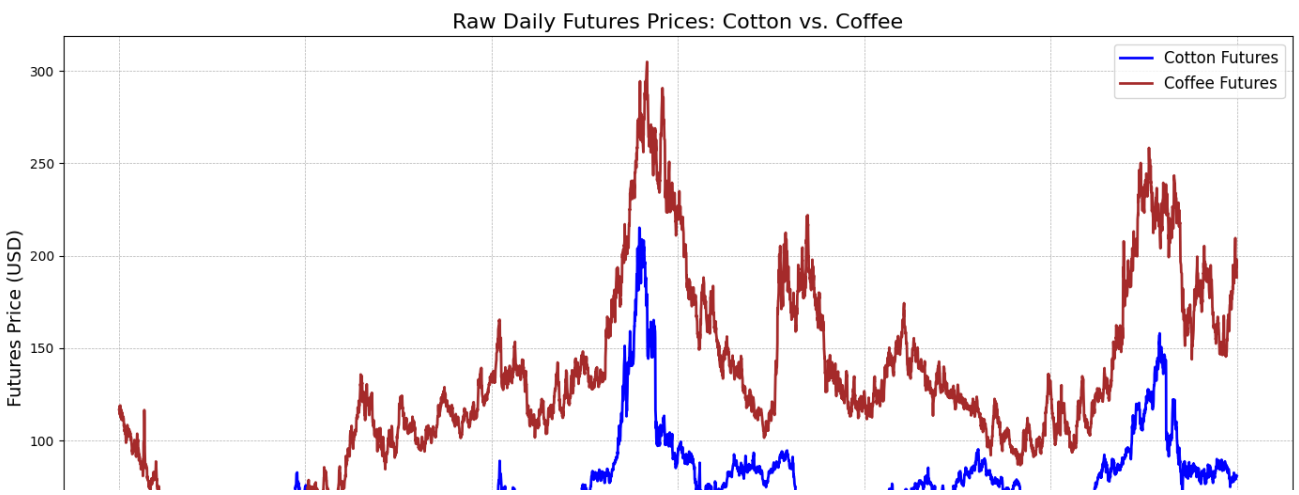
```
Start coding or generate with AI.
```

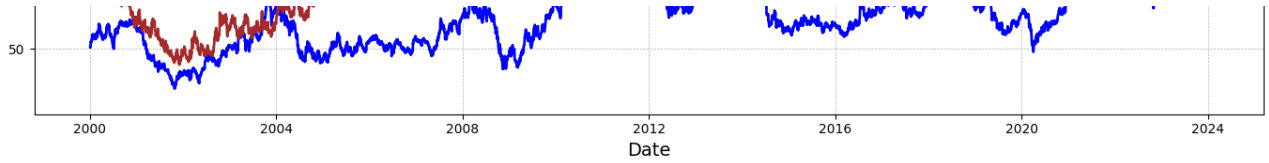
FM5222: Pairs Trading Project -- Caleb Williams

```
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
```

Start looking at coffee and cotton..

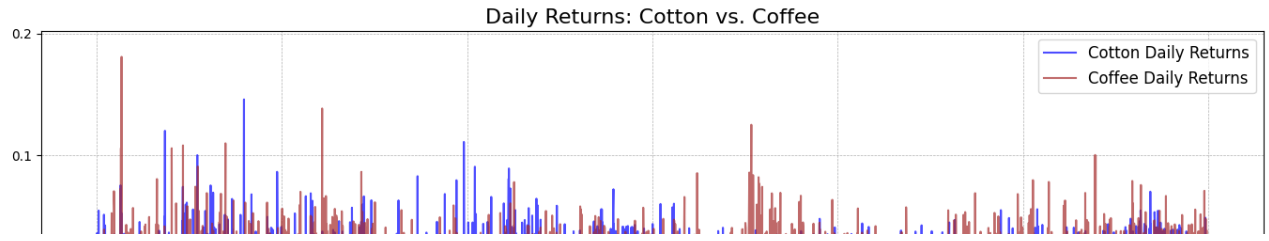
```
cotton = yf.Ticker("CT=F").history(start="1960-01-01", end="2024-01-01", interval='
coffee = yf.Ticker("KC=F").history(start="1960-01-01", end="2024-01-01", interval='

# Combine using the 'Close' prices
data = pd.DataFrame({
    'Cotton': cotton['Close'],
    'Coffee': coffee['Close']
}).dropna()

plt.figure(figsize=(14, 7))
plt.plot(data.index, data['Cotton'], label='Cotton Futures', color='blue', linewidt
plt.plot(data.index, data['Coffee'], label='Coffee Futures', color='brown', linewic
plt.title('Raw Daily Futures Prices: Cotton vs. Coffee', fontsize=16)
plt.xlabel('Date', fontsize=14)
plt.ylabel('Futures Price (USD)', fontsize=14)
plt.legend(fontsize=12)
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.tight_layout()
plt.show()
```

