Stable Portfolio Compression for Index Tracking: Tree-Based Basis Selection, Local Ridge Mapping, and Turnover-Aware Replication*

Angus Cheung

December 14, 2025

Abstract

Stable Portfolio Compression (SPC) is a pipeline for compressing a large index into a compact basis of representative assets while preserving index-like exposures and return behavior. The method combines noise-robust correlation estimation (Ledoit–Wolf and PCA denoising), graph-theoretic structure learning via a Minimum Spanning Tree (MST), basis selection heuristics on trees, local Ridge regressions over neighborhoods, and an analytic or turnover-penalized quadratic program to map index weights to basis weights.

1 Introduction

Managing indices with hundreds or thousands of constituents is costly to trade, monitor, and rebalance. Stable Portfolio Compression (SPC) reduces this complexity by selecting a small, representative set of basis assets that preserves index returns and exposures, enabling efficient tracking and replication with far fewer positions.

SPC converts correlations into distances, builds a Minimum Spanning Tree to reveal market structure, selects diverse basis assets, and models non-basis assets via expanding-window Ridge regressions on nearby basis names. The resulting coefficient maps translate index weights into basis weights that approximate the full index.

A key use case is passive exchange-traded fund (ETF) index tracking, where holding SPC-selected basis assets (or liquid proxies) achieves near-benchmark exposure with lower costs and operational burden.

2 Notation

Let:

- N be the number of assets, T the number of time periods.
- $P_{t,i} > 0$ be the price of asset i at time t; $\boldsymbol{P}_t \in \mathbb{R}_+^N$ the price vector.
- Returns $R_{t,i} := \frac{P_{t,i} - P_{t-1,i}}{P_{t-1,i}}$; $\boldsymbol{R}_t \in \mathbb{R}^N$.

*Project repository: https://github.com/angus4718/stable-portfolio-compression/

- $\widehat{\boldsymbol{\Sigma}} \in \mathbb{R}^{N\times N}$ an estimator of covariance; $\widehat{\boldsymbol{C}}$ the correlation matrix with entries $\widehat{C}_{ij} \in [-1, 1]$.
- Distance matrix $\boldsymbol{D}$ with entries $D_{ij} = \sqrt{2(1 - \widehat{C}_{ij})}$.
- Tree $\mathcal{T} = (V, E)$, $V = \{1, \ldots, N\}$, $E \subset V \times V$ with weights $w_{ij} = D_{ij}$ on edges.
- Basis set $B \subset V$, $|B| = k \ll N$.
- Index weights at time t: $\boldsymbol{w}_t^{\text{idx}} \in \mathbb{R}^N$, row-normalized across active assets.
- Basis weights at time t: $\boldsymbol{w}_t^B \in \mathbb{R}^k$.

3 Returns, Correlation, and Distance Estimation

3.1 Return construction

Given price levels $\{P_{t,i}\}_{t=0}^T$, define simple returns

$$R_{t,i} = \frac{P_{t,i}}{P_{t-1,i}} - 1, \quad t = 1, \ldots, T. \tag{1}$$

Collect $\boldsymbol{R} \in \mathbb{R}^{T\times N}$; for downstream linear models we drop rows that are all NaN (e.g., due to listing churn), and align series by pairwise overlap when forming second-order moments.

3.2 Pairwise-overlap correlation

For assets i, j, let $\mathcal{T}_{ij} := \{t : R_{t,i}, R_{t,j} \text{ both observed}\}$ and $n_{ij} = |\mathcal{T}_{ij}|$. The Pearson correlation under pairwise deletion is

$$\widehat{C}_{ij} = \begin{cases} \dfrac{\sum_{t\in\mathcal{T}_{ij}}(R_{t,i} - \bar{R}_i)(R_{t,j} - \bar{R}_j)}{\sqrt{\sum_{t\in\mathcal{T}_{ij}}(R_{t,i} - \bar{R}_i)^2}\sqrt{\sum_{t\in\mathcal{T}_{ij}}(R_{t,j} - \bar{R}_j)^2}}, & n_{ij} \geq m_{\min}, \\ \text{NaN (to be denoised/imputed)}, & \text{otherwise}, \end{cases} \tag{2}$$

with $\bar{R}_i$ computed on $\{t \in \mathcal{T}_{ij}\}$. Diagonals are fixed to $\widehat{C}_{ii} = 1$ if column i has at least $m_{\min}$ observations. This overlap-aware construction maximizes data usage under missingness patterns typical of dynamic universes.

