WEEK1: Introduction To Pairs Trading

Nimay Shah, Devavrat Patni

December 7, 2024

Abstract

This document serves as a supplementary guide to understanding pairs trading with Kalman filtering. The focus is on the theoretical foundation and practical implementation of the strategy, emphasizing key mathematical concepts like stationarity, integration, and cointegration. Examples and visualizations are provided to enhance understanding.

Contents

1 Introduction to Pairs Trading and Kalman Filtering

Pairs trading is a form of *mean-reversion* that has a distinct advantage of always being *hedged* against market movements. It is generally a *high alpha* strategy when backed up by rigorous statistics. In this section, we delve into the foundational concepts behind this strategy and its implementation. Read up the following definitions to get a flavour of what pairs trading is.

1.1 Detailed Explanation of the Definition

Mean-Reversion: Mean-reversion is a statistical phenomenon where a variable tends to move towards its historical average or mean over time. In the context of pairs trading, it implies identifying financial instruments (such as stocks or ETFs) whose prices have a predictable relationship, and betting that deviations from this relationship will revert to the mean.

Hedged: Being hedged in trading means reducing exposure to systematic market risks. Pairs trading achieves this by taking simultaneous long and short positions on two related assets. For example, if the market moves sharply upward or downward, the losses in one position are partially or fully offset by gains in the other.
High Alpha Strategy: Alpha represents the excess returns of a trading strategy over a market benchmark. A high alpha strategy, like pairs trading, aims to generate consistent returns independent of general market trends. This is achieved by exploiting inefficiencies or mispricings in the relationship between paired assets, i.e. statistical arbitrage in effect.

Rigorous Statistics: Successful pairs trading relies on robust statistical methods to identify and exploit opportunities. Techniques like cointegration testing, linear regression, and, in this case, Kalman filtering, are used to model the dynamic relationship between asset prices and predict profitable trading opportunities. With time, we will dive into more detailed explanation for the same.

1.2 Hypothesis Testing

Hypothesis testing is a fundamental statistical tool used to evaluate assumptions (or hypotheses) about a dataset. It involves the following key elements:

Null Hypothesis (H_0): The null hypothesis represents the default assumption or claim. For example, in pairs trading, a null hypothesis could be that two securities **are not** cointegrated.(you can think about this as correlation for now)

Alternative Hypothesis (H_a): The alternative hypothesis is the opposite of the null hypothesis, representing a different claim. In the same example, the alternative hypothesis would be that the securities **are** cointegrated.

Test Statistic: A test statistic is a value computed from the sample data that is compared to a critical value or threshold to determine whether to reject H_0. Examples include z-scores, t-statistics, or the test statistic from the Augmented Dickey-Fuller (ADF) test(no need to know what these do, they are just tests to find out which of the HYPOTHESIS hold true).

Significance Level (α): The significance level is the probability of rejecting the null hypothesis when it is actually true. Common values for α include 0.05 and 0.01.

P-value: The p-value quantifies the evidence against the null hypothesis. A smaller p-value indicates stronger evidence to reject H_0. For instance, if the p-value is less than α, we reject H_0.

Decision: Based on the comparison of the test statistic with the critical value or the p-value with α, we either reject or fail to reject H_0. In pairs trading, hypothesis testing is crucial for determining whether the relationship between two securities is statistically significant.

2 Stationarity and Non-Stationarity

Stationarity is a commonly untested assumption in time series analysis. We generally assume that a dataset is *stationary* when the parameters of its data-generating process do not change over time. To illustrate, consider two time series: A and B.

- **Series A:** This is a stationary time series where parameters like the mean and standard deviation remain constant over time. It reflects a stable system whose statistical properties do not depend on the time at which they are measured.

- **Series B:** This is a non-stationary time series where the parameters vary with time. For instance, the mean might increase steadily as time progresses, indicating a trend or evolving dynamics in the system.

Mathematical Context

The concept of stationarity can be understood through the probability density function (PDF) of a Gaussian distribution:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{1}$$

Here:

- μ represents the mean of the distribution.
- σ denotes the standard deviation.
- σ^2 is the variance, which is the square of the standard deviation.

