Backtest Report

Universe: Subset of S&P 500 Components (N=100)
Timeframe: 2015-01-01 to 2024-12-31 (Daily Data)

---

1. Strategy Description

This report details the backtest of a mean-reversion trading strategy applied to a portfolio of US equities.
The strategy aims to identify potential short-term price reversals using a combination of technical indicators:

- **Relative Strength Index (RSI):** Measures the magnitude of recent price changes to evaluate overbought or oversold conditions.
- **Bollinger Bands (BBands):** Consist of a middle band (Simple Moving Average) and upper/lower bands based on standard deviations, indicating volatility and potential price extremes.
- **Volume Weighted Moving Average (VWMA):** A moving average that gives more weight to bars with higher volume.

Entry Logic

Long Entry (Strict):
RSI < 25 **AND** Close < Lower Bollinger Band **AND** Close > VWMA(50)

Long Entry (Aggressive):
RSI < 30 **AND** Close < Midpoint between BB-Mid and BB-Lower **AND** Close > VWMA(50)

Short Entry (Strict):
RSI > 75 **AND** Close > Upper Bollinger Band **AND** Close < VWMA(50)

Short Entry (Aggressive):
RSI > 70 **AND** Close > Midpoint between BB-Mid and BB-Upper **AND** Close < VWMA(50)

Exit Logic

Positions are held until the opposite signal is generated (implicitly handled by the portfolio rebalancing).

Parameters Used

Parameter	Value
RSI Period	5 days
Bollinger Bands Period	10 days
Bollinger Bands Std Dev	2.5
VWMA Period	50 days

---

2. Backtesting Process

A hybrid **Python–Java–Python** workflow was used:

Data Preparation (`prepare_data.py` - Python)

- Fetched S&P 500 components (limited to N=100).
- Downloaded daily OHLCV data from Yahoo Finance (`yfinance`).
- Calculated next-day returns.
- Saved processed data to `data_for_java.csv`.

Indicator Calculation & Backtesting (`StrategyProcessor.java` - Java)

- Read `data_for_java.csv`.
- Grouped data by ticker and date.
- Used **ta4j** to calculate RSI, Bollinger Bands, and VWMA.
- Applied `getSignal` logic per stock daily.
- Computed equal-weighted portfolio return:
  `(Sum of Long Returns - Sum of Short Returns) / Total Positions`.
- Saved daily portfolio returns to `portfolio_returns.csv`.
- Calculated performance metrics (Annualized Return, Volatility, Sharpe, Max Drawdown).

Analysis & Visualization (`analysis.py` - Python)

- Read `portfolio_returns.csv`.
- Performed Fama-French 3-Factor regression using `statsmodels`.
- Generated cumulative returns plot with `matplotlib`.

Assumptions

- No trading costs considered.
- Equal-weighted portfolio across all signaled positions daily.
- Trades executed at close of signal day, realized on next day's close.

---

3. Performance Metrics

Metric	Value
Annualized Return	**0.59%**
Annualized Volatility	**18.35%**
Sharpe Ratio	**0.03**
Maximum Drawdown	**-48.80%**

Interpretation

- Slightly positive but negligible return.
- Volatility similar to market levels.
- Sharpe Ratio near zero poor risk-adjusted performance.
- Max Drawdown (~49%) significant downside risk.

---

4. Factor Regression Analysis (Fama-French 3-Factor)

Regression on excess returns vs. **Mkt-RF**, **SMB**, and **HML** factors:

Factor	Coefficient	p-value	Interpretation

const (Alpha)	0.00002023	0.929	Statistically insignificant
Mkt-RF	-0.1373	0.000	Negative market exposure
SMB	0.1119	0.002	Tilt towards small-cap stocks
HML	-0.0940	0.000	Tilt towards growth stocks
R-squared	0.023	-	Only 2.3% of returns explained by factors

Annualized Alpha: 0.51% (p=0.93)
Market Beta: -0.14
Size Beta: +0.11
Value Beta: -0.09

Interpretation

- **Alpha:** Not significant — strategy doesn't outperform risk factors.
- **Market Beta:** Slightly anti-correlated with market; likely short bias.
- **Size/Value Exposure:** Prefers smaller, growth-oriented names.
- **Low R²:** Most variation is idiosyncratic (strategy-specific).

---

5. Trade and Risk Management Considerations

The current backtesting framework implemented in Java using ta4j is primarily vectorized. It calculates indicators and applies rules across the entire dataset or large chunks simultaneously, making it efficient for portfolio-level analysis but less suited for complex, stateful, trade-by-trade management logic.

Concept: Apply a fixed percentage or pip-based Stop Loss (SL) and Take Profit (TP) to each entered trade.

ta4j Approach: ta4j allows incorporating StopLossRule and StopGainRule into the strategy's exit logic. However, these rules are checked at every bar based on the entry price of the current active signal. In a portfolio context where signals might persist or re-trigger, this doesn't perfectly replicate managing a distinct entry with its specific SL/TP levels until that specific trade is closed. The current implementation omits these rules for simplicity, relying only on reversal signals for exits. Implementing fixed SL/TP would require modifications to the runBacktest method in Java to track individual entries, which ta4j is not primarily designed for.

Concept: Dynamically adjust the risk per trade (influencing position size) based on recent strategy performance, often using metrics like the System Quality Number (SQN). Good performance increases risk allocation, while poor performance decreases it ("portfolio heat").

ta4j Approach: Implementing this dynamic, path-dependent risk sizing is not feasible within a standard ta4j vectorized backtest. The calculation of the portfolio return on a given day (calculateDailyPortfolioReturn in Java) assumes equal weighting for simplicity. Implementing dynamic sizing based on SQN would require an event-driven simulation loop that tracks equity, calculates SQN over a rolling window of simulated trades, and adjusts the number of shares/contracts for each stock based on this risk factor before calculating the daily return. This is outside the scope of the current ta4j implementation.

Conclusion on Management Models: While the concepts of fixed SL/TP and dynamic risk sizing are crucial for real-world trading, directly implementing the specific logic from the Python synapse-based classes within this ta4j portfolio backtest is impractical due to framework differences. Portfolio-level risk in this vectorized context is better managed through techniques like position sizing rules applied uniformly (e.g., volatility parity), limiting the number of concurrent positions, or filtering the initial universe, rather than stateful, trade-by-trade adjustments based on P&L.

---

6. Conclusion and Next Steps

Summary

- **Performance:** Near-zero Sharpe, heavy drawdowns → ineffective as-is.
- **Risk Profile:** Negative beta, small-cap and growth tilt, low explanatory power.

Recommendations

Parameter Tuning

- Adjust RSI period (e.g., 14), BB period (e.g., 20), VWMA (20–100).
- Optimize for **Sharpe ratio**, not total return.

Indicator Variants

- Try **EMA** instead of **SMA** for BB midline.
- Explore other signals: moving average crossovers, fundamentals (P/E, P/B).

Signal Logic

- Refine thresholds (e.g., RSI 30/70).
- Add trend filters (e.g., trade only if price > EMA200).

Portfolio Construction

- Use **volatility-based** or **signal-strength-based weighting**.
- Filter out **illiquid stocks** by average volume.
- Trade only **top/bottom X%** of signals.

Risk Management

- Add **stop-loss** and **exposure control** during volatility spikes.

Out-of-Sample Testing

- Separate **in-sample** (training) vs **out-of-sample** (validation) data.
- Check robustness and prevent overfitting.