3.3 Ledoit–Wolf (LW) shrinkage

Let $\boldsymbol{X} \in \mathbb{R}^{T\times N}$ be column-median-imputed returns (residual NaNs $\to 0$) with rows centered by column means when forming the sample covariance $\boldsymbol{S} = \frac{1}{T-1}\boldsymbol{X}^\top\boldsymbol{X}$. The LW estimator shrinks $\boldsymbol{S}$ toward a structured, well-conditioned target $\boldsymbol{F}$:

$$\widehat{\boldsymbol{\Sigma}}_{\text{LW}} = \delta^\star\boldsymbol{F} + (1 - \delta^\star)\boldsymbol{S}, \qquad \delta^\star \in [0, 1]. \tag{3}$$

We adopt the constant-correlation (a.k.a. equicorrelation) or diagonal-scalar target, both common in practice:

- Diagonal-scalar: $\boldsymbol{F} = \bar{\sigma}^2\boldsymbol{I}$ with $\bar{\sigma}^2 = \frac{1}{N}\sum_i S_{ii}$.
- Constant-correlation: $F_{ii} = S_{ii}$, $F_{ij} = \bar{\rho}\sqrt{S_{ii}S_{jj}}$ for $i \neq j$, with $\bar{\rho}$ the average sample correlation.

The optimal shrinkage intensity $\delta^\star$ minimizes $\mathbb{E}\|\widehat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma}\|_F^2$ and admits a closed form:

$$\delta^\star = \frac{\mathrm{Var}[\boldsymbol{S}] - \mathrm{Cov}[\boldsymbol{S}, \boldsymbol{F}] + \|\boldsymbol{S} - \boldsymbol{F}\|_F^2/T}{\|\boldsymbol{S} - \boldsymbol{F}\|_F^2}, \tag{4}$$

estimated consistently from the data (the scikit-learn LedoitWolf implementation computes this). LW guarantees positive definiteness and reduces eigenvalue spread, improving conditioning for downstream inversion (e.g., regressions, QP). Convert to correlation via

$$\widehat{C}_{ij} = \frac{\widehat{\Sigma}_{ij}}{\sqrt{\widehat{\Sigma}_{ii}\widehat{\Sigma}_{jj}}}, \qquad \widehat{C}_{ii} = 1, \tag{5}$$

with diagonal clipping to handle any numerical artifacts. LW is particularly beneficial when N is not negligible versus T (high-dimensional regime), where $\boldsymbol{S}$ is noisy and often ill-conditioned.

3.4 PCA-based correlation reconstruction: factor truncation and stabilization

Let $\boldsymbol{C}$ be the raw pairwise-overlap correlation with NaNs replaced by 0 for numerical routines. Compute its spectral decomposition (or SVD) and retain k principal components to reconstruct a low-rank approximation:

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top, \quad \lambda_1 \geq \cdots \geq \lambda_N, \tag{6}$$

$$k = \begin{cases} \text{user-specified}, & \text{fixed } k, \\ \min\{k' : \sum_{\ell=1}^{k'} \lambda_\ell / \sum_{\ell=1}^{N} \lambda_\ell \geq \tau\}, & \text{threshold } \tau \in (0,1), \end{cases} \tag{7}$$

$$\widehat{\boldsymbol{C}}_{\mathrm{PCA}} = \boldsymbol{U}_{[:,1:k]}\boldsymbol{\Lambda}_{1:k,1:k}\boldsymbol{U}_{[:,1:k]}^\top. \tag{8}$$

We then enforce symmetry, clip to $[-1, 1]$, and set $\mathrm{diag}(\widehat{\boldsymbol{C}}_{\mathrm{PCA}}) = \mathbf{1}$. Truncation suppresses high-frequency (idiosyncratic) noise that inflates small negative eigenvalues and destabilizes graph geometry. Economically, retained components approximate systematic factors; discarded components capture measurement noise and transient co-movements, which would otherwise distort distances on the asset manifold.