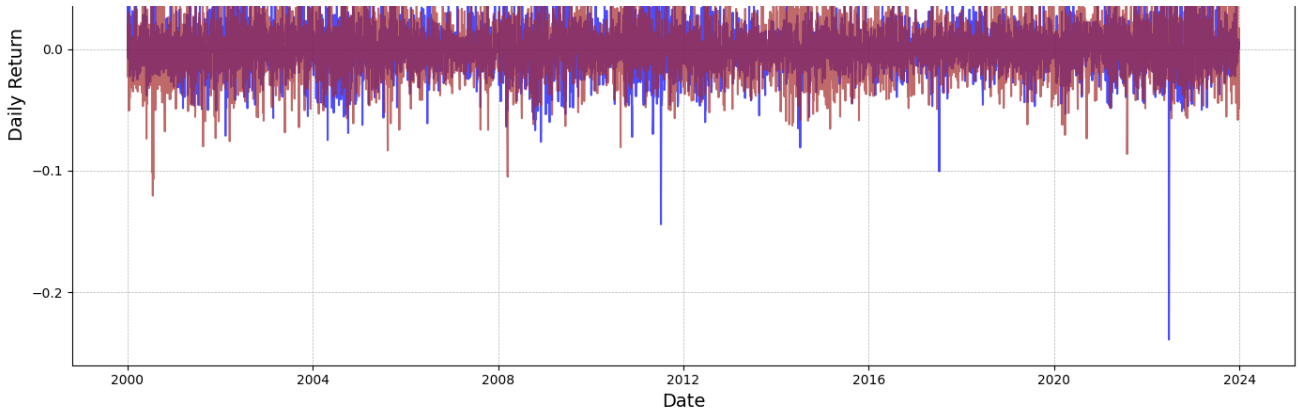




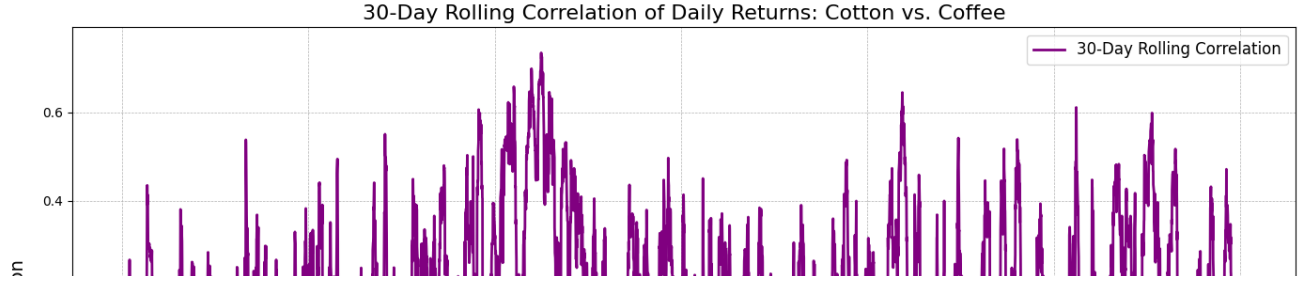
Looks like a promising start... lets continue to look at daily 'returns' on the futures contract...

