

Project Brief: Statistical Arbitrage in the U.S. Equities Market

The Hook

Statistical arbitrage is the absolute bedrock of market-neutral quantitative hedge funds. This classic paper formalizes the pairs-trading process by using Principal Component Analysis (PCA) to strip away the broader market's systematic risk, mathematically isolating the idiosyncratic, mean-reverting "noise" of individual stocks. By building this, an independent researcher learns how real-world quants extract systematic alpha from pricing inefficiencies without taking on raw directional market exposure.

Definition of Done (Project Scope)

For an independent/extra project, a fully completed repository should include:

- **A cleanly structured codebase:** An object-oriented Python repository implementing the end-to-end statistical arbitrage pipeline.
- **The Factor Engine:** A module that computes the correlation matrix of asset returns and uses PCA to construct empirical eigenportfolios (representing systematic risk factors).
- **Signal Generation:** Implementation of the Ornstein-Uhlenbeck (OU) stochastic process to model the residuals. The code must estimate the mean-reversion speed (κ) and volatility (σ) of the residual process $dX_t = \kappa(\theta - X_t)dt + \sigma dW_t$ to generate the dimensionless ss -score.
- **The Backtester:** A vectorized historical backtesting framework that simulates taking long/short positions when the ss -score crosses predefined thresholds (e.g., entering trades at ± 1.25 and closing at mean reversion).
- **Performance Metrics & Visualization:** A cumulative PnL chart comparing the PCA-based strategy against a benchmark index, alongside the calculation of the strategy's Sharpe ratio after assumed transaction costs.
- **Stretch Goal:** Deploy the mathematical backend into a full-stack quantitative trading web app to track live ss -scores and generate real-time long/short signals across the S&P 500.

9-Week Milestone Timeline

- **Week 1 (Feb 20 – Feb 26): Literature Review & PCA Fundamentals**
 - Read the paper, focusing on the mathematical distinction between systematic market factors (extracted via PCA or ETFs) and the idiosyncratic residuals.
- **Week 2 (Feb 27 – Mar 5): Data Sourcing & Cleaning**
 - Set up the GitHub repository.
 - Ingest historical daily adjusted close prices for a highly liquid universe (e.g., S&P 500 components). Handle missing data and calculate standardized daily returns.
- **Week 3 (Mar 6 – Mar 12): Extracting Eigenportfolios**
 - Code the PCA engine. Compute the correlation matrix of the return panel, extract the

eigenvalues and eigenvectors, and construct the first N eigenportfolios that explain the majority of the market's variance.

- **Week 4 (Mar 13 – Mar 19): Modeling the Residuals**
 - Regress individual stock returns against the systematic eigenportfolios to extract the residual returns (the idiosyncratic noise).
- **Week 5 (Mar 20 – Mar 26): The Ornstein-Uhlenbeck Process**
 - Code the OU calibration logic. Formulate the autoregressive model (AR(1)) to calculate the mean reversion speed, equilibrium level, and variance of the residuals to generate the daily S_t -score for each stock.
- **Week 6 (Mar 27 – Apr 2): Developing the Trading Rules**
 - Build the trading logic that scales into short positions when the S_t -score is highly positive (overbought) and long positions when highly negative (oversold).
- **Week 7 (Apr 3 – Apr 9): Vectorized Backtesting**
 - Run the strategy over an out-of-sample historical period.
 - Add realistic constraints, such as trading costs (e.g., 5 basis points per transaction) and slippage, to see if the alpha survives real-world friction.
- **Week 8 (Apr 10 – Apr 16): Competition Readiness**
 - Refactor the data pipeline to ensure it can ingest new data, recalculate the PCA, and output fresh S_t -scores efficiently. This provides a highly viable, battle-tested trading algorithm perfectly timed for the algorithmic trading competition on April 18th.
- **Week 9 (Apr 17 – Apr 23): Documentation & Review**
 - Finalize the GitHub README. Since this is an independent portfolio project, ensure the mathematical translation from the paper to the code is heavily commented to impress quantitative researchers during interviews.

The Toolkit

- **Data:** The yfinance API to pull historical stock data, or a free tier of alpaca-trade-api for slightly more robust historical price ingestion.
- **Libraries:** * scikit-learn (sklearn.decomposition.PCA) to handle the heavy lifting of the Principal Component Analysis without reinventing the wheel.
 - statsmodels for running the linear regressions necessary to extract the residuals and calibrate the AR(1) mean-reversion process.
 - numpy and pandas for vectorized signal generation.
 - matplotlib for plotting the S_t -score thresholds and cumulative PnL curves.