3.5 Correlation-to-distance transform

We map correlations to a Euclidean chordal distance on the unit sphere:

$$D_{ij} = \sqrt{2\left(1 - \widehat{C}_{ij}\right)}. \tag{9}$$

This choice is justified by the embedding argument: If returns are z-scored, the correlation $\widehat{C}_{ij}$ is the cosine of the angle θ_{ij} between unit vectors $\hat{\boldsymbol{r}}_i, \hat{\boldsymbol{r}}_j$. The chordal distance on the unit sphere satisfies $D_{ij} = \|\hat{\boldsymbol{r}}_i - \hat{\boldsymbol{r}}_j\|_2 = \sqrt{2(1 - \cos\theta_{ij})}$, giving a proper metric obeying the triangle inequality.

4 Minimum Spanning Tree (MST)

Given a complete weighted graph with weights $w_{ij} = D_{ij}$, we construct an MST $\mathcal{T}$ via Kruskal's algorithm.

4.1 Kruskal's algorithm

Let $\mathcal{E} = \{(i, j) : i < j\}$ and sort by weights. Maintain a disjoint-set union (*UnionFind*) with path compression and union-by-size.

$$\textbf{Input: } \text{edge list } \{(i, j, w_{ij})\}. \tag{10}$$
$$\textbf{Procedure: } \text{Initialize UF over } V;\ E_T \leftarrow \emptyset. \tag{11}$$
$$\text{For edges in ascending } w_{ij}: \tag{12}$$
$$\text{if UF.find}(i) \neq \text{UF.find}(j): \tag{13}$$
$$\text{UF.union}(i, j),\ E_T \leftarrow E_T \cup \{(i, j, w_{ij})\}. \tag{14}$$
$$\textbf{Output: } \mathcal{T} = (V, E_T). \tag{15}$$

4.2 Tree distances

Because $\mathcal{T}$ is a tree, the path between any u, v is unique. Let the adjacency be $A(u) = \{(v, w_{uv})\}$. Distances $d_{\mathcal{T}}(u, v)$ are computed by BFS/DFS accumulation of edge weights from u. All-pairs can be computed by BFS from each node.

4.3 Degrees and path betweenness

We define the degree as $\deg(u) = |A(u)|$. To compute the *path betweenness count* $\text{betw}(u)$ (the number of unordered pairs (s, t) whose unique path in a tree passes through u) efficiently for all nodes, we use a linear-time algorithm based on rooting the tree, computing parent/child relations, and aggregating subtree sizes.

Algorithm (all-nodes betweenness in a tree). Given a weighted or unweighted tree $\mathcal{T} = (V, E)$ with $n = |V|$:

1. Root selection and parent map:
   - 1.1. Pick an arbitrary root $r \in V$.
   - 1.2. Run a DFS from r to compute a parent map $\text{parent}(\cdot)$ and an order of visitation (children discovered after parent).
2. Subtree sizes:
   - 2.1. Initialize $\text{subtree}[u] \leftarrow 1$ for all u.
   - 2.2. Traverse nodes in reverse DFS order; for each node u, for every child v (i.e., $\text{parent}(v) = u$), accumulate
   $$\text{subtree}[u] \leftarrow \text{subtree}[u] + \text{subtree}[v].$$
3. Component sizes after removing u:
   - 3.1. Removing u partitions the tree into components formed by each child-subtree of u and, if $u \neq r$, the "above-u" remainder of size $n - \text{subtree}[u]$.
   - 3.2. Thus the multiset of component sizes is
   $$\mathcal{S}(u) = \{\text{subtree}[v] : \text{parent}(v) = u\} \cup \big(\{\, n - \text{subtree}[u] \,\} \text{ if } \text{parent}(u) \neq \varnothing\big).$$

4. Betweenness from component sizes:

   4.1. Let $s_\ell \in \mathcal{S}(u)$ and denote $S = \sum_\ell s_\ell = n - 1$. The number of unordered pairs (s, t) whose unique path passes through u equals the number of pairs drawn from different components:

$$\mathrm{betw}(u) \;=\; \sum_{\ell<m} s_\ell s_m \;=\; \frac{S^2 - \sum_\ell s_\ell^2}{2} \;=\; \frac{(n-1)^2 - \sum_\ell s_\ell^2}{2}.$$

5 Basis Selection on Trees

We seek $B \subset V$, $|B| = k$, that balances diversity (spread) and importance (weights). Let $w : V \to \mathbb{R}_{\geq 0}$ be a node weight signal (e.g., market cap) and $\tilde{w}$ its min–max normalization to $[0, 1]$. Stickiness parameter $\beta \in [0, 1]$ biases selection toward nodes in a previous basis `prev_basis` by adding a scaled bonus.

