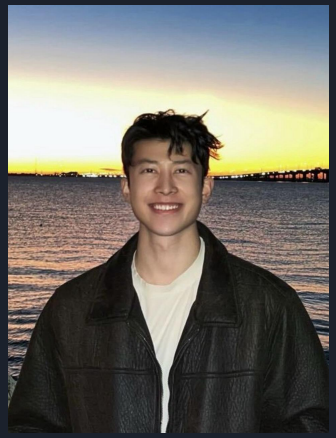
Statistical Arbitrage via Market Segmentation

Gabriel Nelson, Harlan Chen, David Hidary, and Ilham Rifqi

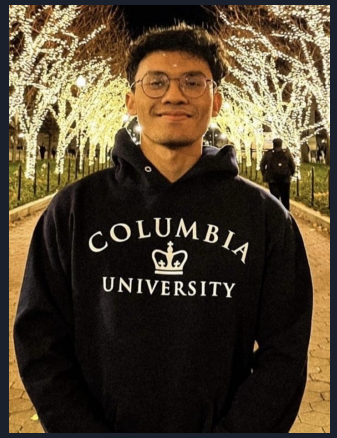
Stat Arb.ipynb

About us

Gabriel Nelson
Applied Math and Statistics
Focus on Probability & Market Research



Harlan Chen
Applied Math and Philosophy
Focus on FX Trading & Credit Research



Ilham Rifqi
Applied Mathematics and Economics
Focus on Monetary Policy & Portfolio Management

David Hidary
Applied Mathematics
Focus on Deep Learning & Physics

Primary References

Bank, Dor, Noam Koenigstein, and Raja Giryes. 2023. “Autoencoders.” Springer EBooks, January, 353–74. https://doi.org/10.1007/978-3-031-24628-9_16.

Plaut, E. (2018). From principal subspaces to principal components with linear autoencoders. arXiv preprint arXiv:1804.10253. https://arxiv.org/abs/1804.1025

Krauss, Christopher. 2016. “Statistical Arbitrage Pairs Trading Strategies: Review and Outlook.” Journal of Economic Surveys 31 (2): 513–45. https://doi.org/10.1111/joes.12153.

Lee, Tae-Hwy, and Ekaterina Seregina. 2020. “Optimal Portfolio Using Factor Graphical Lasso.” ArXiv.org. 2020. https://arxiv.org/abs/2011.00435.

Other References

Elliott, Robert J., John Van Der Hoek *, and William P. Malcolm. 2005. “Pairs Trading.” Quantitative Finance 5 (3): 271–76. https://doi.org/10.1080/14697680500149370.

Gatev, Evan, William N. Goetzmann, and K. Geert Rouwenhorst. 2006. “Pairs Trading: Performance of a Relative-Value Arbitrage Rule.” Review of Financial Studies 19 (3): 797–827. https://doi.org/10.1093/rfs/hhj020.

Ledoit, Olivier, and Michael Wolf. 2004. “A Well-Conditioned Estimator for Large-Dimensional Covariance Matrices.” Journal of Multivariate Analysis 88 (2): 365–411. https://doi.org/10.1016/s0047-259x(03)00096-4.

Zhang, Di. 2025. “Efficient Triangular Arbitrage Detection via Graph Neural Networks.” ArXiv (Cornell University), February. https://doi.org/10.48550/arxiv.2502.03194.

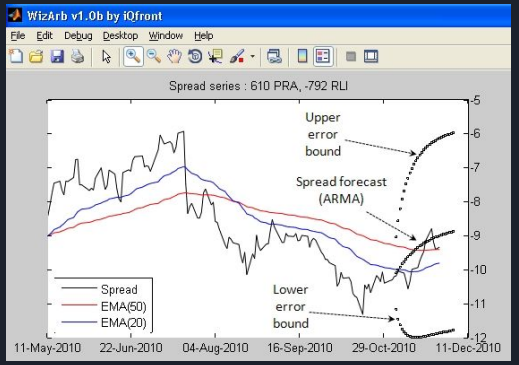
Zhao, Ziping, Rui Zhou, and Daniel P. Palomar. 2019. “Optimal Mean-Reverting Portfolio with Leverage Constraint for Statistical Arbitrage in Finance.” IEEE Transactions on Signal Processing 67 (7): 1681–95. https://doi.org/10.1109/tsp.2019.2893862.

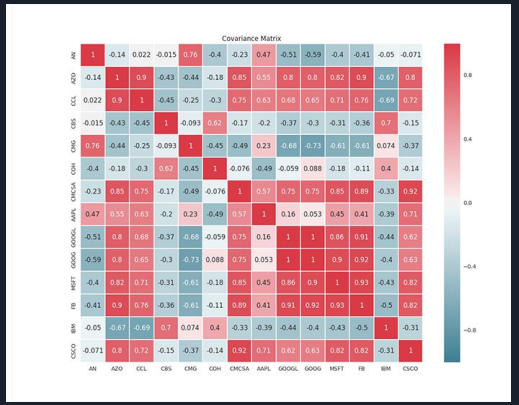
Overview

Background & Foundations
- Statistical Arbitrage (Stat Arb): definition & history
- Introduction to Pairs Trading
- Optimizations for Pairs Trading models

Multi-Asset Extensions and Market Segmentation
- Stat Arb in higher dimensions
- Covariance matrix estimation
- Ledoit–Wolf and other shrinkage methods
- Graphical Lasso Regularization method
- Principal Component Analysis
- Autoencoders and Variational Methods

Future Outlook
- The development of the field
- Our plans after graduation





(Credit: Medium)

Background & Foundations

Columbia and the birth of Stat Arb

Gerry Bamberger, CS graduate in 1983

- Started out just supporting traders at Morgan Stanley
- Coca-Cola vs Pepsi trade
- $500k to $30 million - a test program
- The birth of pair trading and MS Quant Group

Left Columbia faculty post in 1986, David E. Shaw joined the very same MS quant group under Nunzio Tartaglia. In 1988, Shaw left to found D. E. Shaw & Co.

David E. Shaw

What is Statistical Arbitrage?

- Uses statistical models to identify and exploit short-term mispricings
  - Market-direction independent (beta-neutral), often sector-neutral through diversification
  - Efficient Market Hypothesis?
    - Jean-Philippe Bouchaud: “The whole bull run is because of an influx of money” (Financial Times)
  - Guided by the Law of One Price: identical assets should trade at the same value
- Goal: achieve consistent, small, repeatable profits with controlled risk

Why is it Important

- Ideally provides market neutral returns with low risk
- Largely self-funding - returns from shorts are used to buy the long position
- Advances in Deep Learning and AI - new opportunities
  - Also alternative data and optimizations
- Dominated by computing - HFT a must
- Why do we care
  - Not everything can be systematic

Pairs Trading

- 2 stocks (a pair) historically move together
- When they diverge there exists arbitrage opportunities
  - Mean reversion
  - Long on the underperforming stock
  - Short the other
- Exit the position when they revert

Pairs Formation Optimization

Simple Model: Stocks Across Sectors

- Calculate daily returns across equities
- Pairs formation period ~ 1 year
- Find top pairs with high covariance
- Begin trading period
  - Take positions during large deviations
  - Close positions upon mean reversion
- Close all remaining positions at the end of the trading period
- Calculate excess returns
  - Benchmark 0.81% per month (Gatev et al. 2006)

$$\mathbf{COV(X,\ Y)} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - E(X))(y_i - E(Y))$$

$$\mathbf{COR(X,\ Y)} = \frac{\mathbf{COV(X,\ Y)}}{\sqrt{\mathbf{VAR(X)VAR(Y)}}}$$

$$\Sigma = \begin{bmatrix} \sigma_{00}^2 & \sigma_{01}^2 & \sigma_{02}^2 \\ \sigma_{10}^2 & \sigma_{11}^2 & \sigma_{12}^2 \\ \sigma_{20}^2 & \sigma_{21}^2 & \sigma_{22}^2 \end{bmatrix}$$

Covariance Matrix Estimation

$Input$: $Price\ matrix\ P \in R^{\{T*N\}}$

(1) Compute returns

$$for\ each\ asset\ i = 1 \ldots N:\ for\ t = 2 \ldots T:\ R[t-1,\ i] \leftarrow \left(\frac{P[t,\ i]}{P[t-1,\ i]}\right) - 1]$$

