

Methodology for Project: A Time Series Approach to Pairs Trading on the Indian Stock Market

Prepared by: Aryan Gupta, Billa Cherish, Vibha Gupta

Phase 1: Identifying Potential Pairs

In the first phase, we focus on finding pairs of stocks whose prices move together in the long run—a property called cointegration. We will specifically analyze a set of highly liquid stocks from the same economic sector (e.g., Banking or IT), obtained using the `yfinance` library in Python. Since these stocks belong to the same sector and are highly liquid, they are expected to exhibit strong co-movement, allowing us to proceed directly to formal cointegration tests to identify statistically significant relationships. The motivation is that even if two stocks drift apart over short periods, a confirmed cointegrating relationship can make the spread between them mean-reverting, which is useful for pairs trading.

Engle-Granger Cointegration Test

Step 1: Modelling Cointegration using Least Squares

For two stocks, we work with their log price series: $Y_t^{(x)}$ for stock x , and $Y_t^{(y)}$ for stock y . This logarithmic transformation helps stabilize variance and makes returns additive, which is useful in statistical models. To check for a long-run equilibrium relationship, we establish a model:

$$Y_t^{(y)} = \alpha + \eta Y_t^{(x)} + \varepsilon_t$$

where:

- α is the intercept, capturing any consistent difference in log prices,
- η is the hedge ratio, representing how much of one stock offsets changes in the other,
- ε_t is the residual/spread, showing the instantaneous deviation from the equilibrium.

The coefficients $\hat{\alpha}$ and $\hat{\eta}$ are estimated by minimizing the sum of the squared residuals ($\sum \varepsilon_t^2$).

The key output from this regression is the residual series:

$$\hat{\varepsilon}_t = Y_t^{(y)} - (\hat{\alpha} + \hat{\eta} Y_t^{(x)})$$

This spread tells us at each time point how far the prices are from their long-run relationship.

Step 2: Testing Residuals for Stationarity

To be useful for pairs trading, the spread series $\hat{\varepsilon}_t$, should be stationary, meaning it reverts to its mean over time. We check this using the Augmented Dickey-Fuller (ADF) test, which is a statistical test designed to identify stationarity in time series.

The ADF test model is:

$$\Delta \hat{\varepsilon}_t = \gamma \hat{\varepsilon}_{t-1} + \sum_{j=1}^p \psi_j \Delta \hat{\varepsilon}_{t-j} + u_t$$

where:

- $\Delta \hat{\varepsilon}_t$ is the change in spread from one time period to the next,
- γ describes the tendency of the spread to revert back to the mean,
- ψ_j represents the dependence on previous changes in the spread,
- u_t is random noise.

If the test rejects the null hypothesis that $\gamma = 0$ (i.e., the spread contains a unit root), then the spread is stationary. This confirms a cointegrated pair suitable for mean-reversion strategies.

Significance of Cointegration: Cointegration is more robust than correlation because it captures stable, long-term relationships even when individual stocks follow unpredictable paths. Trading the spread exploits temporary deviations, assuming the spread will eventually revert towards the long-term average.

Phase 2: Univariate Benchmark

Modelling the Price Series as an ARIMA

In this phase, we focus on modeling each individual stock time series independently, without considering their interrelationships. The objective is to establish a baseline performance using classical time series methods, specifically an ARIMA(p, d, q) model. These approaches are standard for forecasting temporal data which may have trend, autocorrelation, and possibly seasonality. We first apply a log transformation to the price series, $Y_t = \log(P_t)$. The ARIMA model then represents the stationary series $W_t = (1 - B)^d Y_t$ (after differencing d times):

$$\phi(B)W_t = \theta(B)\varepsilon_t$$

Here, $\phi(B)$ and $\theta(B)$ are polynomials in the lag operator B (where $BY_t = Y_{t-1}$), representing the AR and MA components respectively, and ε_t is a white noise error term.

Step 1: Automated Model Selection (auto_arima)

We will use an automated algorithm like `auto_arima` (from the `pmdarima` library) to systematically determine the optimal model orders (p, d, q). This function automates the following statistical procedures:

- **Determining d :** It runs sequential unit root tests (e.g., ADF, testing $H_0 : \gamma = 0$ in $\Delta W_t = c + \gamma W_{t-1} + \dots$) to find the minimum differencing d required to achieve stationarity.
- **Finding p, q :** It conducts a stepwise search over a range of p and q values. For each combination, it estimates the parameters using Maximum Likelihood Estimation (MLE), which finds the parameters $(\hat{\phi}, \hat{\theta})$ that maximize the log-likelihood function, $\ln(L(\hat{\phi}, \hat{\theta}|W_t))$.
- **Model Selection:** It compares the fitted models using an information criterion, typically the Akaike Information Criterion (AIC):

$$AIC = -2 \ln(\hat{L}) + 2k$$

where k is the total number of parameters ($p + q + 1$) and \hat{L} is the maximized log-likelihood. The model with the lowest AIC score is selected as the best.

