Project: Mean Reversion Calendar Spread Strategy

Author: Madhav Gupta

1. Strategy Overview

This project implements a statistical arbitrage strategy on NSE Futures Calendar Spreads (Near Month vs. Far Month). The hypothesis is that the spread between these two contracts is mean-reverting. We utilize a Z-Score based trigger to identify overextended spreads and capture the reversion to the mean.

2. Tradable Spread Identification

Analysis of initial backtests revealed that many pairs suffered from excessive slippage and transaction costs, turning gross profits into net losses (e.g., LODHA). To "identify tradable spreads," we implicitly filter for quality by:

- **Liquidity Thresholds:** Stocks with zero volume or insufficient data points to calculate Z-scores are automatically excluded.
- **High Conviction Entry:** We increased the Z-Score Entry Threshold to 2.0 (from 1.5). This acts as a quality filter, ensuring we only trade when the spread deviation is statistically significant, thereby improving the signal-to-noise ratio and reducing the impact of fixed transaction costs.

3. Trading Logic

- **Signal:** 60-day Rolling Z-Score of the Spread.
- **Entry:** |Z-Score| > 2.0.
- ■ Short Spread (Sell Near / Buy Far) if Z > 2.0.
  ■ Long Spread (Buy Near / Sell Far) if Z < -2.0.
- **Exit:**
- ■ Take Profit (TP): 0.5 Sigma relative to entry.
  ■ Stop Loss (SL): 1.5 Sigma relative to entry (Tightened to protect capital).
- **Expiry:** Force close all positions at End-Of-Day on Near Month Expiry.

4. Metrics & Assumptions

- **Max Gross Qty:** Calculated as the maximum absolute open position (lots) held at any minute.
- **Max Delta Qty:** Calculated as the maximum change in lots in a single minute (trade size).
- **Costs:** Slippage is calculated based on the observed Bid-Ask spread at the moment of execution. Commission is modelled at ~2 bps per leg.

5. Execution

The simulation is powered by a vectorized Python engine (*simulation_engine.py*) optimized for speed using *ProcessPoolExecutor*. The driver script (*problem2_runner.py*) aggregates these results and generates the final leaderboard.