(2) Demean each column

$$for\ i = 1..N:\ u_i \leftarrow average\ (R[:,\ i])$$

$$for\ t = 1 \ldots (T-1): R_{demean}[t,\ i] \leftarrow R[t,\ i] - u_i$$

(3) *Build covariance matrix*

$$\Sigma \leftarrow \frac{1}{(T-1)} * (R^T_{demean} * R_{demean})$$

$Output$:

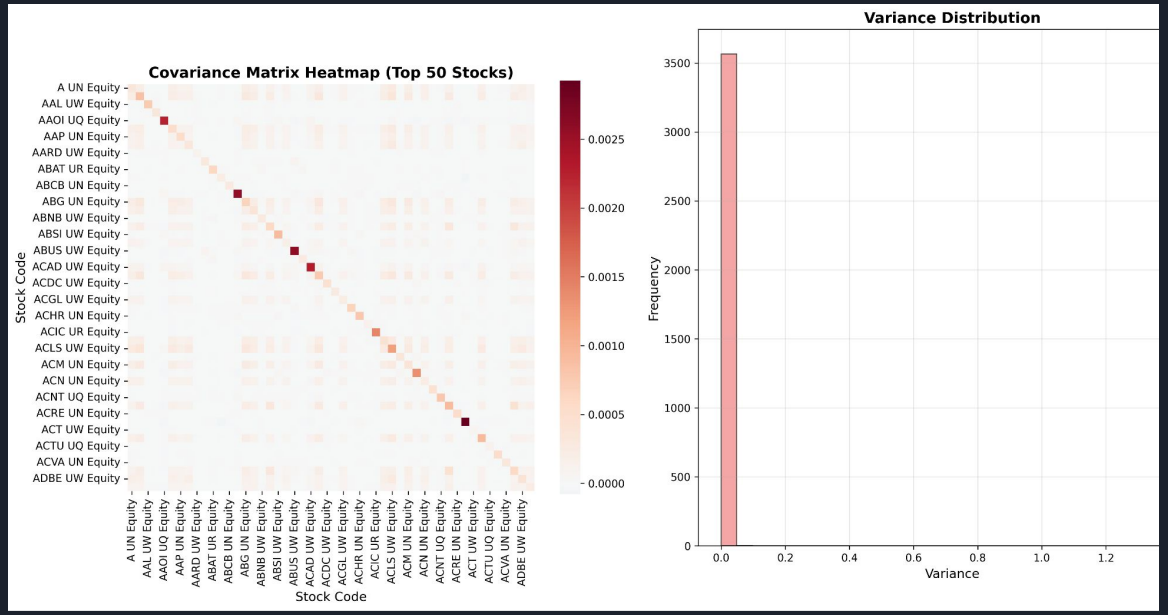
$$\Sigma \in R^{\{N*N\}}$$

$$Diagonal\ \Sigma[i,i] = variance(i)$$

$$Off - diagonal\ \Sigma[i,j] = covariance(i,j)$$

Problem When Equities Exceed Sample Size

- We analyzed 513 stocks over the past trading year (252 days).
- Heatmap: most covariances are near zero, and a few are significant.
- Variance Distribution: The majority of variances collapse near zero; a few outliers dominate.
- This shows the Matrix is ill-conditioned – unstable and close to singular.



Ledoit–Wolf Shrinkage Estimation

What is Ledoit–Wolf?

- A shrinkage method that weights the sample covariance matrix toward a structured target (usually identity).
- Balances sample covariance with prior structure to reduce estimation error.

Why Ledoit–Wolf matters in Finance?

- Stabilizes covariance estimates when the amount of parameters are near the sample size.
- Prevents singular or ill-conditioned covariance matrices.
- Reduces noise in eigenvalues, improving risk modeling.
- Enables more reliable portfolio optimization and factor analysis.

Ledoit–Wolf Shrinkage Estimation

$$Input:\ Return\ matrix\ R \in R^{\{T*N\}}$$

$$(1)\ Empirical\ Covariance$$

$$u_i \leftarrow average\ (R[:, i])$$

$$R\ [:, i] \leftarrow R[:, i] - u_i$$

$$S \leftarrow (\frac{1}{T-1}) * (R^T * R)$$

$$(2)\ Shrinkage\ Target$$

$$\bar{v} \leftarrow (\frac{1}{N}) * trace(S)\ \text{//avg variance}$$

$$Tgt \leftarrow \bar{v} * I_n\ \text{//independent scaled Identity}$$

$$(3)\ Optimal\ Shrinkage\ Intensity\ (Closed\ form)$$

$$\delta^* \leftarrow (\frac{Var[||R||^2]}{Var_{Target}}) = \frac{\Sigma\, Var(S_{ij})}{\Sigma(S_{ij} - Tgt_{ij})^2}$$

$$(4) Ledoit - Wolf\ Covariance$$

$$\widehat{\Sigma} \leftarrow (1 - \delta^*) * S + \delta^* * Tgt$$

$$(5)\ Positive\ Definite\ \&\ Conditional\ Guard$$

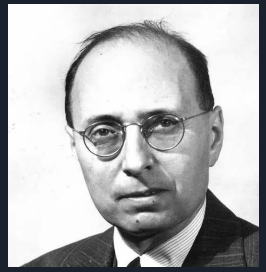
$$if\ \lambda_{min}(\widehat{\Sigma}) \leq 0\ or\ cond(\widehat{\Sigma}) \geq 1$$

$$\epsilon \leftarrow \eta * \frac{trace(\widehat{\Sigma})}{N},\ \widehat{\Sigma} \leftarrow \widehat{\Sigma} + \epsilon * I_n$$

$$Output:\ shrinkage\ covariance$$

$$\Sigma \in R^{\{N*N\}}$$

One Step Further to RMT



Eugene Wigner, 1950s

Jean-Philippe, CFM, 2018

What is RMT (random matrix theory)?

- Initially used to model the complex energy spectra of heavy atomic nuclei.
- Identify and clean noisy small eigenvalues in the covariance matrix
- Removes components that look like random noise rather than true data structure.
- Especially when assets are highly correlated or data is limited.
- Prevents singular / unstable matrices

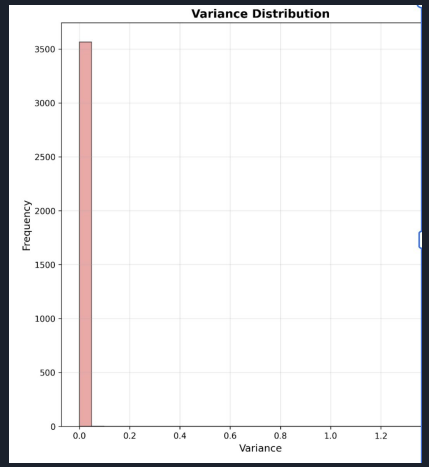
Like tuning a radio, RMT removes “static” frequencies so only real signals come through.

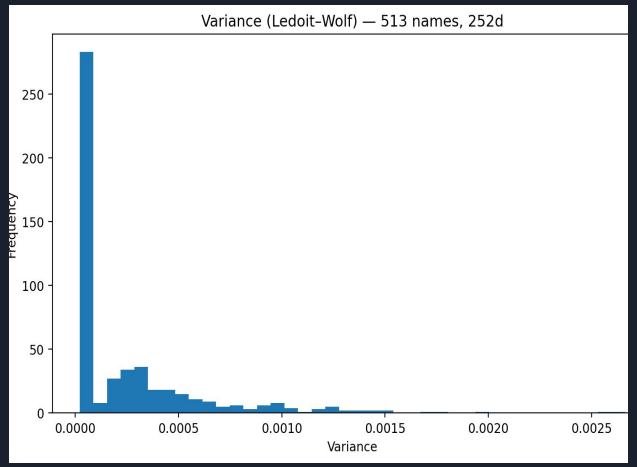
The simple rule here is comparing eigenvalue spectrum against the Marchenko − Pastur distribution