The *empirical rule* states that approximately 66% of the data in a Gaussian distribution lies within $\mu \pm \sigma$. This means that random samples are more likely to cluster around the mean.

Illustration of Stationary vs. Non-Stationary Time Series

To illustrate the difference between stationary and non-stationary time series:

- For **Stationary Series** A, the mean (μ) and variance (σ^2) remain constant throughout the time period. This stability ensures that statistical methods designed for stationary data yield reliable results.

- For **Non-Stationary Series** B, the mean evolves over time, such as increasing linearly. This dependency on time violates the assumption of stationarity and often requires techniques like differencing or detrending for proper analysis.

Visualization

- The stationary series appears to fluctuate consistently around a fixed mean.

- The non-stationary series exhibits a clear trend, where the mean shifts as time progresses.

Most statistical methods, including stationarity tests and Kalman filtering, assume stationarity in the underlying data.

2.1 Augmented Dickey-Fuller Test (ADF)

The Augmented Dickey-Fuller (ADF) test is a statistical tool used to determine if a time series is stationary. This section explains the mathematical foundations and notations used in the ADF test.

Unit Root and Non-Stationarity

Definition of a Unit Root:

A unit root in a time series occurs when the coefficient α in the following equation equals 1:

$$Y_t = \alpha Y_{t-1} + X_t + \epsilon_t$$

where:

- Y_t: Value of the time series at time t.
- Y_{t-1}: Lagged value of the time series (from the previous time step).
- X_t: Exogenous variable (an explanatory variable or deterministic component like a trend or constant).
- ϵ_t: White noise error term.

Interpretation of $\alpha = 1$:

When $\alpha = 1$, the time series exhibits a random walk:

$$Y_t = Y_{t-1} + X_t + \epsilon_t$$

This implies that shocks to the system (ϵ_t) have a persistent effect, and the series does not revert to a mean value, making it non-stationary. Conversely, when $|\alpha| < 1$, the series is stationary, as it tends to revert to a mean and has constant variance over time.

Differencing and Stationarity:

The number of unit roots in a time series indicates the number of differencing operations needed to make it stationary. For example, if a series has one unit root ($\alpha = 1$), taking the first difference:

$$\Delta Y_t = Y_t - Y_{t-1}$$

can remove the non-stationarity, resulting in a stationary series.

Notations and Their Meaning

- y_t: The value of the time series at time t.
- ϕ: The autoregressive coefficient of the time series. It determines the dependence of y_t on its lagged value y_{t-1}.
- $I(d)$: The order of integration of the series, where d is the minimum number of differencing operations needed to make the series stationary.

- Δy_t: The first difference of the series, defined as $\Delta y_t = y_t - y_{t-1}$.
- ε_t: The error term or residual at time t, assumed to be white noise.
- $\hat{\phi}$: The least squares estimate of the autoregressive coefficient ϕ.
- $SE(\hat{\phi})$: The standard error of the estimate $\hat{\phi}$.
- T: The number of observations in the time series.
- $\phi(\ddagger)$: The autoregressive polynomial, expressed as $\phi(\ddagger) = 1 - \phi\ddagger$.

Mathematical Foundations

The Hypothesis Test

The ADF test is framed as a hypothesis test:

$$\begin{aligned} H_0 : \phi = 1 \quad &\implies \quad y_t \sim I(1) \quad \text{(unit root, non-stationary)} \\ H_1 : |\phi| < 1 \quad &\implies \quad y_t \sim I(0) \quad \text{(stationary)} \end{aligned}$$

- Under H_0, the time series has a unit root and is non-stationary.
- Under H_1, the time series is stationary.

Unit Root and Trend Stationarity

A unit root occurs when the autoregressive polynomial:

$$\phi(\ddagger) = 1 - \phi\ddagger$$

has a root equal to 1, indicating a non-stationary process.

Trend stationarity assumes that y_t can be represented as:

$$y_t = \delta t + \Delta\ddagger_t$$

where δ is a deterministic trend.

