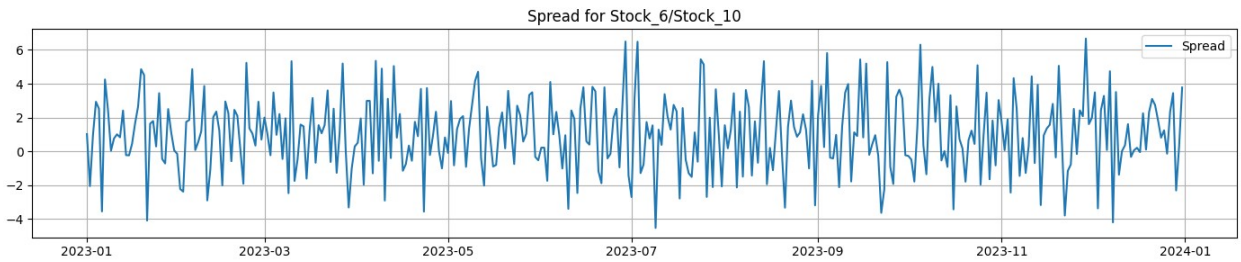

Spread Z-Score for Stock_6/Stock_7
Spread Z-Score
3
2
1
0
−1
−2
−3
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

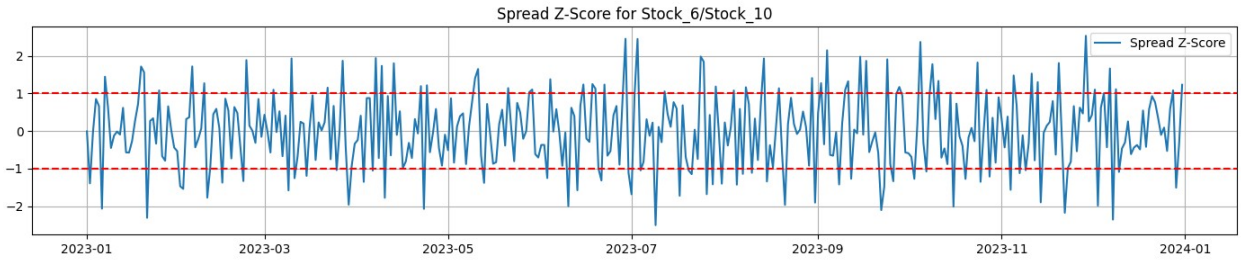

Spread for Stock_6/Stock_8
Spread
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

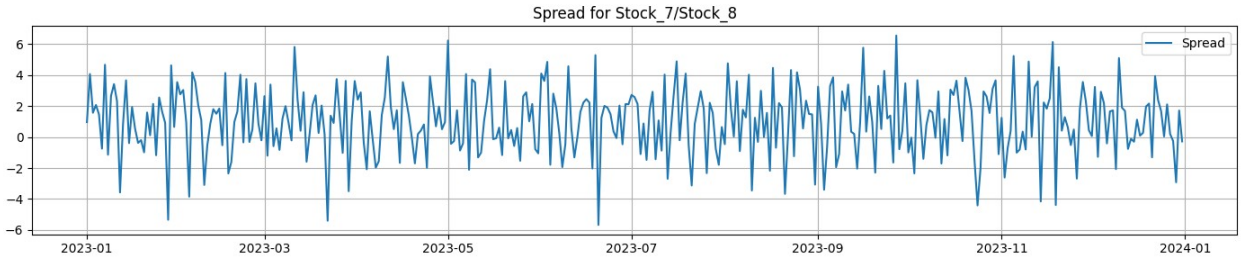

Spread Z-Score for Stock_6/Stock_8
Spread Z-Score
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