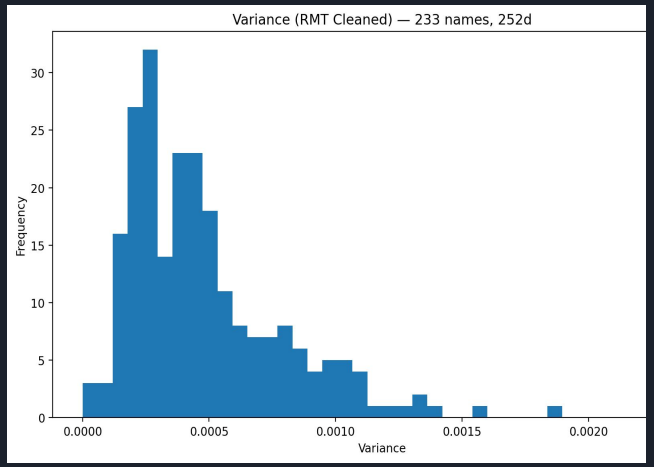
$$\rho = \frac{Q}{2\pi\sigma^2} * \frac{\sqrt{(\lambda_+ - \lambda)(\lambda - \lambda_-)}}{\lambda}, \text{ where bounds } \lambda_\pm = \sigma^2\left(1 \pm \sqrt{\frac{1}{Q}}\right)^2, \; Q = \frac{T}{N}$$

If $\lambda_i \leq \lambda_+$: *treat as noise, shrink/replace, otherwise keep as signal*

Variance Distribution: Sample, LW, and RMT





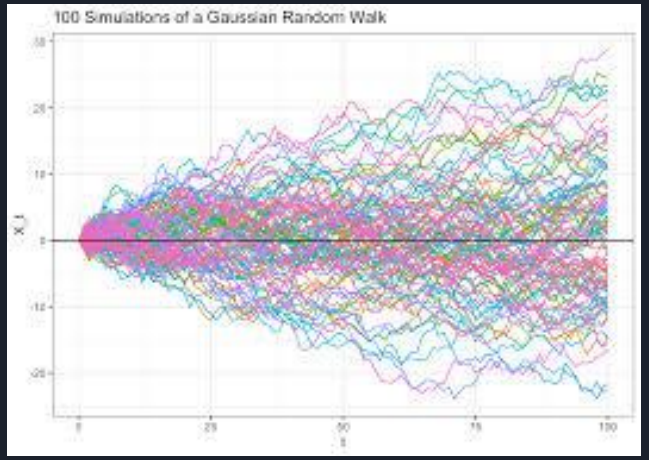


Further Pairs Trading Methods

- Simple model shown earlier is distance approach
- Measures squared euclidean distance and takes positions when spread is larger than two standard deviations
- Here the optimal model has no spread and therefore no profit

Time Series Approach (Ornstein–Uhlenbeck)

- Spread of two equities can be modeled mean-reverting Gaussian Markov chain
- Very similar to the Ornstein–Uhlenbeck process
- Very useful - want large spread and quick mean reversion



(Credit: Bookdown)

Time Series

$$x_{k+1} - x_k = (a - bx_k)\,\tau + \sigma\sqrt{\tau}\,\varepsilon_{k+1}$$

- x_k: latent state variable at time step k
  (the "true" spread, before measurement noise)
- a: long-run pull constant
- b: mean reversion strength (the larger b, the faster reversion)
- τ: time step size (e.g., 1 day)
- $\sigma\sqrt{\tau}\,\varepsilon_{k+1}$: random shock, with variance scaling as $\sigma^2\tau$

Ornstein–Uhlenbeck

$$dS_t = \kappa(\mu - S_t)\,dt + \sigma\,dW_t$$

- S_t: the spread (state variable) at time t
- $\kappa > 0$: speed of mean reversion. Higher κ means faster pull back toward μ
- μ: long-run equilibrium level (mean) of the process
- $\sigma > 0$: volatility parameter, controls the size of random fluctuations
- W_t: standard Brownian motion (continuous-time random noise)

Quick Test: Mean Reversion AR(1)

AR(1) Model with OLS

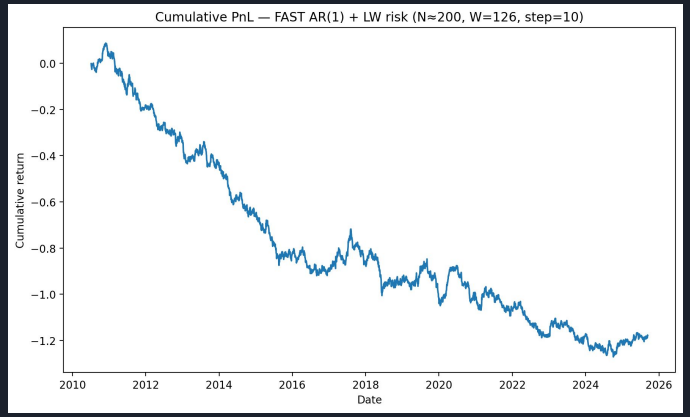
$$r_t = \alpha + \phi r_{t-1} + \varepsilon_t$$

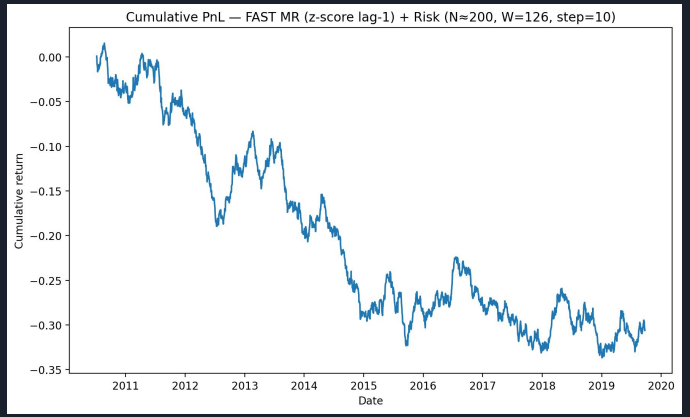
- OLS estimates parameters by minimizing squared errors.
- **ϕ (slope):** measures how strongly today depends on yesterday.
- **α (intercept):** baseline average level.

Interpretation:

- $|\phi| < 1$ → mean reversion (pulls back to average)
- $\phi \approx 1$ → random walk (no reversion)
- $\phi > 1$ → explosive, unstable

Mean Reversion AR(1) Quick Test Results





- Mean reversion techniques with cleaned covariance matrices.
- The cumulative P&L shows a downward trend, failed to generate alpha.
- Even with slight recovery in the “Fast MR” version, the overall performance remains negative.
- We wonder if stable covariance alone is not enough, a better market segmentation is the solution we explore to separate signal from noise…