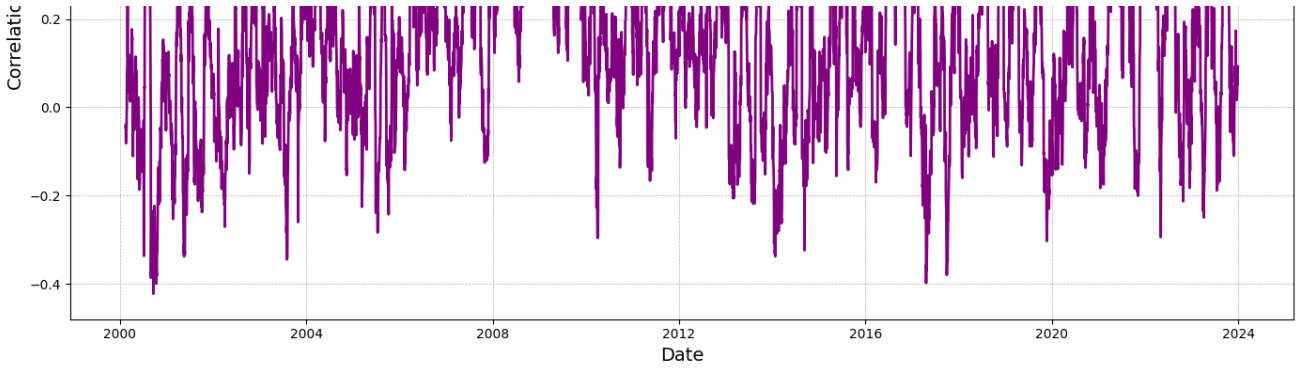
```
data['Cotton_Returns'] = data['Cotton'].pct_change()
data['Coffee_Returns'] = data['Coffee'].pct_change()

# Drop the first row that will be NaN
data = data.dropna()


plt.figure(figsize=(14, 7))
plt.plot(data.index, data['Cotton_Returns'], label='Cotton Daily Returns', color='b
plt.plot(data.index, data['Coffee_Returns'], label='Coffee Daily Returns', color='b
plt.title('Daily Returns: Cotton vs. Coffee', fontsize=16)
plt.xlabel('Date', fontsize=14)
plt.ylabel('Daily Return', fontsize=14)
plt.legend(fontsize=12)
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.tight_layout()
plt.show()
```






```
rolling_corr = data['Cotton_Returns'].rolling(window=30).corr(data['Coffee_Returns'

# Plot the 30-day rolling correlation
plt.figure(figsize=(14, 7))
plt.plot(rolling_corr.index, rolling_corr, label='30-Day Rolling Correlation', colo
plt.title('30-Day Rolling Correlation of Daily Returns: Cotton vs. Coffee', fontsiz
plt.xlabel('Date', fontsize=14)
plt.ylabel('Correlation', fontsize=14)
plt.legend(fontsize=12)
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.tight_layout()
plt.show()
```

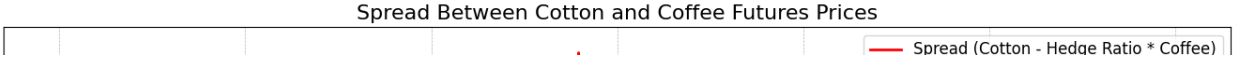





```
X = sm.add_constant(data['Coffee'])
model = sm.OLS(data['Cotton'], X).fit()
hedge_ratio = model.params['Coffee']
print("Estimated Hedge Ratio:", hedge_ratio)

# Get residuals
spread = data['Cotton'] - hedge_ratio * data['Coffee']
```

```
plt.figure(figsize=(14, 7))
plt.plot(spread.index, spread, label='Spread (Cotton - Hedge Ratio * Coffee)', colo
plt.title('Spread Between Cotton and Coffee Futures Prices', fontsize=16)
plt.xlabel('Date', fontsize=14)
plt.ylabel('Spread', fontsize=14)
plt.legend(fontsize=12)
plt.grid(True, linestyle='--', linewidth=0.5)
plt.tight_layout()
plt.show()
```

```
Estimated Hedge Ratio: 0.39783920591448557
```




```
from statsmodels.tsa.stattools import adfuller

adf_result = adfuller(spread)
print("ADF Statistic:", adf_result[0])
print("p-value:", adf_result[1])
```

```
ADF Statistic: -4.777259702878092
p-value: 6.010352083690663e-05
```