Step 2: Forecasting and Trading Strategy

Using the best-fit ARIMA model from Step 1, we generate one-step-ahead forecasts (\hat{Y}_{t+1}) in a rolling walk-forward fashion. These log-price forecasts are converted back to price units: $\hat{P}_{t+1} = \exp(\hat{Y}_{t+1})$.

- **Entry Signal (Long):** If $\hat{P}_{t+1} > P_t \times (1 + k)$ (forecast predicts significant rise) and no position is held, enter a long position (Buy).
- **Entry Signal (Short):** If $\hat{P}_{t+1} < P_t \times (1 - k)$ (forecast predicts significant fall) and no position is held, enter a short position (Sell Short).
- **Exit Signal (from Long):** If a long position is held AND $\hat{P}_{t+1} < P_t \times (1 - k)$ (forecast signal reverses to 'Sell'), exit the long position (Sell).
- **Exit Signal (from Short):** If a short position is held AND $\hat{P}_{t+1} > P_t \times (1 + k)$ (forecast signal reverses to 'Buy'), exit the short position (Buy to Cover).

Here, P_t is the current price and k is a small threshold to cover transaction costs and filter noise.

Purpose of This Phase: This process establishes a baseline: what kind of forecast accuracy we can achieve without considering the relationships between different stocks. It serves as a reference to compare with pairs trading methods, and helps validate our implementation and modelling choices as we proceed to more complex approaches.

Phase 3: Static Hedge Ratio Estimation

This phase acts as a baseline approach to pairs trading by assuming a constant relationship between two stocks, rather than allowing it to change dynamically. The key is to estimate a fixed hedge ratio using simple linear regression and generate trading signals from the spread.

Step 1: Calculating the Hedge Ratio

We model the log price series of two stocks as a linear relationship:

$$Y_t^{(y)} = \hat{\alpha} + \hat{\eta} Y_t^{(x)} + \hat{\varepsilon}_t$$

which is essentially same as what we have done previously for cointegration test.

The spread, which we use for trading, is simply the deviation from the predicted value:

$$\text{Spread}_t = Y_t^{(y)} - (\hat{\alpha} + \hat{\eta} Y_t^{(x)})$$

Step 2: Z-Score Normalization

To make the spread comparable across time and assets, we standardize it into a z-score, which measures how far the current spread is from its rolling average, in units of standard deviation:

$$Z_t = \frac{\text{Spread}_t - \mu_T(\text{Spread})}{\sigma_T(\text{Spread})}$$

where μ_T and σ_T are the rolling mean and standard deviation calculated over a look-back window of length T .

This z-score allows us to set simple entry and exit rules that are robust against scale changes and market shifts.

Step 3: Trading Signal Logic

Trading rules are based on thresholds. For example:

- **Enter Long:** If $Z_t < -k$ (spread is unusually low): Buy Stock Y and Short Sell Stock X (scaled by η). Bet on the spread increasing.
- **Enter Short:** If $Z_t > +k$ (spread is unusually high): Short Sell Stock Y and Buy Stock X (scaled by η). Bet on the spread decreasing.
- **Exit:** If a position is held AND $|Z_t| < k_{exit}$ (spread has reverted to mean): Exit the position as mentioned before.

Purpose and Limitations: This static approach gives us the simplest possible implementation of a mean-reversion strategy. While effective in stable environments, it may miss opportunities or fail in changing markets where relationships between stocks evolve over time.

Phase 4: Dynamic Hedge Ratio Estimation

Kalman Filters

This phase introduces the Kalman Filter to estimate the hedge ratio dynamically, allowing it to change over time as market conditions evolve. Unlike a static value from ordinary regression, the Kalman Filter continuously updates the hedge ratio as new data comes in. The motivation is that the relationship between two stocks (the hedge ratio) may not remain constant due to shifts in economic regimes, company fundamentals, or other factors.

Step 1: State-space Representation

We define a vector representing the hidden "state" of our system:

$$\beta_t = \begin{bmatrix} \alpha_t \\ \eta_t \end{bmatrix}$$

where α_t is the intercept and η_t is the time-varying hedge ratio at time t . The observed log price of stock y at time t is modeled as:

$$Y_t^{(y)} = H_t \beta_t + v_t$$

Here, $H_t = [1 \quad Y_t^{(x)}]$ is a vector incorporating a constant (1) and the current log price of stock x . v_t is random measurement noise. The state itself evolves over time as a simple random walk:

$$\beta_t = \beta_{t-1} + w_t$$

w_t is process noise, representing small, unpredictable changes in the hedge ratio.

