Risk Management in Pairs Trading Using Kalman Filtering

Manya Shah

Mentor: Nimay Shah and Devarat Patni

January 05, 2025

1 Introduction

This report presents the implementation of a pairs trading strategy incorporating risk management techniques and dynamic hedge ratio adjustment using Kalman Filtering. The objective is to enhance the robustness and profitability of the strategy by applying key techniques like stop-loss orders, position sizing, and volatility threshold.

2 Mathematical Background

2.1 Kalman Filtering

Kalman Filtering is used to dynamically estimate and adjust the hedge ratio between two stocks in a pair. The Kalman Filter uses a series of measurements observed over time, containing statistical noise and other inaccuracies, to produce estimates of unknown variables that tend to be more precise than those based on a single measurement alone.

$$x_{k|k-1} = Ax_{k-1|k-1} + Bu_k$$

$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q$$

$$K_k = P_{k|k-1}H^T(HP_{k|k-1}H^T + R)^{-1}$$

$$x_{k|k} = x_{k|k-1} + K_k(z_k - Hx_{k|k-1})$$

$$P_{k|k} = (I - K_kH)P_{k|k-1}$$

2.2 Cointegration Tests

Cointegration tests are used to identify pairs of stocks that have a stable, long-term relationship. The Engle-Granger test is one of the methods employed for this purpose.

3 Risk Management Techniques

3.1 Stop-Loss Orders

Stop-loss orders are used to limit losses by exiting a trade when the spread between two stocks exceeds a predefined threshold. The threshold is determined based on historical data or risk tolerance levels.

$$\text{Exit Condition:} \quad |\text{Spread}| > \text{Threshold}$$

```
def stop_loss(spread, threshold):
    return spread > threshold
```

3.2 Position Sizing

Position sizing involves determining the size of the trade based on the volatility of the spread. A common method is to use the inverse of volatility:

$$\text{Position Size} = \frac{\text{Risk-Factor}}{\text{Volatility}}$$

```
def position_sizing(spread, risk_factor):
    spread = pd.Series(spread)
    spread_volatility = spread.rolling(window=30).std()
    position_size = risk_factor / spread_volatility
    return position_size.fillna(0)
```

3.3 Volatility Calculation

Volatility is a measure of the spread's variability and is calculated using the standard deviation over a rolling window:

$$\text{Volatility} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

```
def volatility_threshold(spread, threshold):
    spread_volatility = pd.Series(spread).rolling(window=30).std()
    return spread_volatility > threshold
```

4 Implementation Details

4.1 Data Acquisition

Data is acquired from Yahoo Finance using the 'yfinance' library in Python.
Below is the code snippet for fetching stock data:

```
tickers = ["MSFT", "AAPL"]
start_date="2013-01-01"
end_date="2019-01-01"
data = yf.download(tickers, start=start_date, end=end_date)["Close"]
```

4.2 Kalman Filter Implementation

The Kalman Filter is implemented to adjust the hedge ratio dynamically:

```
def KalmanFilterRegression(prices1, prices2):

    delta = 1e-5
    trans_cov = delta / (1 - delta) * np.eye(2)

    prices1 = np.asarray(prices1)
    prices2 = np.asarray(prices2)

    obs_mat = np.vstack([prices1, np.ones(prices1.shape)]).T[:, np.newaxis]

    kf = KalmanFilter(
        n_dim_obs=1,
        n_dim_state=2,
        initial_state_mean=np.zeros(2),
        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, state_covs = kf.filter(prices2)

    return state_means


def kalman_spread(price1, price2):
    state_means = KalmanFilterRegression(price2, price1)
    hedge_ratios = state_means[:, 0]
    alphas = state_means[:, 1]
    spread2 = price1 - hedge_ratios * price2 - alphas

    spread = abs(spread2)
    return pd.Series(spread, index=price1.index)
```