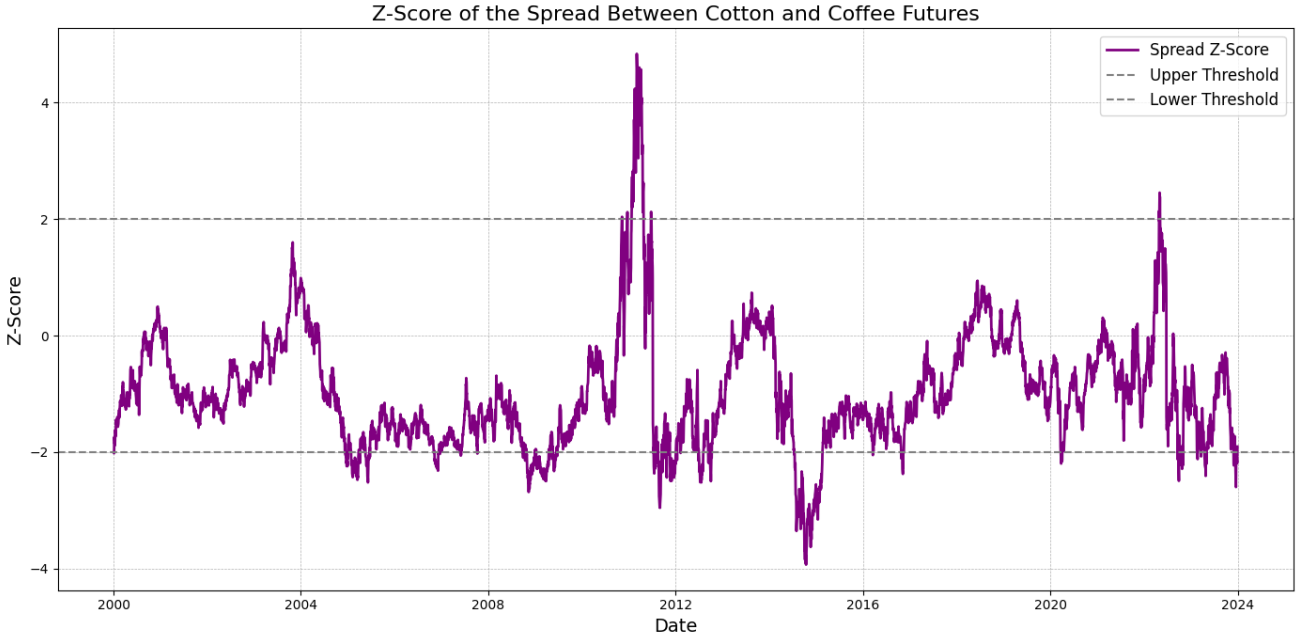
```
## Second Cointegration test

from statsmodels.tsa.stattools import coint

series1 = data['Coffee']
series2 = data['Cotton']
```

```
# Run the cointegration test
score, pvalue, _ = coint(series1, series2)

# Print the test statistic and p-value
print("Cointegration Test Statistic:", score)
print("Cointegration Test P-Value:", pvalue)
```

```
Cointegration Test Statistic: -4.702812155821395
Cointegration Test P-Value: 0.0005576197597411593
```

```
Cspread_mean = spread.mean()
spread_std = spread.std()
zscore = (spread - spread_mean) / spread_std
```

```
plt.figure(figsize=(14, 7))
plt.plot(zscore.index, zscore, label='Spread Z-Score', color='purple', linewidth=
plt.axhline(2, color='grey', linestyle='--', label='Upper Threshold')
plt.axhline(-2, color='grey', linestyle='--', label='Lower Threshold')
plt.title('Z-Score of the Spread Between Cotton and Coffee Futures', fontsize=16)
plt.xlabel('Date', fontsize=14)
plt.ylabel('Z-Score', fontsize=14)
plt.legend(fontsize=12)
plt.grid(True, linestyle='--', linewidth=0.5)
plt.tight_layout()
plt.show()
```

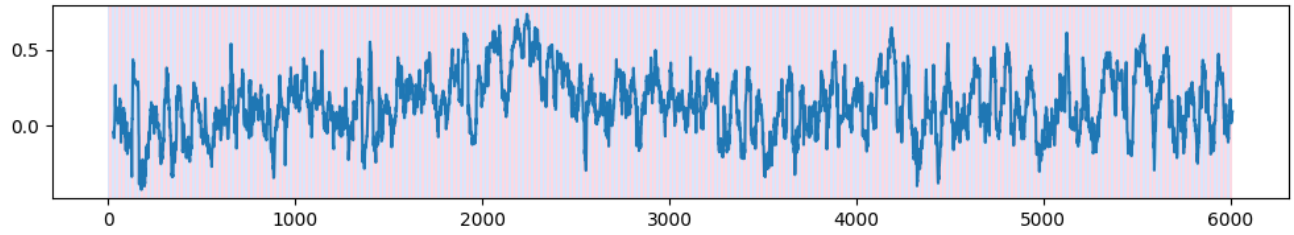


```
!pip install ruptures detecta
```

```
Requirement already satisfied: ruptures in /usr/local/lib/python3.11/dist-pacl
Requirement already satisfied: detecta in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packag
```