Step 2: Kalman Filter Recursive Steps

The Kalman Filter combines prediction (based on previous state estimates) and correction (using new evidence from observations).

- Prediction step: Estimate the next state and its uncertainty.

$$\hat{\beta}_{t|t-1} = \hat{\beta}_{t-1|t-1}$$

$$P_{t|t-1} = P_{t-1|t-1} + Q$$

where Q is the process noise covariance, and P is the error covariance matrix.

- Update step: Adjust the predicted state in light of the new observation.

$$e_t = Y_t^{(y)} - H_t \hat{\beta}_{t|t-1}$$

This is the innovation, or measurement residual (difference between what we expect and what we observe).

$$S_t = H_t P_{t|t-1} H_t^T + R$$

S_t is the total uncertainty, combining predicted uncertainty and measurement noise R .

$$K_t = P_{t|t-1} H_t^T S_t^{-1}$$

K_t is the Kalman Gain, determining how much trust to put in the new observation versus the prediction.

$$\hat{\beta}_{t|t} = \hat{\beta}_{t|t-1} + K_t e_t$$

$$P_{t|t} = (I - K_t H_t) P_{t|t-1}$$

This recursive process continues, updating our best estimate of the hedge ratio and its uncertainty with each new time point.

Step 3: Dynamic Spread and Z-score Signal Generation

Once the dynamic hedge ratio $\hat{\eta}_t$ is estimated, we re-calculate the spread:

$$\hat{\varepsilon}_t = Y_t^{(y)} - \hat{\alpha}_t - \hat{\eta}_t Y_t^{(x)}$$

To filter out normal fluctuations and focus on statistically significant deviations, we standardize the spread using a z-score:

$$Z_t = \frac{\hat{\varepsilon}_t - \mu(\hat{\varepsilon})}{\sigma(\hat{\varepsilon})}$$

Here, $\mu(\hat{\varepsilon})$ and $\sigma(\hat{\varepsilon})$ are the rolling mean and standard deviation of the spread.

Signals are generated when the z-score crosses pre-defined thresholds. For example, we enter a trade when the z-score is unusually high or low, expecting it to revert toward zero, as done previously in static hedge ratio case.

Intuitive Summary: The Kalman Filter offers a mathematically principled way to adapt the hedge ratio as the joint behavior of the stocks changes over time.

Phase 5: Multivariate Modeling

Vector Error Correction Model – VECM

This phase extends pairs trading to a multivariate setup, where we can simultaneously model and exploit relationships between more than two stocks. The Vector Error Correction Model (VECM) generalizes cointegration and allows us to capture both short-term fluctuations and long-term equilibrium in a group of non-stationary series.

Step 1: Mathematical Structure and Intuition

For N stocks, we arrange their log-prices into a vector:

$$\mathbf{y}_t = [y_t^{(1)}, y_t^{(2)}, \dots, y_t^{(N)}]^T$$

Each $y_t^{(i)}$ is typically non-stationary (e.g., their mean changes over time), but a particular linear combination of them may be stationary (mean-reverting). VECM focuses on these combinations to find robust trading signals.

First, we take differences (log-returns):

$$\mathbf{r}_t = \Delta \mathbf{y}_t = \mathbf{y}_t - \mathbf{y}_{t-1}$$

VECM is specified as:

$$\mathbf{r}_t = \Pi \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Gamma_i \mathbf{r}_{t-i} + \mathbf{w}_t$$

Here:

- Π ($N \times N$ matrix) captures the long-term equilibrium relationships,
- Γ_i ($N \times N$ matrix) reflects short-term interactions,
- \mathbf{w}_t ($N \times 1$ vector) is random noise,
- p is the lag order (how many previous periods we look back on).

Step 2: Understanding Cointegration and Error Correction

The matrix Π can be decomposed if there are r cointegrating relationships ($r < N$):

$$\Pi = \alpha \beta^T$$

where:

- β ($N \times r$ matrix) gives the cointegrating vectors (linear combinations that are stationary),
- α ($N \times r$ matrix) tells us how quickly the prices adjust toward equilibrium.

For trading, the key signal is the Error Correction Term (ECT):

$$ECT_{t-1} = \beta^T \mathbf{y}_{t-1}$$

When ECT_{t-1} is far from zero, it signals the system is away from equilibrium and is likely to revert, creating a trading opportunity.