4.3 Backtesting

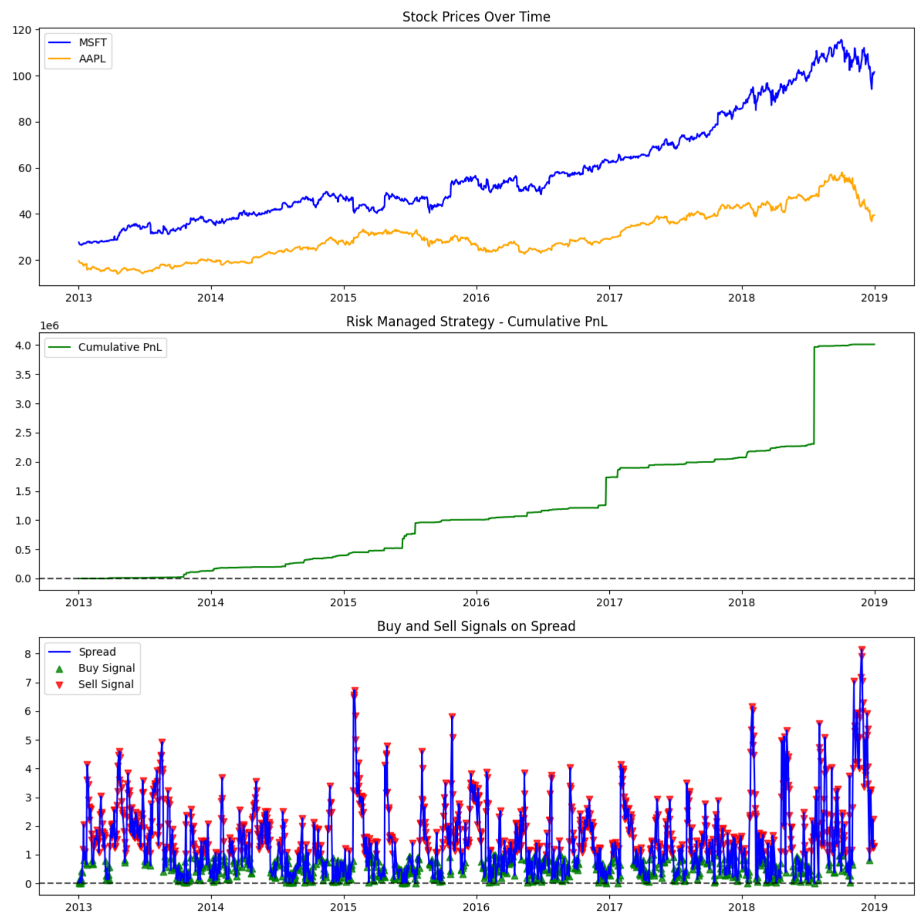
Backtesting is performed to evaluate the strategy's performance by applying buy and sell signals to historical market data while incorporating risk management metrics such as stop-loss, position sizing, and volatility thresholds. The evaluation uses cumulative PnL to assess the overall profitability of the strategy under various risk management condition

```
def backtest_with_risk_management(data, stock1, stock2, stop_loss_threshold=2, r
    spread = kalman_spread(data[stock1], data[stock2])

    buy_signals = spread < 1
    sell_signals = spread > 1

    stop_loss_flag = stop_loss(spread, stop_loss_threshold)
    volatility_flag = volatility_threshold(spread, volatility_threshold_value)

    position_size = position_sizing(spread, risk_factor) * 10000

    daily_returns = spread.pct_change().shift(-1).fillna(0)

    position = np.zeros(len(spread))


    position[buy_signals] = 1
    position[sell_signals] = -1
    position[stop_loss_flag] = 0
    position[volatility_flag] = position_size[volatility_flag]

    pnl = position * daily_returns
    cumulative_pnl = pnl.cumsum()
```

5 Results

5.1 Stock Prices, Cumulative PnL, and Buy/Sell Signals

The following graph shows the stock prices, cumulative PnL achieved, and the buy/sell signals generated during the backtesting period.



5.2 Cumulative PnL Achieved

The final cumulative PnL achieved is summarized as follows:

$$\text{Final Cumulative PnL:} \quad \$\ 4012842.61$$

.

6 Conclusion

This project demonstrates the effectiveness of risk management and dynamic hedge ratio adjustment in pairs trading. The results indicate that the strategy can improve profitability while managing risk effectively.