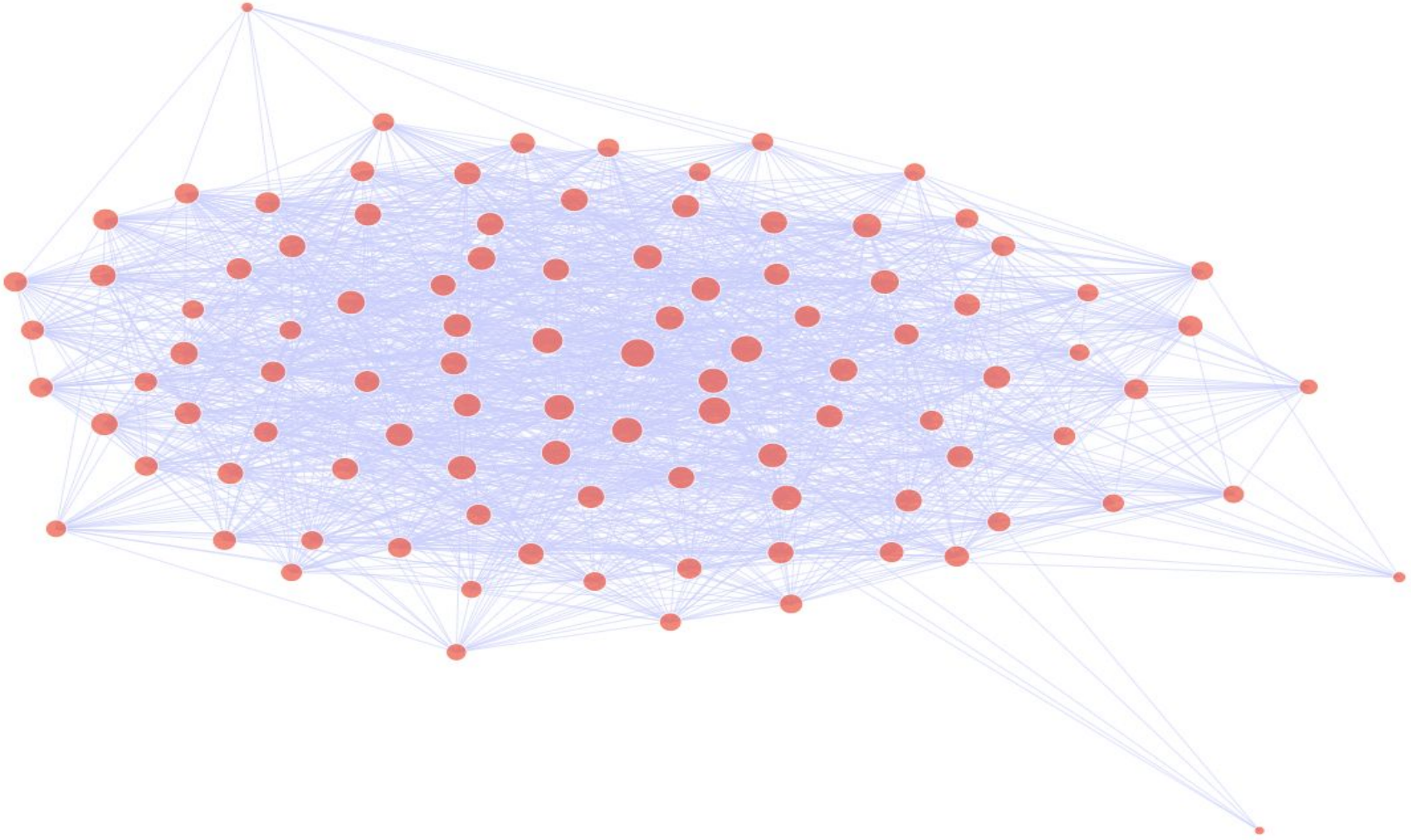
From Formation to Trading Period

Graphical Lasso

- Cannot always simply take highest correlated stocks
  - There could be lurking variables
  - Everything has some covariance with each other
- Use the precision matrix (invert covariance matrix) to find direct relationships.
  - Even with Ledoit Wolf matrix is noisy
  - Add L1 penalty term
  - Used in OLS

$$\hat{\beta} = \arg\min_{\beta} \|y - X\beta\|^2 + \lambda\|\beta\|_1$$

- Graphical lasso starts with connected graph
  - Uses L1 penalty to create a sparse network



GLASSO Pseudocode

Graphical Lasso

Input: Centered data $X \in \mathbb{R}^{T \times N}$, sparsity $\lambda > 0$, tolerances $\varepsilon_{\text{lasso}}, \varepsilon_{\text{outer}}$
Output: Sparse precision $\widehat{\Theta} \succ 0$, partial correlations P, graph $\mathcal{G}$
Step A: Likelihood Function

Assume $X_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mu, \Sigma)$ with precision $\Theta = \Sigma^{-1} \succ 0$ and

$$p(x \mid \mu, \Theta) = (2\pi)^{-N/2} (\det \Sigma)^{-1/2} \exp\Big(-\tfrac{1}{2}(x-\mu)^\top \Theta (x-\mu) \Big).$$

Likelihood and log-likelihood:

$$L(\mu, \Theta) = \prod_{t=1}^{T} p(X_t \mid \mu, \Theta), \qquad \log L(\mu, \Theta) = \sum_{t=1}^{T} \log p(X_t \mid \mu, \Theta).$$

Expand the sum:

$$\log L(\mu, \Theta) = -\tfrac{TN}{2} \log(2\pi) + \tfrac{T}{2} \log\det \Theta - \tfrac{1}{2} \sum_{t=1}^{T} (X_t - \mu)^\top \Theta (X_t - \mu).$$

GLASSO Pseudocode

Step B: Differentiate w.r.t. μ (eliminate the mean)

Use the identity

$$\nabla_\mu \, (X_t - \mu)^\top \Theta (X_t - \mu) = -2\,\Theta\,(X_t - \mu).$$

Therefore,

$$\nabla_\mu \log L(\mu, \Theta) = -\tfrac{1}{2} \sum_{t=1}^{T} \big(-2\,\Theta\,(X_t - \mu) \big) = \Theta \sum_{t=1}^{T} (X_t - \mu).$$

Set to zero (stationarity) and solve:

$$\Theta \sum_{t=1}^{T} (X_t - \mu) = 0 \implies \sum_{t=1}^{T} (X_t - \mu) = 0 \implies \widehat{\mu} = \bar{X} := \tfrac{1}{T} \sum_{t=1}^{T} X_t.$$

Center data: $\tilde{X}_t = X_t - \bar{X}$. Then

$$\sum_{t=1}^{T} \tilde{X}_t^\top \Theta \tilde{X}_t = \sum_{t=1}^{T} \operatorname{tr}\big(\Theta \tilde{X}_t \tilde{X}_t^\top\big) = \operatorname{tr}\Big(\Theta \sum_{t=1}^{T} \tilde{X}_t \tilde{X}_t^\top\Big) = T \operatorname{tr}(\Theta S),$$

with $S = \frac{1}{T} \sum_{t=1}^{T} \tilde{X}_t \tilde{X}_t^\top$. Hence

$$\log L(\hat{\mu}, \Theta) = -\tfrac{TN}{2} \log(2\pi) + \tfrac{T}{2} \log\det \Theta - \tfrac{T}{2} \operatorname{tr}(S\Theta).$$

GLASSO Pseudocode

Step C: Average negative log-likelihood and its gradient in Θ

Drop constants and scale:

$$\mathcal{L}(\Theta) \;=\; -\log\det\Theta + \operatorname{tr}(S\Theta).$$

Matrix-calculus identities:

$$\nabla_\Theta(-\log\det\Theta) = -\Theta^{-1}, \qquad \nabla_\Theta \operatorname{tr}(S\Theta) = S.$$

Therefore the (unpenalized) score is

$$\nabla_\Theta \mathcal{L}(\Theta) \;=\; -\Theta^{-1} + S.$$

Stationarity (MLE, $\lambda = 0$) gives $\Theta^{-1} = S \Rightarrow \widehat{\Sigma} = S$ (when S is invertible).

Step D: Glasso objective and KKT with subgradient

Penalize off-diagonals for sparsity:

$$\widehat{\Theta} = \arg\min_{\Theta \succ 0} \Big\{ -\log\det\Theta + \operatorname{tr}(S\Theta) + \lambda\|\Theta\|_{1,\text{off}} \Big\}, \quad \|\Theta\|_{1,\text{off}} = \sum_{i \neq j} |\Theta_{ij}|.$$

Subgradient Z for off-diagonals:

$$Z_{ij} \in \begin{cases} \{\operatorname{sign}(\Theta_{ij})\}, & i \neq j,\ \Theta_{ij} \neq 0, \\ [-1, 1], & i \neq j,\ \Theta_{ij} = 0, \\ \{0\}, & i = j. \end{cases}$$

KKT stationarity:

$$\boxed{\Theta^{-1} \;=\; S + \lambda Z} \quad (\text{plus } \Theta \succ 0).$$

GLASSO Pseudocode

Step E: Initialization

$\Theta \leftarrow \mathrm{diag}\big(1/\max(s_{ii}, \delta)\big)$
(*SPD seed*, $\delta > 0$).

Step F: Outer loop (block coordinate descent over columns)

repeat

$\Theta_{\text{prev}} \leftarrow \Theta$.

for $j = 1$ **to** N **do**

`// (1) Partition wrt. j`

Extract $S_{11} \in \mathbb{R}^{(N-1)\times(N-1)}$, $\boldsymbol{s}_{12} \in \mathbb{R}^{N-1}$ from S.