5.1 Max-spread selection

Define the average tree distance of node v to all nodes by $\bar{d}(v) = \frac{1}{|V|} \sum_{u \in V} d_{\mathcal{T}}(v, u)$. The initial score combines spread and weight; stickiness adds a bonus equal to a fraction of the maximum baseline score:

$$s_0^{\text{base}}(v) = \alpha\, \bar{d}(v) + (1 - \alpha)\, \tilde{w}(v), \tag{16}$$

$$s_0(v) = s_0^{\text{base}}(v) + \beta\, \mathbf{1}\{v \in \texttt{prev\_basis}\} \cdot \max_{u \in V} s_0^{\text{base}}(u). \tag{17}$$

Seed with $b_1 = \arg\max_v s_0(v)$ and maintain the distance $\texttt{nearest\_dist}[v] = \min_{b \in B} d_{\mathcal{T}}(v, b)$ to the nearest basis for every node v.
Greedy fill until $|B| = k$: for each candidate $v \notin B$,

$$s^{\text{base}}(v) = \alpha \cdot \texttt{nearest\_dist}[v] + (1 - \alpha) \cdot \tilde{w}(v), \tag{18}$$

$$s(v) = s^{\text{base}}(v) + \beta\, \mathbf{1}\{v \in \texttt{prev\_basis}\} \cdot \max_{u \notin B} s^{\text{base}}(u). \tag{19}$$

Select $v^\star = \arg\max_{v \notin B} s(v)$, update $B \leftarrow B \cup \{v^\star\}$, and refresh
$\texttt{nearest\_dist}[u] \leftarrow \min\{\texttt{nearest\_dist}[u], d_{\mathcal{T}}(u, v^\star)\}$ for all u.

5.2 Hub-and-branch selection

Compute a mixed centrality signal from degree and path betweenness, each min–max normalized to $[0, 1]$:

$$c(v) = \alpha \cdot \widetilde{\deg}(v) + (1 - \alpha) \cdot \widetilde{\texttt{betweenness}}(v), \tag{20}$$

$$h^{\text{base}}(v) = (1 - \gamma) \cdot \widetilde{c}(v) + \gamma \cdot \tilde{w}(v), \tag{21}$$

$$h(v) = h^{\text{base}}(v) + \beta\, \mathbf{1}\{v \in \texttt{prev\_basis}\} \cdot \max_{u \in V} h^{\text{base}}(u), \tag{22}$$

with $\gamma \in [0, 1]$ trading off centrality vs weights. Choose hubs H as the top-h nodes by $h(v)$ where

$$h = \max\big\{1,\ \min\big(k - 1,\ \lfloor 0.2k \rceil\big)\big\},$$

and add H to B.

Remove hubs from the tree and compute connected components (branches) $\{C\}$. For each branch, score candidates by distance to the nearest hub $u^\star$ mixed with weights and stickiness:

$$r^{\text{base}}(v) = \eta \cdot d_{\mathcal{T}}(v, u^\star) + (1-\eta) \cdot \tilde{w}(v), \tag{23}$$

$$r(v) = r^{\text{base}}(v) + \beta\, \mathbf{1}\{v \in \texttt{prev\_basis}\} \cdot \max_{u \in C} r^{\text{base}}(u), \tag{24}$$

with $\eta \in [0,1]$. Add one representative per branch in descending $r(v)$ until $|B| = k$. If short after exhausting branches, fill the remainder via a farthest-first greedy pass over residual nodes using `nearest_dist` updates as in max-spread selection.

6 Local Ridge Regressions

For each non-basis asset $i \in V \setminus B$, select q nearest basis neighbors by the full-sample distance matrix $\boldsymbol{D}$:

$$\mathcal{N}_i = \text{argmin}_{B' \subset B,\, |B'|=q} \sum_{b \in B'} D_{ib} \quad \text{(implemented by sorting } \{D_{ib}\}_{b \in B} \text{ and taking } q \text{ smallest).} \tag{25}$$

Fit Ridge regression without intercept on the overlapping sample:

$$\min_{\boldsymbol{\beta}_i \in \mathbb{R}^{|\mathcal{N}_i|}} \sum_{t \in \mathcal{T}_i} \left(R_{t,i} - \sum_{b \in \mathcal{N}_i} \beta_{i,b} R_{t,b} \right)^2 + \lambda \|\boldsymbol{\beta}_i\|_2^2, \tag{26}$$

with closed-form solution $\widehat{\boldsymbol{\beta}}_i = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$ where $\boldsymbol{y}$ is $R_{t,i}$ and $\boldsymbol{X}$ columns are $\{R_{t,b}\}_{b \in \mathcal{N}_i}$.

