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 & Optimization

To identify the most robust tradable spreads and minimize slippage impact, we conducted a rigorous analysis combining liquidity filtering with a parameter grid search.

- **Liquidity Filtering:** Stocks with zero volume or insufficient history for Z-score calculation were automatically excluded to prevent execution errors.
- **Grid Search Optimization:** We utilized a **Grid Search** (*run_grid.py*) to simulate combinations of Entry Thresholds (1.5, 2.0, 2.5), Take Profits (0.5, 1.0), and Stop Losses. The results were aggregated into a leaderboard.
- **Parameter Selection:** The leaderboard revealed that lower Entry thresholds (1.5) and tight Take Profits (0.5) consistently yielded negative Net PnL due to churn costs. The optimal configuration was identified as **Entry=2.5, TP=1.0**, which generated the highest Net PnL (~5.3L) by trading less frequently but with higher conviction.

3. Trading Logic

- **Signal:** 60-day Rolling Winsorized Z-Score of the Spread.
- **Entry:** |Z-Score| > 2.5 (Selected via Grid Search).
- ■ Short Spread (Sell Near / Buy Far) if Z > 2.5.
  ■ Long Spread (Buy Near / Sell Far) if Z < -2.5.
- **Exit:**
- ■ Take Profit (TP): 1.0 Sigma relative to entry (Allowed profits to run).
  ■ Stop Loss (SL): 2.0 Sigma relative to entry.
- **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*) runs the optimal parameter set and generates the final *Results.csv*.