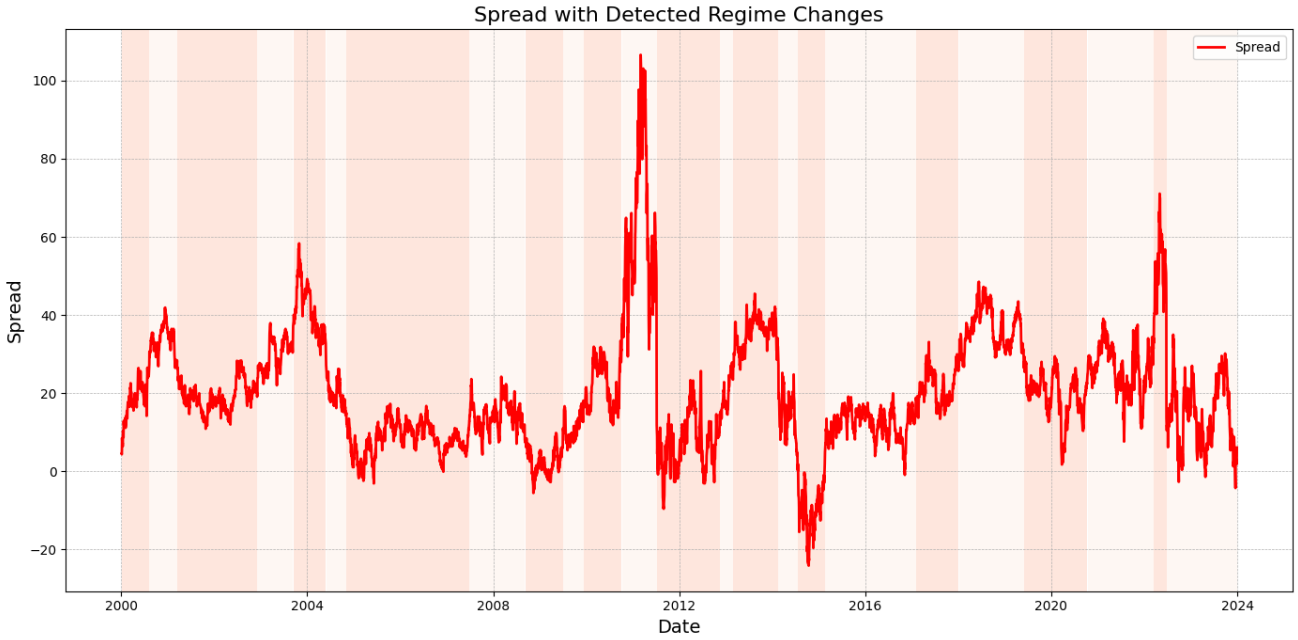
```
import ruptures as rpt
import detecta as dta

corr_values = rolling_corr.values

algo = rpt.Pelt(model='rbf').fit(corr_values)

result = algo.predict(pen=20)

rpt.display(corr_values, result)
```

```
(<Figure size 1000x200 with 1 Axes>, [<Axes: >])
```