Reconstruction and diagnostics. For all times with non-missing neighbor returns, define reconstructed returns

$$\widehat{R}_{t,i} = \sum_{b \in \mathcal{N}_i} \widehat{\beta}_{i,b} R_{t,b}. \tag{27}$$

Compute RMSE and R^2 per asset on the fit sample:

$$\text{RMSE}_i = \sqrt{\frac{1}{|\mathcal{T}_i|} \sum_{t \in \mathcal{T}_i} (R_{t,i} - \widehat{R}_{t,i})^2}, \tag{28}$$

$$R_i^2 = 1 - \frac{\sum_t (R_{t,i} - \widehat{R}_{t,i})^2}{\sum_t (R_{t,i} - \bar{R}_i)^2}. \tag{29}$$

7 Weight Mapping: Assets to Basis

Let $\boldsymbol{A}_t \in \mathbb{R}^{N \times k}$ be the asset-to-basis mapping at time t formed by:

1. Identity rows for basis assets present in the universe: $(A_t)_{b,b} = 1$.
2. For non-basis i, coefficient row $(A_t)_{i,b} = \widehat{\beta}_{i,b}$ from the selected coefficient pivot date (static or most recent $\le t$).

Given index weights $\boldsymbol{w}_t^{\text{idx}} \in \mathbb{R}^N$, the analytic mapping is

$$\boldsymbol{w}_t^B = \boldsymbol{A}_t^\top \boldsymbol{w}_t^{\text{idx}} \quad \text{(or equivalently } \boldsymbol{w}_t^{B\top} = (\boldsymbol{w}_t^{\text{idx}})^\top \boldsymbol{A}_t). \tag{30}$$

This does not constrain $\mathbf{1}^\top \boldsymbol{w}_t^B = 1$, leaving residual mass as cash if the mapping under-fills.

7.1 Turnover-penalized quadratic program

To stabilize rebalancing, solve

$$\min_{\boldsymbol{w}\in\mathbb{R}^k} \underbrace{\left\|\boldsymbol{A}_t\boldsymbol{w}-\boldsymbol{w}_t^{\mathrm{idx}}\right\|_2^2}_{\text{fit}} + \lambda \underbrace{\left\|\boldsymbol{w}-\boldsymbol{w}_{t-1}^{B}\right\|_2^2}_{\text{turnover}}, \tag{31}$$

with unconstrained $\boldsymbol{w}$ (negative basis weights allowed). The objective is strictly convex with solution

$$\boldsymbol{w}_t^B = \left(\boldsymbol{A}_t^\top\boldsymbol{A}_t + \lambda\boldsymbol{I}\right)^{-1}\left(\boldsymbol{A}_t^\top\boldsymbol{w}_t^{\mathrm{idx}} + \lambda\boldsymbol{w}_{t-1}^B\right), \tag{32}$$

used directly or via a numerical solver (e.g., SLSQP). In the implementation, when no previous $\boldsymbol{w}_{t-1}^B$ exists, the mapping defaults to the analytic $\boldsymbol{A}_t^\top\boldsymbol{w}_t^{\mathrm{idx}}$.

8 Index and Replicate Returns; Backtesting

Given prices $\boldsymbol{P}_t$ and returns $\boldsymbol{R}_t$, index return uses lagged weights

$$r_t^{\mathrm{idx}} = \sum_{i=1}^{N} w_{t-1,i}^{\mathrm{idx}} R_{t,i}. \tag{33}$$

Replicate return uses basis returns and lagged basis weights

$$r_t^{\mathrm{rep}} = \sum_{b\in B} w_{t-1,b}^{B} R_{t,b}. \tag{34}$$

Active return $a_t = r_t^{\mathrm{rep}} - r_t^{\mathrm{idx}}$.