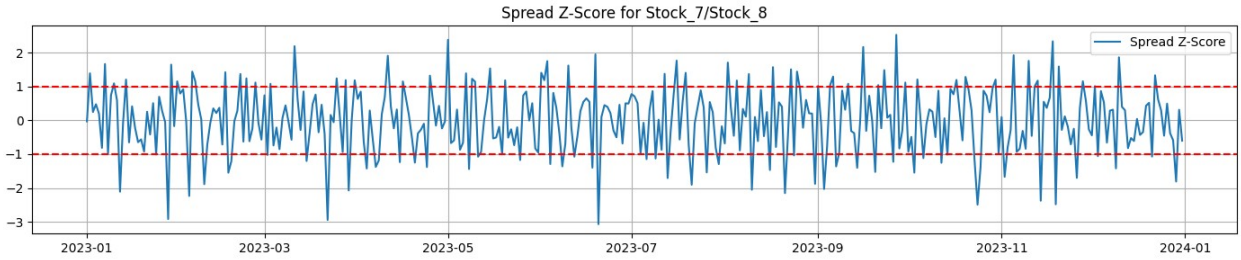

Spread for Stock_6/Stock_9
Spread
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

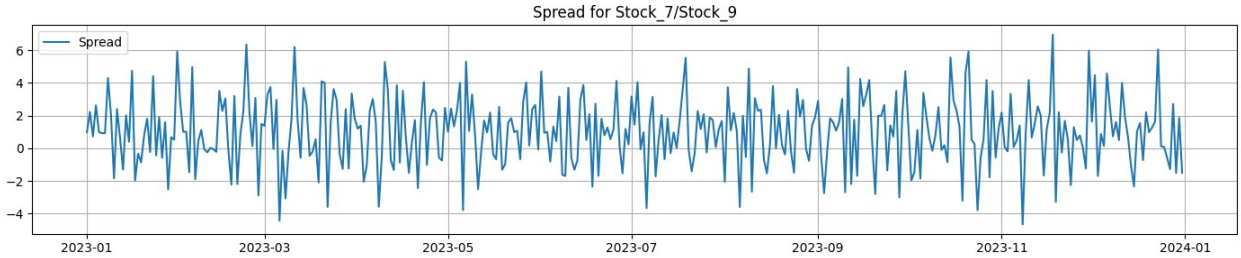

Spread Z-Score for Stock_6/Stock_9
Spread Z-Score
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

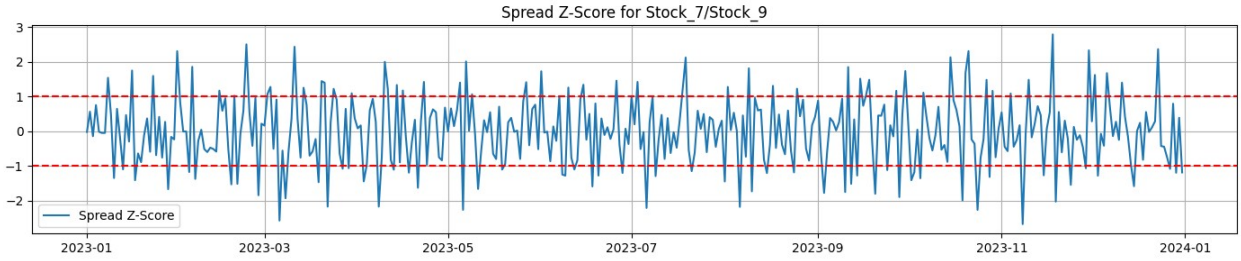

Spread for Stock_6/Stock_10
Spread
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

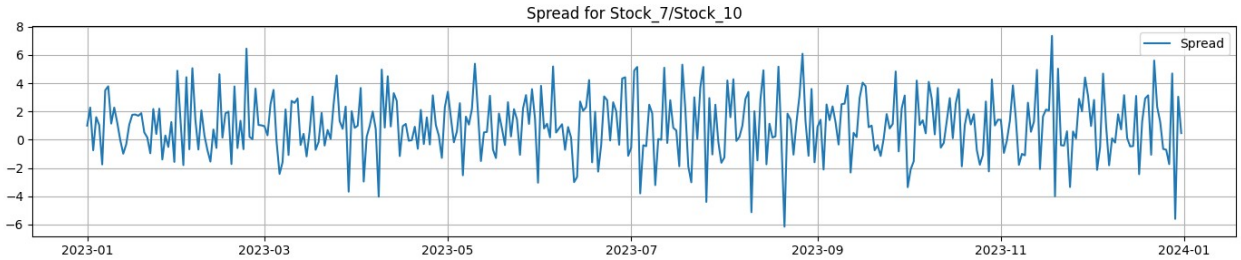

Spread Z-Score for Stock_6/Stock_10
Spread Z-Score
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