First Differencing

To remove the trend and test for stationarity, we difference the series:

$$\Delta y_t = \delta + \Delta\ddagger_t$$

Here:

$$\Delta\ddagger_t = \phi\Delta\ddagger_{t-1} + \varepsilon_t - \varepsilon_{t-1}$$

Test Statistic

The test statistic is computed as:

$$t_{\phi=1} = \frac{\hat{\phi} - 1}{SE(\hat{\phi})}$$

- $\hat{\phi}$: The estimated value of the autoregressive coefficient.
- $SE(\hat{\phi})$: The standard error of $\hat{\phi}$, representing the uncertainty in the estimate.

Distribution of the Test Statistic

Under Stationarity (H_1)

If the series is stationary, the test statistic $t_{\phi=1}$ follows a normal distribution:

$$t_{\phi=1} \overset{\text{A}}{\sim} N(0,1)$$

Under Non-Stationarity (H_0)

If the series is non-stationary, the distribution of the test statistic diverges, and the time series retains a unit root.

Practical Interpretation

The ADF test determines whether a time series is stationary:

- A p-value below a chosen significance level (e.g., 0.05) implies rejection of the null hypothesis, concluding that the series is stationary.
- A p-value above the significance level indicates failure to reject the null hypothesis, suggesting that the series is non-stationary.

If the math seems too complicated, its not an issue you can skip this for now. The more important point to understand is how we can use this test and the statistic obtained to test for the stationarity of a process.

3 Cointegration

Cointegration is a statistical concept used to determine whether two or more time series have a long-term equilibrium relationship, even if the individual series themselves are non-stationary. This section explains the steps and mathematical foundations of cointegration tests.

Steps for Testing Cointegration

The process of testing for cointegration involves the following steps:
1. **Unit Root Testing**: Test each time series y_t individually for a unit root using methods such as the Augmented Dickey-Fuller (ADF) test or the Phillips-Perron (PP) test.
2. **Cointegration Testing**: If the unit root cannot be rejected for each series (indicating non-stationarity), test for cointegration among the time series. This involves testing whether a linear combination of the time series is stationary ($I(0)$).
If the series are found to have a unit root(i.e. unit root can't be rejected), we proceed to test for cointegration.

Methods for Cointegration Testing

There are three main methods for testing cointegration:

- Johansen Test
- Engle-Granger Test
- Phillips-Ouliaris Test

For simplicity, we focus on the Engle-Granger two-step test.

Regression Model for Cointegration

The Engle-Granger method uses the following regression model to test for cointegration:

$$y_{1t} = \delta D_t + \phi_1 y_{2t} + \phi_2 y_{3t} + \cdots + \phi_{m-1} y_{mt} + \varepsilon_t$$

Here:

- $y_{1t}, y_{2t}, \ldots, y_{mt}$: The time series under consideration.
- D_t: The deterministic term, which could include a constant, trend, or seasonal components.
- $\phi_1, \phi_2, \ldots, \phi_{m-1}$: The coefficients of the cointegrating relationship.
- ε_t: The residual term, which we test for stationarity.

The null hypothesis (H_0) and alternative hypothesis (H_1) for the test are:

$$\begin{aligned} H_0 &: \varepsilon_t \sim I(1) \quad \text{(no cointegration)} \\ H_1 &: \varepsilon_t \sim I(0) \quad \text{(cointegration)} \end{aligned}$$

If ε_t is stationary ($I(0)$), the time series y_t are considered cointegrated, with the cointegration vector $\alpha = (1, \phi_1, \ldots, \phi_{m-1})$.

Residual-Based Unit Root Test

To confirm cointegration, we apply a unit root test to the residuals ε_t of the regression. The model for the residuals is:

$$\Delta\varepsilon_t = \lambda\varepsilon_{t-1} + \sum_{j=1}^{p-1} \varphi_j \Delta\varepsilon_{t-j} + \alpha_t$$

Where:

- $\Delta\varepsilon_t = \varepsilon_t - \varepsilon_{t-1}$: The first difference of the residuals.
- λ: The coefficient for the lagged residuals ε_{t-1}.
- φ_j: The coefficients for additional lagged differences of the residuals.
- α_t: The error term, assumed to be white noise.