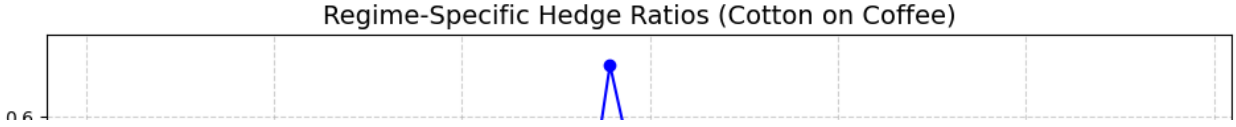


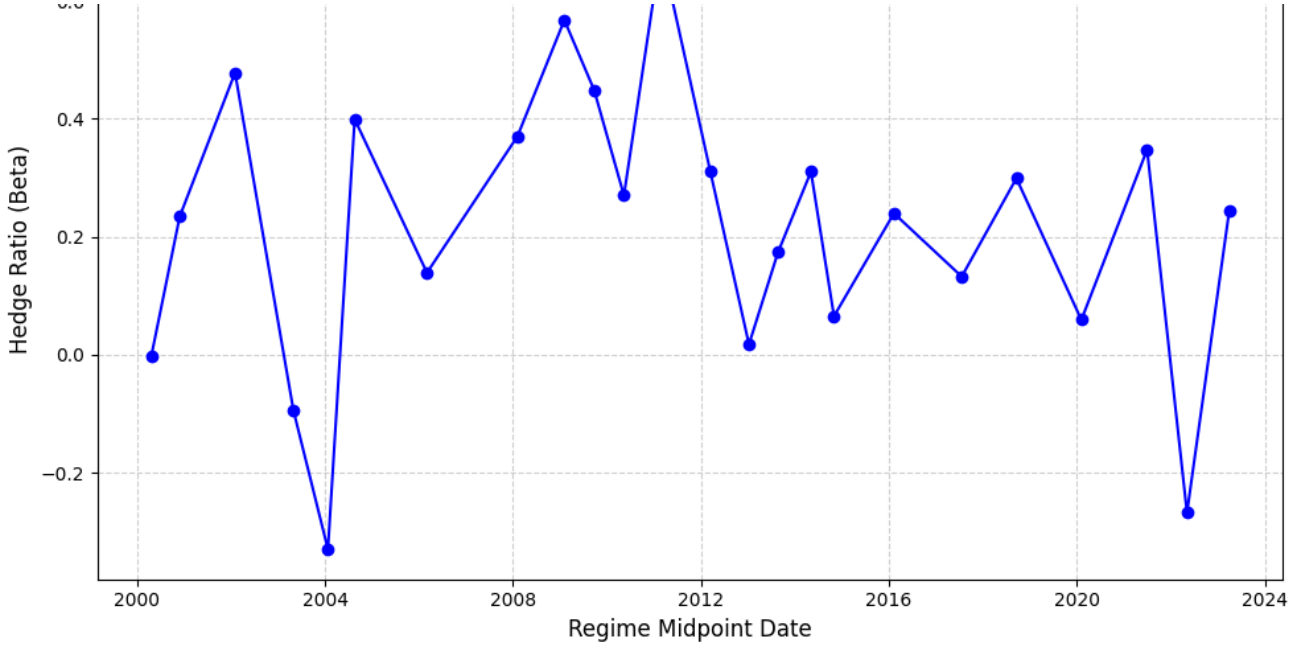

```
spread_values = spread.values

algo = rpt.Pelt(model="rbf").fit(spread_values)

breakpoints = algo.predict(pen=30)
```

```
plt.figure(figsize=(14, 7))
plt.plot(spread.index, spread, label='Spread', color='red', linewidth=2)

# Shade each segment
prev_idx = 0
colors = ['#fcae91', '#fee5d9']
color_idx = 0
for bp in breakpoints:
    plt.axvspan(spread.index[prev_idx], spread.index[bp-1],
                facecolor=colors[color_idx % len(colors)], alpha=0.3)
    prev_idx = bp
    color_idx += 1

plt.title("Spread with Detected Regime Changes", fontsize=16)
plt.xlabel("Date", fontsize=14)
plt.ylabel("Spread", fontsize=14)
plt.legend()
plt.grid(True, linestyle='--', linewidth=0.5)
plt.tight_layout()
plt.show()
```



```
## Just to see how optimal hedge ratio changes for each of the regimes found by r

breakpoints = sorted(breakpoints)
if breakpoints[-1] != len(data):
    breakpoints.append(len(data))
regime_dates = []   # To store a representative date for each regime (e.g., mid-d
hedge_ratios = []   # To store the hedge ratio for each regime

prev_idx = 0
for bp in breakpoints:
    # Extract the segment
    segment = data.iloc[prev_idx:bp]

    if len(segment) < 10:
        prev_idx = bp
        continue  # Skip segments that are too short

    # OLS
    X = sm.add_constant(segment['Coffee'])
    model = sm.OLS(segment['Cotton'], X).fit()
    hedge_ratio = model.params['Coffee']
    hedge_ratios.append(hedge_ratio)

    mid_date = segment.index[len(segment)//2]
    regime_dates.append(mid_date)

    prev_idx = bp

plt.figure(figsize=(10, 6))
plt.plot(regime_dates, hedge_ratios, marker='o', linestyle='-', color='blue')
plt.title('Regime-Specific Hedge Ratios (Cotton on Coffee)', fontsize=14)
plt.xlabel('Regime Midpoint Date', fontsize=12)
plt.ylabel('Hedge Ratio (Beta)', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```





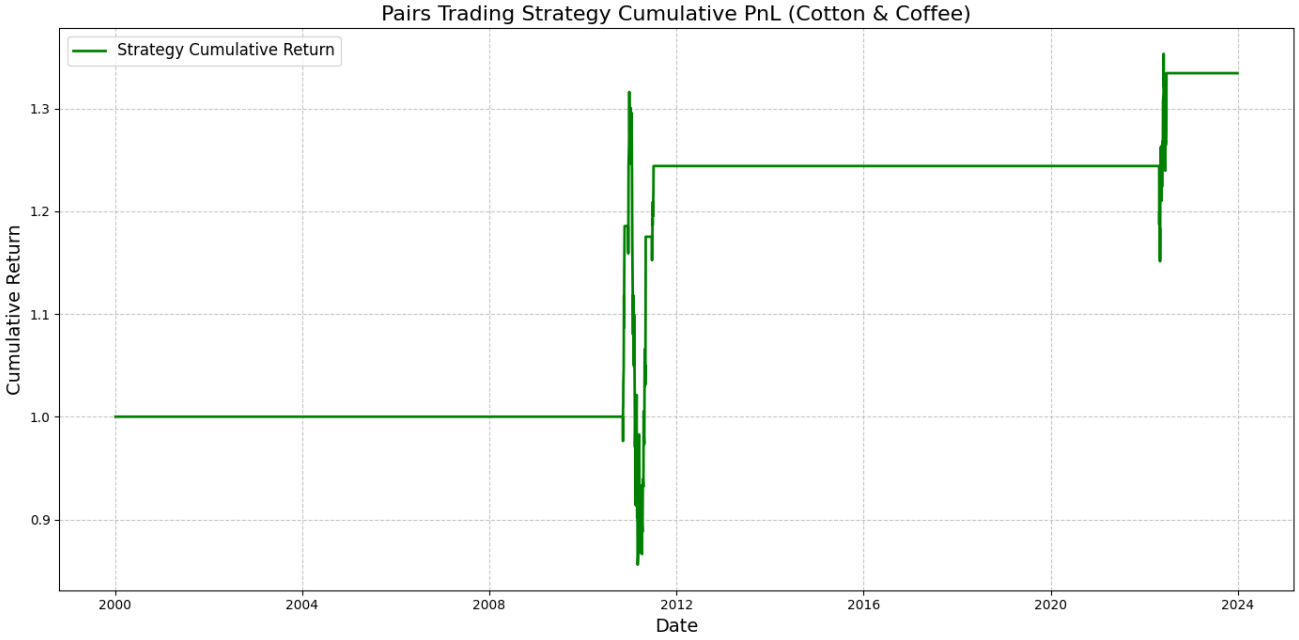