`// (2) Lasso subproblem (column j)`

$$\boldsymbol{\beta}^\star = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^{N-1}} \left\{ \tfrac{1}{2}\boldsymbol{\beta}^\top S_{11}\boldsymbol{\beta} - \boldsymbol{s}_{12}^\top\boldsymbol{\beta} + \lambda\|\boldsymbol{\beta}\|_1 \right\}.$$

Coordinate descent:

$$r \leftarrow s_{12,k} - \sum_{\ell\neq k}(S_{11})_{k\ell}\beta_\ell, \qquad \beta_k \leftarrow \mathrm{ST}\left(\frac{r}{(S_{11})_{kk}}, \frac{\lambda}{(S_{11})_{kk}}\right),$$

with $\mathrm{ST}(u,\tau) = \mathrm{sign}(u)\max(|u| - \tau, 0)$.

`// (3) Block update back to precision`

Map $\boldsymbol{\beta}^\star$ to $(\boldsymbol{\theta}_{12}, \theta_{22})$ and overwrite row/col j in Θ (keep symmetry/SPD).

until $\|\Theta - \Theta_{prev}\|_\infty \le \varepsilon_{outer}$

Step G: Partial correlations and graph

$d_i \leftarrow \sqrt{\Theta_{ii}}$; $P_{ij} \leftarrow -\Theta_{ij}/(d_i d_j)$ with $P_{ii} = 1$.

Graph: edge (i,j) iff $\Theta_{ij} \neq 0$, weight $= P_{ij}$.

Stat Arb.ipynb

Principal Component Analysis In Statistical Arbitrage

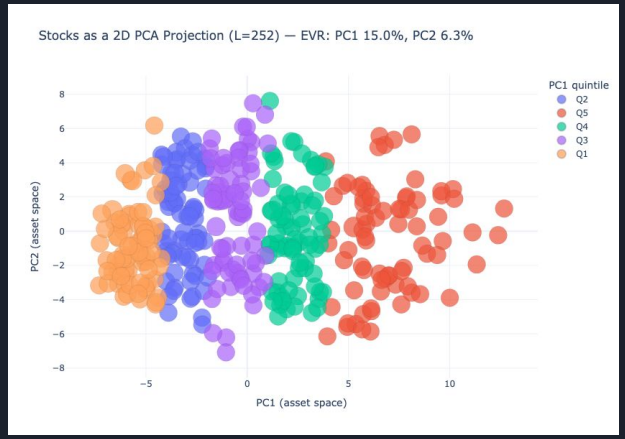
Principal Component Analysis

What is PCA?

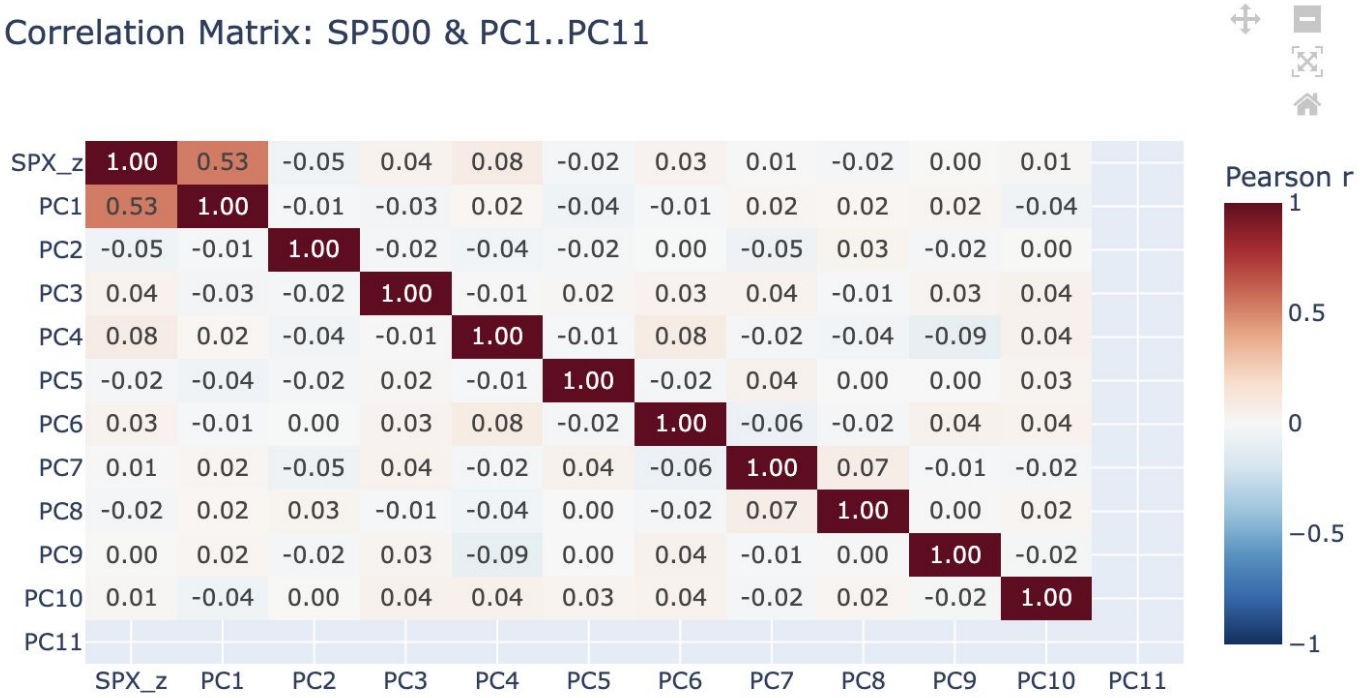
- PCA finds new axes (principal components) that maximizes variance
- Rotate the coordinate system to capture most variance with fewer axes

Why PCA in Statistical Arbitrage?

- Decompose stock returns – market vs idiosyncratic returns
- Adapts to changing regime – rolling windows capture market dynamics
- Ensures Market neutrality – removes PC1 market beta



Correlation Matrix of SPX vs PCs (PC1 Captures Market Return)



Example of PCA Application in Statistical Arbitrage

- **Step 1: PCA Output**
  Eigenvectors $U \in R^{N \times k}$ (stocks $\times$ factors).
  Each column of U is a portfolio weight vector ("eigenportfolio").

- **Step 2: Scale Eigenvectors**
  Raw U overweight volatile stocks.
  Scale each weight by stock volatility σ:

$$W_f = \frac{U}{\sigma}$$

- **Step 3: Factor Returns**
  Stock returns at time t: $r_t \in R^N$.
  Apply eigenportfolios:

$$F_t = W_f^\top r_t$$

  Factor returns $F_t \in R^k$ represent daily returns of PCs.

- **Step 4: Residual Extraction**
  For stock i at time t: regress returns on factor returns.

$$r_{i,t} = \alpha_i + \beta_i^\top F_t + \varepsilon_{i,t}$$

  Residual $\varepsilon_{i,t}$ = idiosyncratic return.

- **Step 5: Cumulative Process**
  Build residual spread as cumulative sum:

$$X_i(t) = \sum_{\tau \le t} \varepsilon_{i,\tau}$$

- **Step 6: Z-Score Standardization**
  Rolling mean $\bar{X}_i$ and std σ_i over lookback window.

$$z_i(t) = \frac{X_i(t) - \bar{X}_i}{\sigma_i}$$

- **Step 7: Trading Signals**
  If $z_i(t) \ge +enter \rightarrow$ enter short.
  If $z_i(t) \le -enter \rightarrow$ enter long.
  Exit when $|z_i(t)| \le exit$.