The hypothesis test for the residuals is:

$$\begin{aligned} H_0 &: \lambda = 0 \quad \text{(unit root, non-stationary)} \\ H_1 &: \lambda < 1 \quad \text{(stationary)} \end{aligned}$$

Test Statistic

The test statistic for the residual-based unit root test is given by:

$$t_\lambda = \frac{\hat{\lambda}}{SE(\hat{\lambda})}$$

Where:

- $\hat{\lambda}$: The estimated value of λ from the regression model.
- $SE(\hat{\lambda})$: The standard error of $\hat{\lambda}$.

This statistic follows a non-standard distribution under the null hypothesis, and critical values are provided for different significance levels.

Interpretation of Cointegration

- If ε_t is stationary ($I(0)$), the time series are cointegrated, indicating a long-term equilibrium relationship.
- If ε_t is non-stationary ($I(1)$), the time series are not cointegrated, implying no long-term relationship.

4 Trading Signals

When conducting any type of trading strategy, it's essential to clearly define the points at which trades will be executed. That is, what is the best indicator to determine when to buy or sell a particular stock? This section will explore how to define trading signals using a ratio time series and how these signals guide buying and selling decisions in a pair trading strategy.

Setup Rules

We are going to use the ratio time series created between two stocks to determine whether to buy or sell at any given moment in time. The ratio represents the relative movement between the two stocks, and the signal to trade is based on changes in this ratio. To predict the direction of the ratio movement, we first create a prediction variable Y.

The prediction rule is defined as follows:

$$Y_t = \text{sign}(\text{Ratio}_{t+1} - \text{Ratio}_t)$$

Where:

- Y_t is the trading signal at time t.
- Ratio_t represents the ratio between the two stocks at time t.
- $\text{sign}(x)$ is the sign function, which returns:
  - 1 for positive changes in the ratio (indicating a "buy" signal).
  - -1 for negative changes in the ratio (indicating a "sell" signal).

The ratio moving upwards signals a "buy", and the ratio moving downwards signals a "sell". The strength of this approach is that we do not need to predict the absolute future price of the stocks but merely whether the ratio will move up or down. (If this confuses you know, read further ahead and it will make more sense then).

4.1 Feature Engineering

To improve the model's ability to predict the direction of the ratio's movement, we need to identify key features that help determine the ratio's future direction. Given that the ratios tend to revert to their mean over time, moving averages and statistical measures related to the mean are likely to be useful features for the model.

Here are the features we will use:

- 60-day Moving Average of the Ratio: This helps capture long-term trends.
- 5-day Moving Average of the Ratio: This helps capture short-term fluctuations.
- 60-day Standard Deviation: This measures the volatility of the ratio over the long term.
- Z-score: The Z-score is a normalized value that indicates how many standard deviations the current ratio is from its mean.

The following Python code demonstrates how to calculate these features:

$$\text{ratios\_mavg5} = \text{train}.rolling(window = 5, center = False).mean()$$

$$\text{ratios\_mavg60} = \text{train}.rolling(window = 60, center = False).mean()$$

$$\text{std\_60} = \text{train}.rolling(window = 60, center = False).std()$$

$$\text{zscore\_60\_5} = \frac{\text{ratios\_mavg5} - \text{ratios\_mavg60}}{\text{std\_60}}$$

The following plot shows the ratio alongside its 5-day and 60-day moving averages:

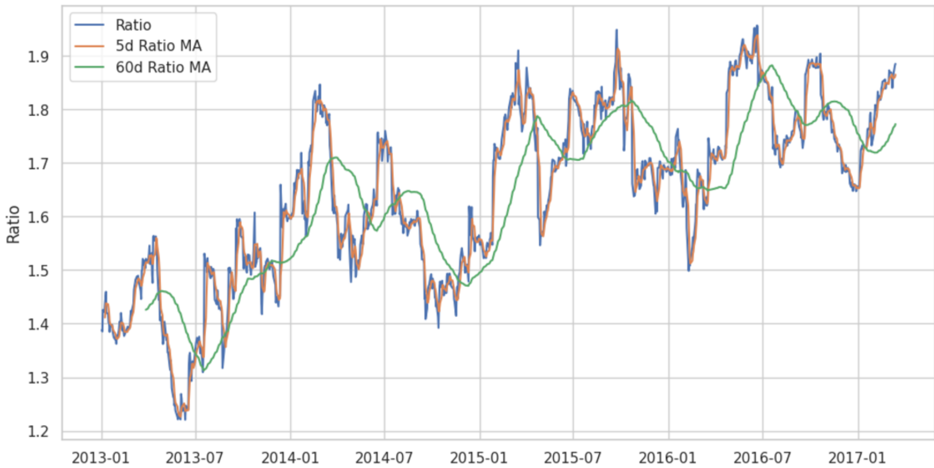


Figure 1:

In the plot, the blue line represents the ratio, the orange line represents the 5-day moving average, and the green line represents the 60-day moving average. By examining these moving averages, we can get a sense of whether the ratio is trending upward or downward over different time horizons.

4.2 Creating a Model

The key idea in pair trading is that prices tend to revert to the mean. To model this behavior, we look at the Z-score, which tells us how far the ratio is from its moving average in terms of standard deviations. A Z-score greater than 1 or less than -1 indicates that the ratio is either far from its mean and likely to revert, providing a signal for trade. Here's how we can define trading signals based on the Z-score:

- **Buy Signal (1):** If the Z-score is below -1, we expect the ratio to increase (i.e., revert to the mean), so we signal a "buy."
- **Sell Signal (-1):** If the Z-score is above 1, we expect the ratio to decrease, signaling a "sell."

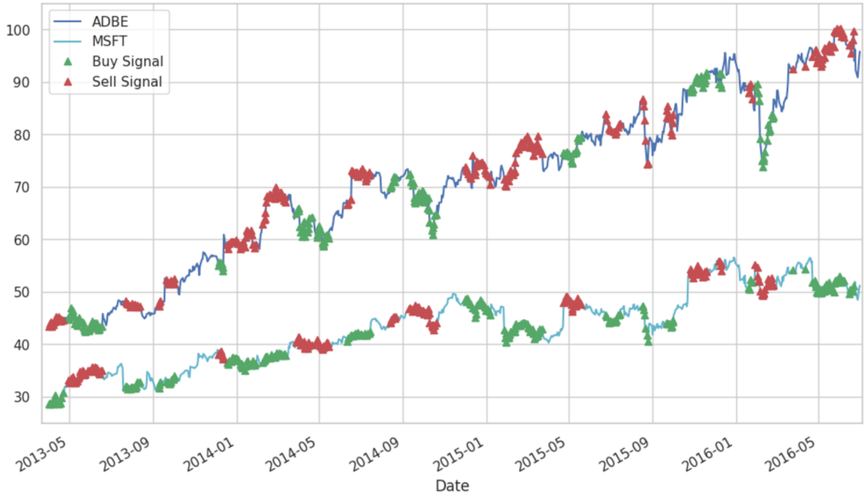
We use these Z-score thresholds to identify when to enter or exit trades. For the following, the Z-score is computed using the 60-day moving average and the 5-day moving average, divided by the standard deviation of the 60-day moving average.

The following code generates the buy and sell signals based on the Z-score:

$$\text{buy}[zscore\_60\_5 > -1] = 0$$

$$\text{sell}[zscore\_60\_5 < 1] = 0$$

A plot of the trading signals overlaid on the ratio time series is shown below. Green triangles represent buy signals, and red triangles represent sell signals:



4.3 Implementing the Trading Strategy

Once we have defined the buy and sell signals, we can implement the trading strategy by simulating the buy and sell actions using the defined Z-score thresholds. In this strategy, the goal is to trade pairs of stocks where the ratio between the two stock prices exhibits mean-reverting behavior.

To implement this strategy, we simulate the following actions:

- When the Z-score is less than -1, it triggers a "buy" signal for the ratio, meaning we go long on the first stock and short on the second stock.
- When the Z-score is greater than 1, it triggers a "sell" signal for the ratio, meaning we short the first stock and go long on the second stock.
- If the Z-score is between -0.75 and 0.75, we exit any open positions, effectively clearing the portfolio.