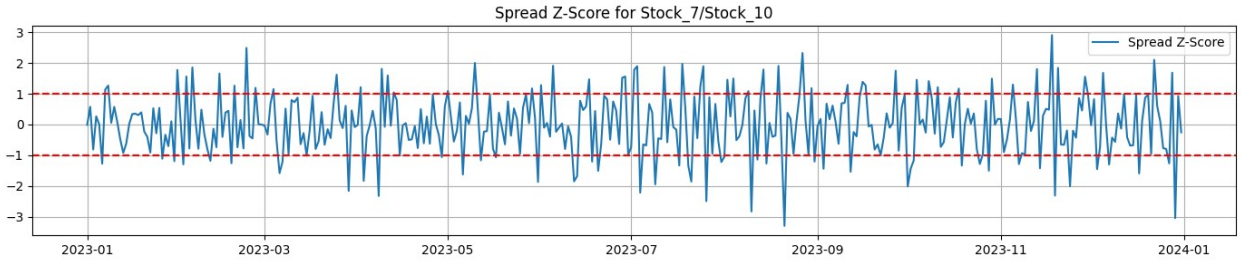

Spread for Stock_7/Stock_8
Spread
6
4
2
0
−2
−4
−6
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

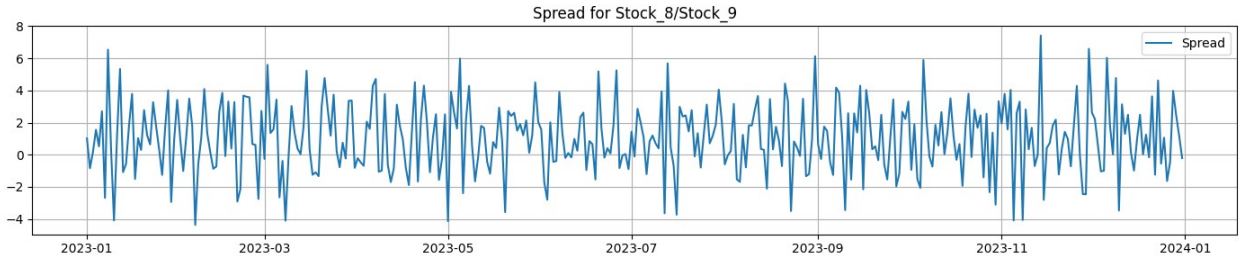

Spread Z-Score for Stock_7/Stock_8
Spread Z-Score
2
1
0
−1
−2
−3
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

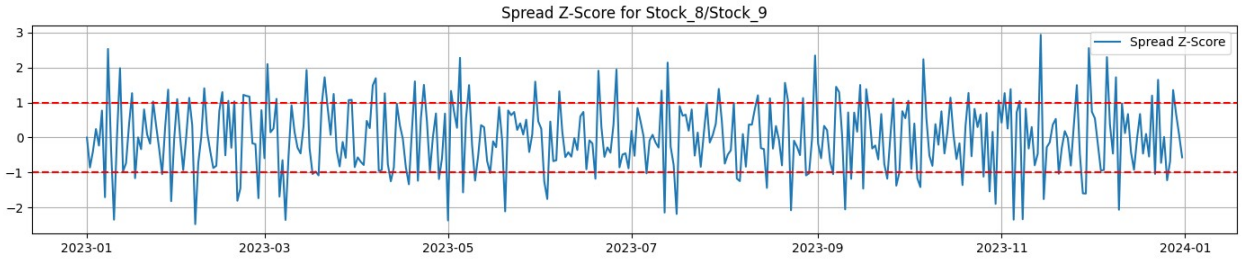

Spread for Stock_7/Stock_9
Spread
6
4
2
0
−2
−4
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