PCA

⇩

Factor Return

⇩

Residual Extraction

⇩

Cumulative Mean

⇩

Z-Score trading signals

PCA Pseudocode

1) Standardization

- **Define** `standardize_window(window_values)`
- **Input**:
  - Matrix of returns ($L \times N_w$), NaNs already dropped
- **Steps**:
  - Compute column means μ
  - Compute column standard deviations σ (ddof=1)
  - Replace any $\sigma = 0$ with 1.0
  - Standardize each column: $Z = (W - \mu)/\sigma$
- **Output**:
  - Z (standardized window)
  - μ (means)
  - σ (standard deviations)

2) PCA on Standardized Returns

- **Define** `pca_window(Z, fixed_k=None, var_threshold=0.55)`
- **Input**:
  - Z: standardized returns matrix $(L \times N)$
  - Optional: fixed number of components k or variance threshold τ
- **Steps**:
  - Compute covariance matrix:

$$C = \frac{1}{L-1} Z^\top Z$$

  - Perform eigendecomposition: $Cu^{(j)} = \lambda_j u^{(j)}$
  - Sort eigenvalues λ_j and eigenvectors in descending order
  - Choose k:
    - If k is fixed, keep exactly k
    - Else, select minimum k such that

$$\frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{N} \lambda_j} \geq \tau$$

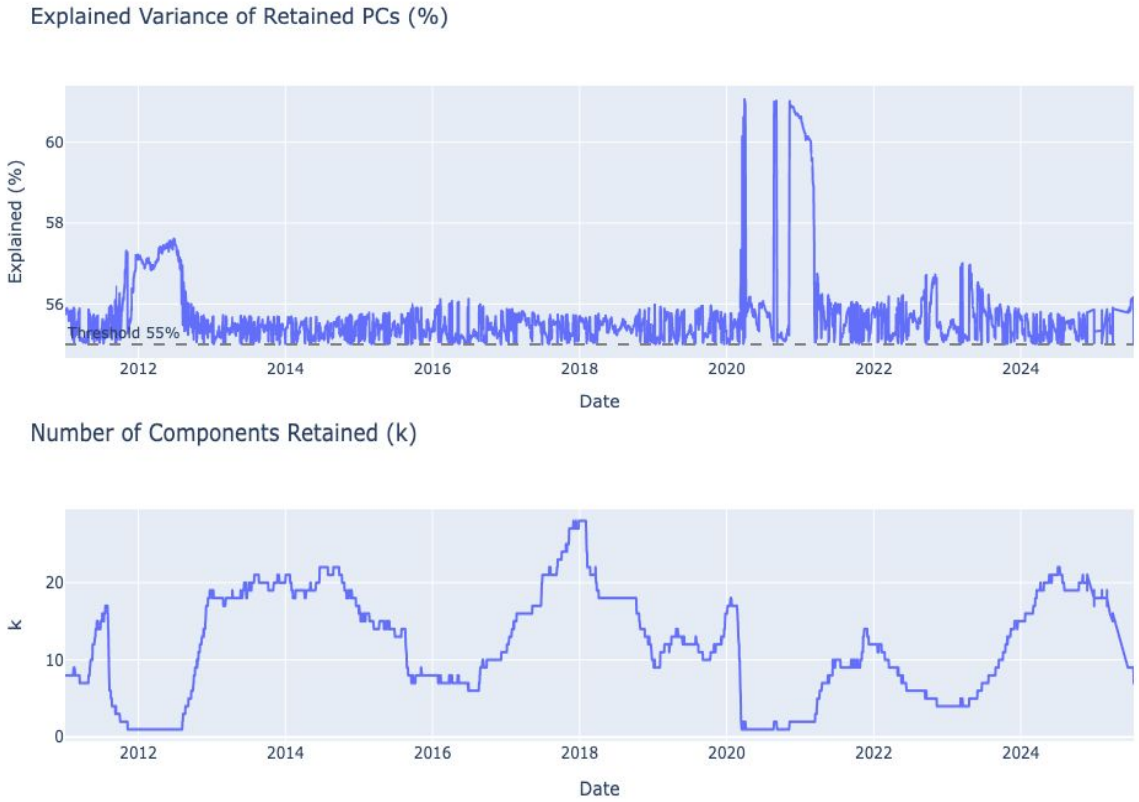
- **Output**:
  - U: eigenvectors (principal components, $N \times k$)
  - λ: top k eigenvalues
  - k: number of components used
  - explained variance fraction

PCA Pseudocode

3) Rolling PCA (Driver)

- **Define** `run_rolling_pca(Z, L=252, var_threshold=0.55)`
- **Input**:
  - Z: standardized return matrix (dates $\times$ tickers), no NaNs
  - L: rolling window length
  - Variance threshold τ
- **Steps**:
  - Initialize empty list `results`
  - For each end date t (from L to T):
    - Take the last L rows of Z ending at t: $Z_{t-L+1:t}$
    - Run PCA: $(U, \lambda, k, \mathit{explained}) \leftarrow$ `pca_window`$(Z_{t-L+1:t}, \tau)$
    - Append record $\{t, U, \lambda, k, \mathit{explained}\}$ to `results`
- **Output**:
  - `results`: list of PCA results for each rolling window

Variance Explained Time Series



PCA Pseudocode

3) Factor Returns

- **Define** `compute_factor_returns(rets, rolling_pca_results, L=252)`
- **Input**:
  - `rets`: returns (dates × tickers)
  - `rolling_pca_results`: list of {`date`, U, k, `valid_cols`}
- **Steps**:
  - Initialize empty table `fac`
  - For each record r:
    - Take PCA loadings U and compute weights $W_f = U \oslash \sigma$
    - Multiply by last-day returns r_t to get factor returns:

$$F_t = W_f^\top r_t$$

    - Append {date t, PC1=$F_{t,1}$, …, PCk=$F_{t,k}$} to `fac`
  - After loop: assemble table of PC returns by date
- **Output**:
  - `fac`: DataFrame of factor returns (PC1..PCK)

4) Regression for Residuals

- **Define** `compute_eps_last(rets, factor_returns, lookback=60)`
- **Input**:
  - `rets`: asset returns (dates × tickers)
  - `factor_returns`: DataFrame of PC factor returns (PC1..PCk)
  - `lookback`: window length for regression
- **Steps**:
  - For each date t:
    - Take past `lookback` days of returns for each stock
    - Regress stock returns on factor returns (plus intercept)
    - At date t, compute residual:

$$\varepsilon_{i,t} = r_{i,t} - \left(\alpha_i + \beta_i^\top F_t\right)$$

  - Store $\varepsilon_{i,t}$ for all stocks
- **Output**:
  - Matrix of last-day residuals $\varepsilon_{i,t}$ (dates × tickers)

PCA Pseudocode

5) Cumulative Spread Process

- **Define** `build_spread_from_eps(eps_last)`
- **Input**:
  - `eps_last`: matrix of last-day residuals $\varepsilon_{i,t}$ (dates × tickers)
- **Steps**:
  - Compute cumulative sum over time for each asset:

$$X_i(t) = \sum_{\tau \le t} \varepsilon_{i,\tau}$$

- **Output**:
  - `X`: cumulative residual process (dates × tickers)

6) Score Computation (Z-score)

- **Define** `compute_z_score(X, lookback=252, min_periods=126)`
- **Input**:
  - X: cumulative residual process (dates × tickers)
  - `lookback`: rolling window length; `min_periods`: minimum points to compute stats
- **Steps**:
  - Compute rolling mean $\bar{X}_i(t)$ over `lookback`
  - Compute rolling std $\sigma_i(t)$ (unbiased, ddof = 1)
  - Guard: if $\sigma_i(t) = 0$ or NaN $\Rightarrow$ set $z_i(t)$ to NaN
  - Compute Z-score:

$$z_i(t) = \frac{X_i(t) - \bar{X}_i(t)}{\sigma_i(t)}$$

- **Output**:
  - Z-score matrix z (dates × tickers)

7) Trading Signals (Z-score)