Step 3: Estimation and Trading Signal Generation

We typically estimate the parameters (how many cointegrating relationships there are, and what they look like) using procedures like Johansen's test. We will use `statsmodels.tsa.vector_ar.vecm` in Python to implement the VECM. The practical steps are mentioned below:-

1. **Input:** price series for N assets, lookback window W , lag order candidate set for p , thresholds (z_{enter} , z_{exit}).
2. Compute log-prices (y_t) over the lookback window.
3. Choose lag (p) via information criteria (AIC/BIC) fitted on a VAR(p) in *levels*.
4. Run Johansen test on y_{t-1} levels to obtain cointegration rank r and estimate β . Johansen test uses eigenvectors and eigenvalues to estimate β . Also obtain α and preliminary Γ_i if available.
5. For each time t (or on each re-estimation step):
 - (a) compute $ECT * t = \beta^T y_t$ ($r \times 1$ vector)
 - (b) compute standardized scores $z_t = (ECT_t - \mu_{ECT}) / \sigma_{ECT}$, where μ, σ are estimated with a long or rolling window
 - (c) if any $|z_{t,i}| > z_{\text{enter}}$:
 - determine trade sign from the sign of $z_{t,i}$ (positive → short the β -defined basket; negative → long the basket)
 - compute hedged positions using the corresponding column(s) of β (weights $w \propto -\beta$ for that ECT), normalize (e.g., sum abs weights = 1) and scale by risk budget (target volatility)
 - (d) if all $|z_{t,i}| < z_{\text{exit}}$ then close corresponding positions

Summary and Advantages: VECM is useful because it doesn't just consider isolated pairs — it can include many stocks, finding robust combinations that revert together. It handles both fast short-term moves and slow drift back to economic equilibrium, making it especially powerful for multi-dimensional statistical arbitrage.

Performance Evaluation and Backtesting

After designing trading strategies using cointegration, ARIMA, static hedge ratio, Kalman filter, and VECM, it is necessary to evaluate their performance. This is done through backtesting and by calculating several standard financial metrics.

Step 1: Backtesting Approach

Backtesting means simulating how the strategy would have performed in the past, using historical data. We operate in a rolling or walk-forward fashion: at each step, the model is trained on recent data, then used to predict and generate signals on the next unseen period.

Step 2: Key Financial Metrics

Several measures help us objectively evaluate and compare strategies:

- **Net Profit and Loss (Net PnL):** The total profit earned after deducting all transaction costs and fees. NetPL is the real amount that matters to a trader or investor.

$$\text{Net PnL} = \text{Gross PnL} - \sum_{i=1}^{N_{\text{trades}}} \text{TC}_{\text{total},i}$$

- **Compound Annual Growth Rate (CAGR):** This tells us the yearly growth rate of the portfolio, assuming profits are reinvested. The formula is:

$$CAGR = \left(\frac{\text{End Value}}{\text{Start Value}} \right)^{1/\text{Years}} - 1$$

- **Sharpe Ratio (SR):** This is a measure of risk-adjusted return. It shows how much extra return is received per unit risk taken. The formula is:

$$SR = \frac{ER_p - R_f}{\sigma_p} \sqrt{T}$$

where ER_p is expected portfolio return, R_f is risk-free return, σ_p is standard deviation of returns, and T is the number of periods in a year.

- **Maximum Drawdown (MDD):** This quantifies the largest drop from a peak to the trough in the portfolio value:

$$MDD = \max_{0 \leq t < T} \frac{PV_{\text{peak}} - PV_t}{PV_{\text{peak}}}$$

where PV_{peak} is the highest value reached and PV_t is value at time t .

- **Transaction Costs:** Real strategies must account for costs per trade (broker fees, taxes like STT, slippage). We model total transaction costs using a combined approach:

$$TC_{\text{total}} = (C_{\text{fixed_pct}} + \alpha) \times \text{Transaction Value}$$

where $C_{\text{fixed_pct}}$ represents fixed percentage costs (brokerage, statutory fees $\approx 0.05\%$) applied to turnover (Transaction Value), and α represents estimated variable costs (slippage/impact) proportional to the Transaction Value.

Step 3: How Metrics Guide Development

These metrics allow us to answer key questions:

- Did the mean-reversion model actually translate into profits?
- Was the profit stable and risk-adjusted, or volatile and risky?
- How much of the returns were eaten up by trading costs?
- Did the strategy survive market downturns (MDD)?

By comparing results for all models on these metrics, we objectively determine which approach offers the best combination of returns, risk management, and robustness.