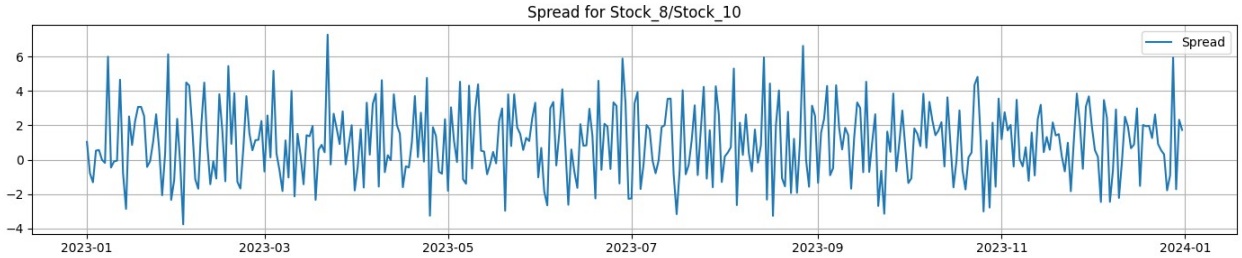

Spread Z-Score for Stock_7/Stock_9
Spread Z-Score
3
2
1
0
−1
−2
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

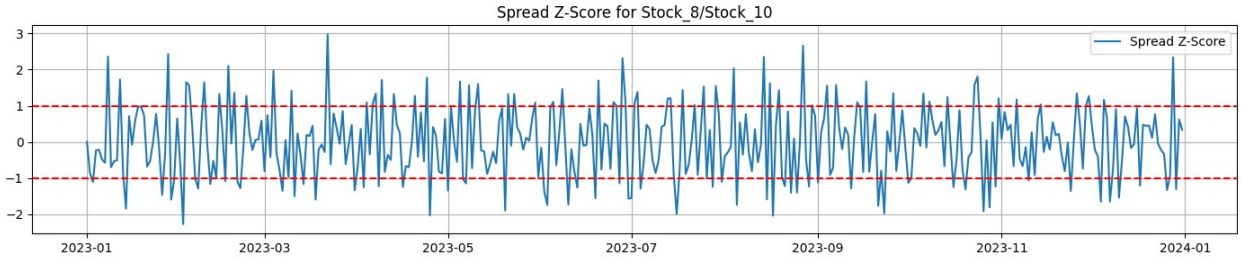

Spread for Stock_7/Stock_10
Spread
8
6
4
2
0
−2
−4
−6
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

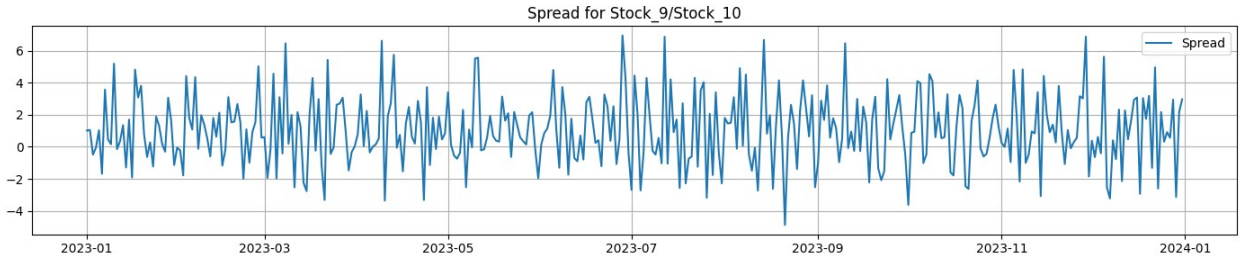

Spread Z-Score for Stock_7/Stock_10
Spread Z-Score
3
2
1
0
−1
−2
−3
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