```
import statsmodels.tsa.stattools as ts

## Just checking to see whether cointegration holds within each of the regimes

breakpoints = sorted(breakpoints)
if breakpoints[-1] != len(data):
    breakpoints.append(len(data))

# Loop over each regime
start_idx = 0
for i, bp in enumerate(breakpoints):
    segment = data.iloc[start_idx:bp]

    if len(segment) < 30:
        print(f"Regime {i+1} is too short for reliable testing (length = {len(seg
        start_idx = bp
        continue

    coint_t, p_value, crit_values = ts.coint(segment['Cotton'], segment['Coffee']
    print(f"Regime {i+1} ({segment.index[0]} to {segment.index[-1]}): p-value = {
```

```
    start_idx = bp

     Regime 1 (2000-01-04 00:00:00-05:00 to 2000-08-08 00:00:00-04:00): p-value = (
     Regime 2 (2000-08-09 00:00:00-04:00 to 2001-03-15 00:00:00-05:00): p-value = (
     Regime 3 (2001-03-16 00:00:00-05:00 to 2002-12-05 00:00:00-05:00): p-value = (
     Regime 4 (2002-12-06 00:00:00-05:00 to 2003-09-12 00:00:00-04:00): p-value = (
     Regime 5 (2003-09-15 00:00:00-04:00 to 2004-05-27 00:00:00-04:00): p-value = (
     Regime 6 (2004-05-28 00:00:00-04:00 to 2004-11-03 00:00:00-05:00): p-value = (
     Regime 7 (2004-11-04 00:00:00-05:00 to 2007-06-26 00:00:00-04:00): p-value = (
     Regime 8 (2007-06-27 00:00:00-04:00 to 2008-09-05 00:00:00-04:00): p-value = (
     Regime 9 (2008-09-08 00:00:00-04:00 to 2009-06-30 00:00:00-04:00): p-value = (
     Regime 10 (2009-07-01 00:00:00-04:00 to 2009-12-11 00:00:00-05:00): p-value =
     Regime 11 (2009-12-14 00:00:00-05:00 to 2010-10-06 00:00:00-04:00): p-value =
     Regime 12 (2010-10-07 00:00:00-04:00 to 2011-07-11 00:00:00-04:00): p-value =
     Regime 13 (2011-07-12 00:00:00-04:00 to 2012-11-14 00:00:00-05:00): p-value =
     Regime 14 (2012-11-15 00:00:00-05:00 to 2013-02-27 00:00:00-05:00): p-value =
     Regime 15 (2013-02-28 00:00:00-05:00 to 2014-02-18 00:00:00-05:00): p-value =
     Regime 16 (2014-02-19 00:00:00-05:00 to 2014-07-18 00:00:00-04:00): p-value =
     Regime 17 (2014-07-21 00:00:00-04:00 to 2015-02-17 00:00:00-05:00): p-value =
     Regime 18 (2015-02-18 00:00:00-05:00 to 2017-01-31 00:00:00-05:00): p-value =
     Regime 19 (2017-02-01 00:00:00-05:00 to 2018-01-05 00:00:00-05:00): p-value =
     Regime 20 (2018-01-08 00:00:00-05:00 to 2019-05-30 00:00:00-04:00): p-value =
     Regime 21 (2019-05-31 00:00:00-04:00 to 2020-10-09 00:00:00-04:00): p-value =
     Regime 22 (2020-10-12 00:00:00-04:00 to 2022-03-16 00:00:00-04:00): p-value =
     Regime 23 (2022-03-17 00:00:00-04:00 to 2022-06-27 00:00:00-04:00): p-value =
     Regime 24 (2022-06-28 00:00:00-04:00 to 2023-12-29 00:00:00-05:00): p-value =
```

Backtest