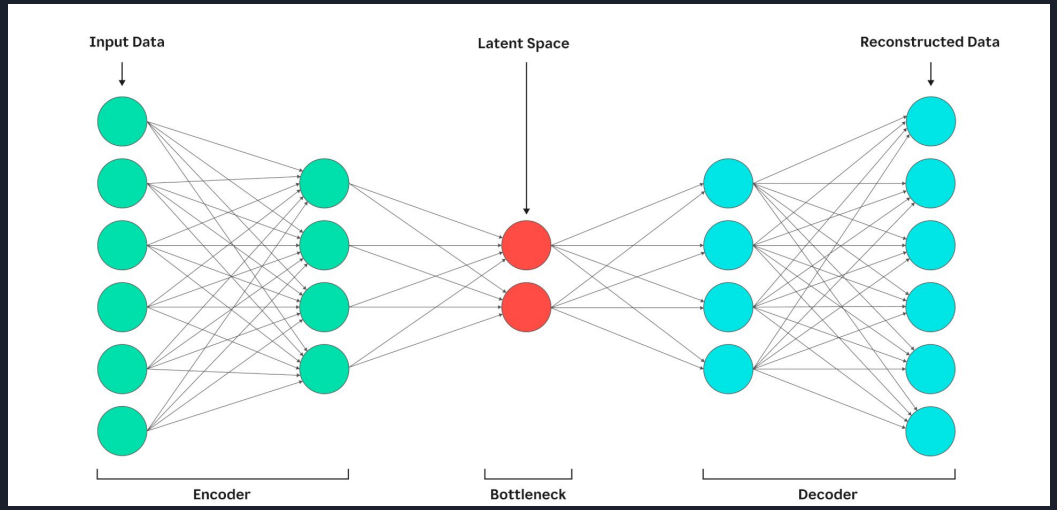
- **Define** `generate_signals_from_zscore(z, enter=1.25, exit=0.75)`
- **Input**:
  - z: matrix of Z-scores (dates $\times$ tickers)
  - `enter`, `exit`: entry/exit thresholds
- **Steps**:
  - For each asset i, maintain a `state` $\in \{-1, 0, +1\}$ (short, flat, long)
  - For each date t in order:
    - If $z_{i,t}$ is NaN: keep previous state
    - If state $= 0$ (flat):
      - If $z_{i,t} \leq -$`enter`: set state $\leftarrow +1$ (enter long)
      - If $z_{i,t} \geq +$`enter`: set state $\leftarrow -1$ (enter short)
    - Else (currently in a position):
      - If $|z_{i,t}| \leq$ `exit`: set state $\leftarrow 0$ (exit to flat)
    - Record signal $sig_{i,t} \leftarrow$ state
- **Output**:
  - sig: trading signals $\in \{-1, 0, +1\}$ (dates $\times$ tickers)

Trading Signals

2012-12-20 00:00:00	1	1	0	0	1	1	1	1	-1	-1	-1	1	-1	-
2012-12-21 00:00:00	1	1	0	0	1	1	1	1	-1	-1	-1	1	-1	-
2012-12-24 00:00:00	1	1	0	0	1	1	1	1	-1	-1	0	1	-1	-
2012-12-26 00:00:00	1	1	0	0	1	1	1	1	-1	-1	-1	1	-1	-
2012-12-27 00:00:00	1	1	0	0	1	1	1	1	-1	-1	0	1	0	-
2012-12-28 00:00:00	1	0	1	0	1	0	0	0	-1	-1	0	1	0	
2012-12-31 00:00:00	0	0	1	0	0	0	0	0	-1	-1	0	1	0	
2013-01-02 00:00:00	0	0	1	0	0	0	-1	0	-1	-1	0	1	0	
2013-01-03 00:00:00	0	0	1	0	0	0	0	-1	-1	-1	0	1	0	

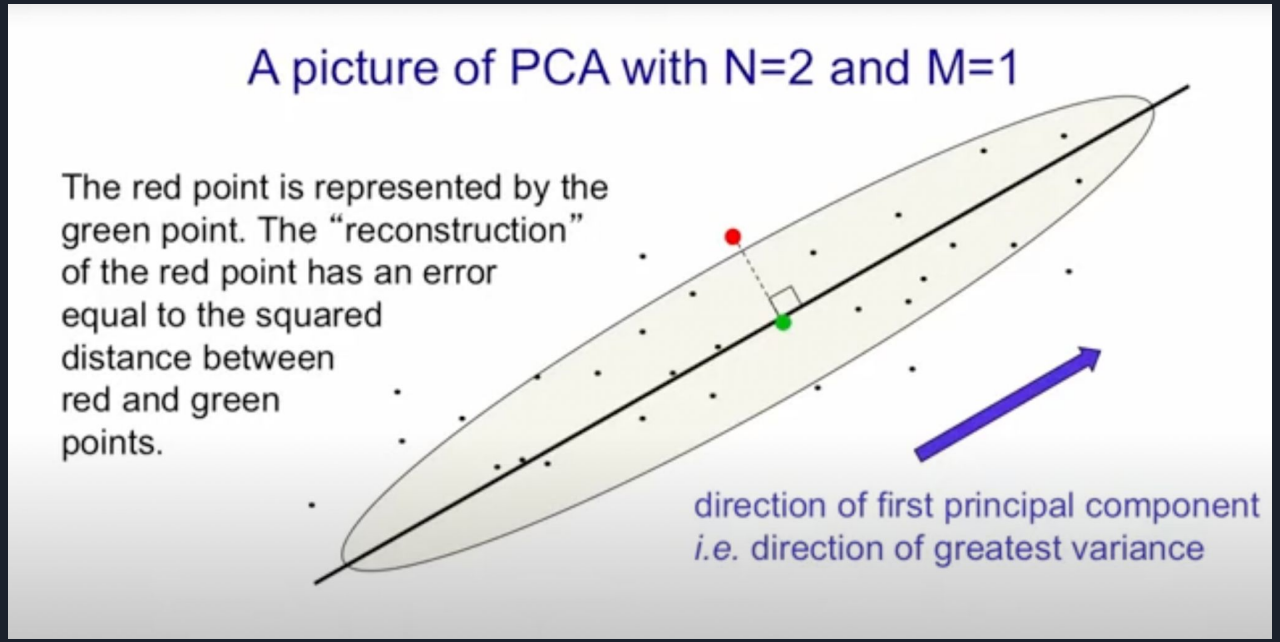
PCA vs Autoencoders

Autoencoders



- PCA is strictly linear
- Autoencoders can be nonlinear
- PCA finds the directions of maximum variance
- Autoencoders are trained with backprop to minimize reconstruction error
- A linear autoencoder (with one hidden layer, no nonlinear activation, and MSE loss) will actually learn the same subspace as PCA
- But nonlinear autoencoders go beyond PCA by capturing curved manifolds in the data

Intuition PCA ⊆ Autoencoders



Proof PCA ⊆ Autoencoders

Setup

Let $X \in \mathbb{R}^{n \times d}$ be a data matrix whose rows are centered. The compact singular value decomposition (SVD) is

$$X = U\Sigma V^\top,$$

where $U \in \mathbb{R}^{n \times r}$, $V \in \mathbb{R}^{d \times r}$, $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$ with $\sigma_1 \geq \cdots \geq \sigma_r > 0$, and $r = \mathrm{rank}(X)$.

For $k \leq r$, denote by U_k, Σ_k, V_k the matrices consisting of the first k singular vectors/values. The rank-k truncated SVD of X is

$$X_k := U_k \Sigma_k V_k^\top.$$

The k–dimensional PCA reconstruction projects X onto the span of the top k right singular vectors:

$$XV_kV_k^\top.$$

We show this equals X_k.

Indeed,

$$XV_kV_k^\top = (U\Sigma V^\top)V_kV_k^\top = U\Sigma\big(V^\top V_k\big)V_k^\top.$$

Since V is orthogonal, $V^\top V_k = \begin{bmatrix} I_k \\ 0 \end{bmatrix}$, hence

$$XV_kV_k^\top = U\Sigma \begin{bmatrix} I_k \\ 0 \end{bmatrix} V_k^\top = U_k\Sigma_k V_k^\top = X_k.$$

Thus, PCA reconstruction equals the truncated SVD.

The sample covariance of centered data is

$$S = \tfrac{1}{n} X^\top X.$$

Substituting $X = U\Sigma V^\top$ gives

$$S = \tfrac{1}{n} V\Sigma^2 V^\top.$$

Therefore, the eigenvalues of S are $\frac{1}{n}\sigma_i^2$ and the eigenvectors are the columns of V. Hence the top k eigenvectors of S are exactly V_k, the top right singular vectors of X. These are the principal directions of PCA.

Proof PCA ⊆ Autoencoders

Theorem 1. *Let $X \in \mathbb{R}^{n\times d}$ be centered data. A linear autoencoder with encoder $E \in \mathbb{R}^{d\times k}$ and decoder $D \in \mathbb{R}^{d\times k}$ achieves the same reconstruction as PCA:*

$$XED^\top \;=\; X_k \;=\; XV_kV_k^\top,$$

where $X_k = U_k\Sigma_kV_k^\top$ is the rank-k truncated SVD of X. Moreover, $\mathrm{col}(E) = \mathrm{span}(V_k)$ up to a change of basis, and the minimum squared loss is $\sum_{j>k}\sigma_j^2$.