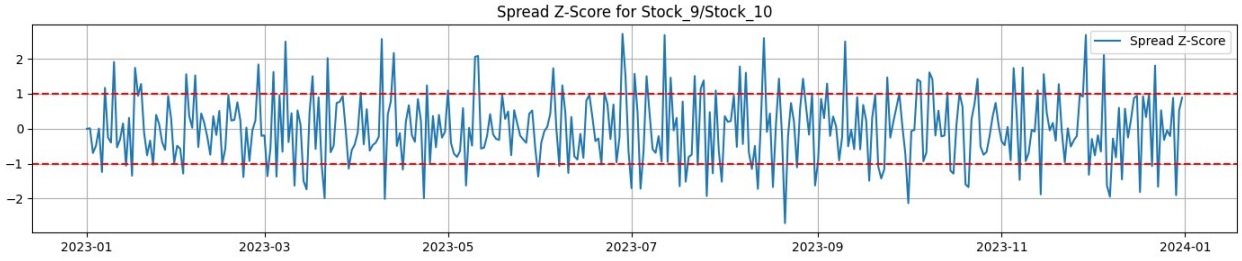

Spread for Stock_8/Stock_9
Spread
8
6
4
2
0
−2
−4
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

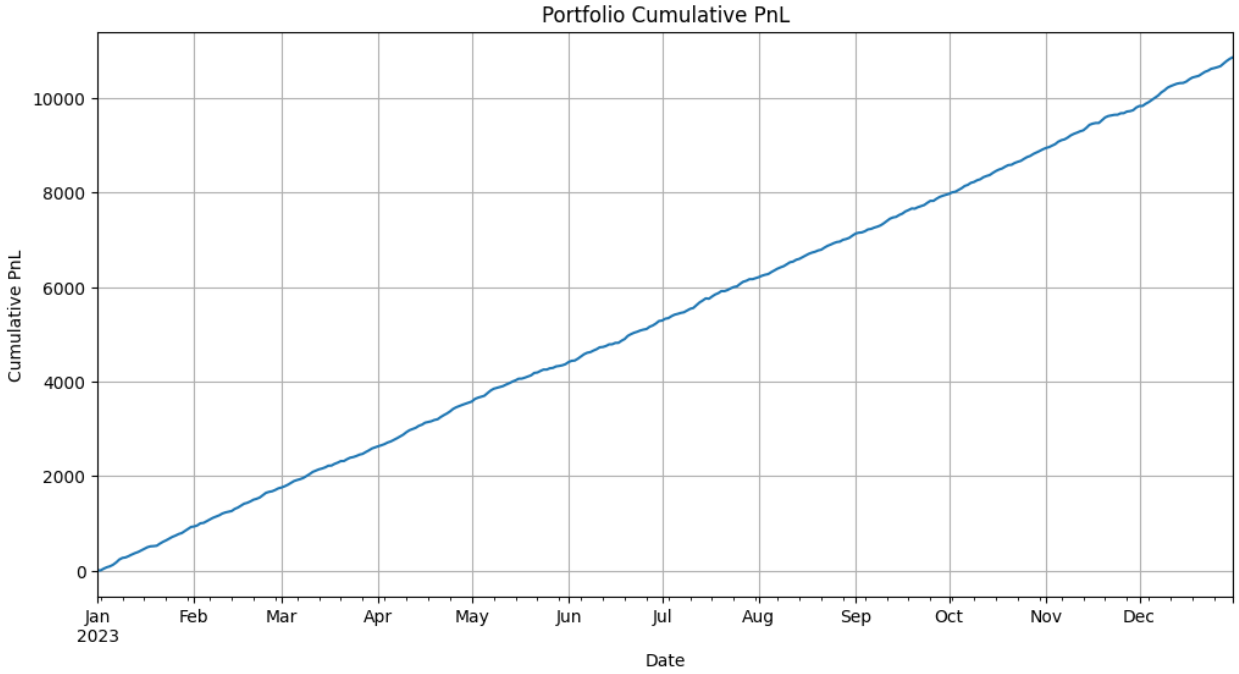

Spread Z-Score for Stock_8/Stock_9
Spread Z-Score
3
2
1
0
−1
−2
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01


Spread for Stock_8/Stock_10
Spread
6
4
2
0
−2
−4
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01


Spread Z-Score for Stock_8/Stock_10
Spread Z-Score
3
2
1
0
−1
−2
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01


Spread for Stock_9/Stock_10
Spread
6
4
2
0
−2
−4
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01


Spread Z-Score for Stock_9/Stock_10
Spread Z-Score
2
1
0
−1
−2
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01


Portfolio Cumulative PnL
Cumulative PnL
10000
8000
6000
4000
2000
0
Jan
2023
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Date