```
threshold_entry = 3.0 # Z score
threshold_exit  = 1.0


data['Signal'] = 0.0

position = 0  # current position (1 = long spread, -1 = short spread, 0 = flat)
signals = []

for z in zscore:
    if position == 0:
        # Not in a trade: check entry conditions.
        if z < -threshold_entry:
            position = 1  # go long spread: long cotton, short coffee
        elif z > threshold_entry:
            position = -1  # go short spread: short cotton, long coffee
    else:
        # In a trade: check exit condition.
        if abs(z) < threshold_exit:
            position = 0
    signals.append(position)
```

```
data['Signal'] = signals


data['Strategy_Return'] = 0.0
data.loc[data['Signal'] == 1, 'Strategy_Return'] = data['Cotton_Returns'] - hedge
data.loc[data['Signal'] == -1, 'Strategy_Return'] = -data['Cotton_Returns'] + hed

data['Cumulative_Strategy'] = (1 + data['Strategy_Return']).cumprod()

plt.figure(figsize=(14, 7))
plt.plot(data.index, data['Cumulative_Strategy'], label='Strategy Cumulative Retu
plt.title('Pairs Trading Strategy Cumulative PnL (Cotton & Coffee)', fontsize=16)
plt.xlabel('Date', fontsize=14)
plt.ylabel('Cumulative Return', fontsize=14)
plt.legend(fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

print(f"Final return: {max(data['Cumulative_Strategy']):0.2f}")
```




```
Final return: 1.35
```

Find current conditions of this trade

```
cotton = yf.Ticker("CT=F").history(start="1960-01-01", end="2025-04-04", interval='
coffee = yf.Ticker("KC=F").history(start="1960-01-01", end="2025-04-04", interval='

data = pd.DataFrame({
    'Cotton': cotton['Close'],
    'Coffee': coffee['Close']
}).dropna()


data['Cotton_Returns'] = data['Cotton'].pct_change()
data['Coffee_Returns'] = data['Coffee'].pct_change()

# Drop the first row that will be NaN
data = data.dropna()

X = sm.add_constant(data['Coffee'])
model = sm.OLS(data['Cotton'], X).fit()
hedge_ratio = model.params['Coffee']
print("Estimated Hedge Ratio:", hedge_ratio)

# Get residuals
spread = data['Cotton'] - hedge_ratio * data['Coffee']


adf_result = adfuller(spread)
print("ADF Statistic:", adf_result[0])
print("p-value:", adf_result[1])


spread_mean = spread.mean()
spread_std = spread.std()
zscore = (spread - spread_mean) / spread_std


plt.figure(figsize=(14, 7))
plt.plot(zscore.index, zscore, label='Spread Z-Score', color='purple', linewidth=2)
plt.axhline(3, color='grey', linestyle='--', label='Upper Threshold')
plt.axhline(-3, color='grey', linestyle='--', label='Lower Threshold')
plt.title('Z-Score of the Spread Between Cotton and Coffee Futures', fontsize=16)
```

```
plt.xlabel('Date', fontsize=14)
plt.ylabel('Z-Score', fontsize=14)
plt.legend(fontsize=12)
plt.grid(True, linestyle='--', linewidth=0.5)
plt.tight_layout()
plt.show()
```

```
Estimated Hedge Ratio: 0.2735009275761203
ADF Statistic: -2.7843017300146764
p-value: 0.06055137871298016
```