Proof. A linear autoencoder produces reconstruction

$$\hat{X} \;=\; XED^\top \;=\; XA, \qquad A := ED^\top \in \mathbb{R}^{d\times d}.$$

Since $\mathrm{rank}(A) \le k$, we have $\mathrm{rank}(\hat{X}) \le k$ and $\mathrm{col}(\hat{X}) \subseteq \mathrm{col}(X)$. Thus the autoencoder objective is

$$\min_{E,D}\|X-\hat{X}\|_F^2 = \min_{\substack{Y=XA\\ \mathrm{rank}(Y)\le k}}\|X-Y\|_F^2 = \min_{\substack{Y\\ \mathrm{col}(Y)\subseteq\mathrm{col}(X),\,\mathrm{rank}(Y)\le k}}\|X-Y\|_F^2. \quad (\star)$$

By the Eckart–Young–Mirsky theorem, if $X = U\Sigma V^\top$ and $X_k = U_k\Sigma_kV_k^\top$, then

$$\min_{\mathrm{rank}(Y)\le k}\|X-Y\|_F^2 \;=\; \|X-X_k\|_F^2, \qquad \text{attained at } Y = X_k,$$

with optimal value $\sum_{j>k}\sigma_j^2$.

Moreover, $X_k = U_k\Sigma_kV_k^\top = XV_kV_k^\top$ lies in the feasible set of $(\star)$, since $\mathrm{col}(X_k) \subseteq \mathrm{col}(X)$ and $\mathrm{rank}(X_k) \le k$. Thus the constrained minimum equals the unconstrained one, and any global minimizer of the autoencoder satisfies

$$XED^\top \;=\; X_k \;=\; XV_kV_k^\top.$$

Finally, since $XED^\top = XV_kV_k^\top$, one valid factorization is $E = V_kR$ and $D = V_kR^{-\top}$ for some invertible $R \in \mathbb{R}^{k\times k}$. Hence $\mathrm{col}(E) = \mathrm{span}(V_k)$ up to a change of basis.

Therefore, a linear autoencoder exactly recovers PCA: its optimal reconstruction is X_k, the encoder spans the top-k right singular vectors, and the minimum squared loss is $\sum_{j>k}\sigma_j^2$. □

Eckart–Young–Mirsky

Lemma 1 (Unitary invariance of Frobenius norm)**.** *For any orthogonal Q_1, Q_2 and any matrix A,*

$$\|Q_1 A Q_2\|_F = \|A\|_F .$$

Theorem 1 (Eckart–Young–Mirsky (Frobenius case))**.** *Let $X = U\Sigma V^\top$ and $X_k = U_k \Sigma_k V_k^\top$. Then*

$$\min_{\operatorname{rank}(Y)\le k} \|X - Y\|_F^2 = \|X - X_k\|_F^2 = \sum_{j>k} \sigma_j^2,$$

and $Y = X_k$ is an optimal solution.

Proof. Let Y be any matrix with $\operatorname{rank}(Y) \le k$. Set $C = U^\top Y V$. By unitary invariance,

$$\|X - Y\|_F^2 = \|\Sigma - C\|_F^2.$$

Partition

$$\Sigma = \begin{bmatrix} \Sigma_k & 0 \\ 0 & \Sigma_\perp \end{bmatrix}, \qquad C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}.$$

Then

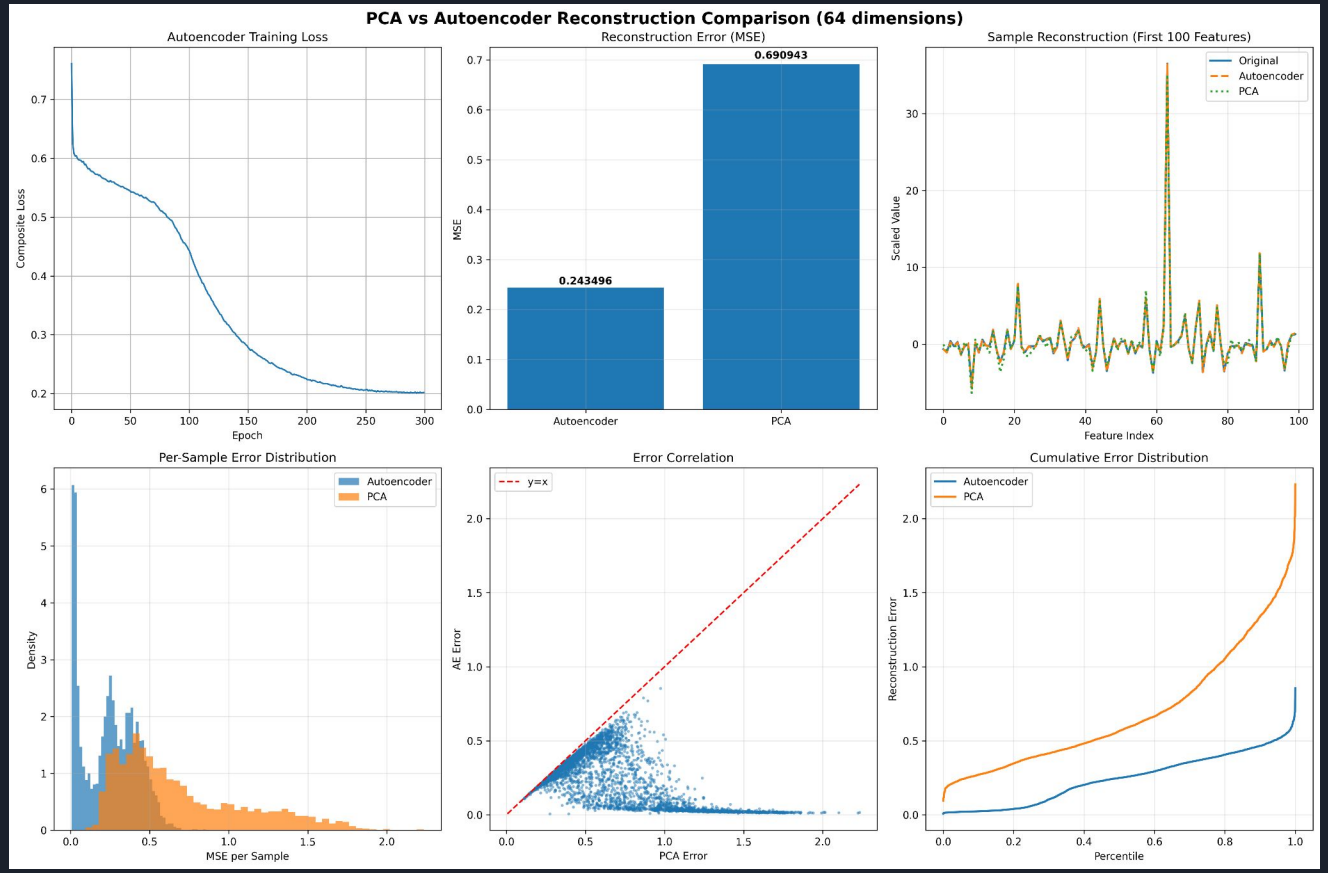
$$\|\Sigma - C\|_F^2 = \|\Sigma_k - C_{11}\|_F^2 + \|C_{12}\|_F^2 + \|C_{21}\|_F^2 + \|\Sigma_\perp - C_{22}\|_F^2.$$

All terms are nonnegative, so

$$\|\Sigma - C\|_F^2 \;\ge\; \|\Sigma_\perp\|_F^2 = \sum_{j>k} \sigma_j^2.$$

For $Y = X_k$, we have $U^\top Y V = \begin{bmatrix} \Sigma_k & 0 \\ 0 & 0 \end{bmatrix}$, which achieves the lower bound exactly. Thus X_k is optimal and the minimum value is $\sum_{j>k} \sigma_j^2$. □

Autoencoders for Stat Arb



Future of the Field

Future of Quantitative Modeling

- AI & ML integration → beyond linear models
- Alternative data (satellite, social, ESG signals)
- Real-time risk management with massive data
- Today’s laptop has more computing power than all MS quant computers in the 80s combined

Our Future!

Thank You!