The following Python function simulates the trading strategy:

```
def trade(S1, S2, window1, window2):
    # If window length is 0, algorithm doesn't make sense, so exit
    if (window1 == 0) or (window2 == 0):
        return 0

    # Compute rolling mean and rolling standard deviation
    ratios = S1/S2
    ma1 = ratios.rolling(window=window1, center=False).mean()
    ma2 = ratios.rolling(window=window2, center=False).mean()
```

```python
    std = ratios.rolling(window=window2, center=False).std()
    zscore = (ma1 - ma2)/std

    # Simulate trading
    # Start with no money and no positions
    money = 0
    countS1 = 0
    countS2 = 0
    for i in range(len(ratios)):
        # Sell short if the z-score is > 1
        if zscore[i] < -1:
            money += S1[i] - S2[i] * ratios[i]
            countS1 -= 1
            countS2 += ratios[i]
        # Buy long if the z-score is < -1
        elif zscore[i] > 1:
            money -= S1[i] - S2[i] * ratios[i]
            countS1 += 1
            countS2 -= ratios[i]
        # Clear positions if the z-score between -.5 and .5
        elif abs(zscore[i]) < 0.75:
            money += S1[i] * countS1 + S2[i] * countS2
            countS1 = 0
            countS2 = 0

    return money
```

This function simulates the trading activity for two stocks, where $S1$ and $S2$ represent the price series of two stocks, and ‘window1‘ and ‘window2‘ define the lengths of the moving averages and standard deviation windows. The function returns the total money after applying the trading strategy.

4.4 Conclusion

The pair trading strategy outlined in this section uses the mean-reverting behavior of stock price ratios and the Z-score as a key indicator to decide when to enter or exit trades. By using rolling windows to compute moving averages and standard deviations, the model generates buy and sell signals that align with market trends, helping to capitalize on mean reversion in stock price ratios. This strategy can be further refined by optimizing the window sizes or incorporating additional features to improve prediction accuracy.

5 Where Kalman Filter Comes In

While iterating over various window lengths can help identify the optimal trading window, we must also be cautious of the risk of overfitting. Overfitting occurs when a model performs exceptionally well on historical data but fails to generalize to new, unseen data. This issue arises because the model has simply memorized the historical data rather than uncovering genuine patterns or trends that would hold in future market conditions. Thus, finding a robust and reliable window length is essential to avoid the overfitting pitfall.
In our model, we used rolling parameter estimates to generate signals for trading, but we may want to go further and optimize the window length. To achieve this, we can iterate over all possible, reasonable window lengths and select the one that maximizes our profit and loss (PnL) on the training data. Here's a simple approach to find the optimal window length:

```
# Find the window length 0-254
# that gives the highest returns using this strategy
length_scores = [trade(df['ADBE'].iloc[:1057],
                df['MSFT'].iloc[:1057], l, 5)
                for l in range(255)]
best_length = np.argmax(length_scores)
print ('Best window length:', best_length)

length_scores2 = [trade(df['ADBE'].iloc[1057:],
                  df['MSFT'].iloc[1057:],l,5)
                  for l in range(255)]
print (best_length, 'day window:', length_scores2[best_length])
# Find the best window length based on this dataset,
# and the returns using this window length
best_length2 = np.argmax(length_scores2)
print (best_length2, 'day window:', length_scores2[best_length2])
```

In the above code, we test all possible window lengths from 0 to 254 and compute the PnL for both training and testing data. The best window length corresponds to the one that maximizes the PnL. From the results, it is evident that using a window length above around 50 days offers a reasonable choice for our trading strategy. However, we must avoid picking a length that is too short and susceptible to noise, which would overfit our model.

To further reduce the risk of overfitting, we can use economic reasoning to choose the window length. Trading decisions driven purely by data can be misleading if not backed by a solid economic rationale. Therefore, combining data-driven insights with fundamental knowledge of market cycles, company-specific events, and economic factors can help refine our window selection process.

Moreover, an even more sophisticated approach would be to replace static window lengths with adaptive techniques. One such technique is the **Kalman filter**, which is a recursive method for estimating the state of a system from noisy measurements. The Kalman filter doesn't require us to specify a fixed window length, and it can dynamically adjust its estimates based on incoming data.