8.1 Tracking error and diagnostics

$$\mathrm{TE}_{\mathrm{RMSE}} = \sqrt{\frac{1}{T'}\sum_t a_t^2}, \tag{35}$$

$$\mathrm{running\_RMSE}_t = \sqrt{\frac{1}{t}\sum_{s=1}^{t} a_s^2}, \tag{36}$$

$$\text{ann. TE} \approx \mathrm{TE}_{\mathrm{RMSE}}\cdot\sqrt{12}, \tag{37}$$

$$R^2 := 1 - \frac{\sum_t (r_t^{\mathrm{idx}} - r_t^{\mathrm{rep}})^2}{\sum_t (r_t^{\mathrm{idx}} - \bar{r}^{\mathrm{idx}})^2}. \tag{38}$$

Turnover at date t for replicate weights $\boldsymbol{w}_t^B$:

$$\mathrm{turnover}_t = \frac{1}{2}\sum_{b\in B}\left|w_{t,b}^B - w_{t-1,b}^B\right|. \tag{39}$$

9 Synthetic Market Simulator

We present a configurable stochastic generator for monthly equity prices, shares outstanding, market capitalizations, and index membership. The model couples sector structure, heavy-tailed dispersion in shares outstanding, sector-correlated returns with optional drift, and Poisson-driven universe churn (additions/removals). It produces monthly panels of prices and market caps, market-cap–normalized index weights, and per-month constituent sets for a top-K index.

9.1 Universe, sectors, and shares outstanding

Let $\mathcal{S}$ be the sector set with $|\mathcal{S}| = S$. Sector $s \in \mathcal{S}$ initially contains n_s assets. Each asset i is assigned a sector $s(i) \in \mathcal{S}$. Shares outstanding Q_i are heavy-tailed to induce skewed market caps:

$$Q_i \sim \text{Shifted-Pareto}(\alpha, \text{scale}) \quad \text{with} \quad Q_i \leftarrow \max\big((X+1)\cdot \text{scale},\, 1\big),\ X \sim \text{Pareto}(\alpha). \tag{40}$$

Initial prices are fixed at $P_{0,i} = 100$, yielding initial market caps $M_{0,i} = Q_i P_{0,i}$. Tickers are uniquely identified and tagged with sector metadata and creation/removal months.

9.2 Within-sector correlation and monthly returns

For each month and sector s, we generate correlated Gaussian shocks with constant within-sector correlation. Let n_s be the number of active assets in sector s. Define the correlation matrix $\boldsymbol{C}_s \in \mathbb{R}^{n_s \times n_s}$ by

$$(\boldsymbol{C}_s)_{ij} = \begin{cases} 1, & i = j, \\ \rho_s, & i \neq j, \end{cases} \qquad \rho_s \in [0, 1), \tag{41}$$

and, if needed, stabilize to positive-definiteness via a small diagonal jitter. Draw standard normals $\boldsymbol{z}_s \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_{n_s})$ and set

$$\boldsymbol{\epsilon}_s = \boldsymbol{L}_s \boldsymbol{z}_s, \quad \boldsymbol{L}_s \boldsymbol{L}_s^\top = \boldsymbol{C}_s, \tag{42}$$

$$\sigma_s = \frac{\sigma_s^{\text{ann}}}{\sqrt{12}}, \qquad \mu_s \sim \mathcal{U}[\mu_{\min}, \mu_{\max}], \tag{43}$$

$$\boldsymbol{r}_s = \mu_s \boldsymbol{1} + \sigma_s \boldsymbol{\epsilon}_s. \tag{44}$$

Each asset i in sector s takes $R_{t,i} = (\boldsymbol{r}_s)_i$. Prices evolve with clipping to avoid non-positive values:

$$P_{t,i} = \max\Big(P_{t-1,i}\,(1 + R_{t,i}),\, \varepsilon\Big), \qquad \varepsilon > 0 \text{ small}. \tag{45}$$

Market caps update as $M_{t,i} = Q_i P_{t,i}$.

9.3 Additions and removals (churn)

At month t, the active universe is U_t. The number of removals and additions scale with $|U_t|$ via Poisson draws:

$$N_{\text{rem}} \sim \text{Poisson}(\lambda_{\text{rem}}\,|U_t|), \qquad N_{\text{add}} \sim \text{Poisson}(\lambda_{\text{add}}\,|U_t|). \tag{46}$$

Removals are sampled without replacement from the bottom-p fraction (by current market cap) of removable names in U_t. Additions are created with sector drawn uniformly from $\mathcal{S}$; each new asset receives fresh Q_i from the shifted-Pareto law, is initialized at $P_{t,i} = 100$, and begins contributing to prices and market caps from its creation month onward.

9.4 Index formation and weights

Let $I_t \subseteq U_t$ be the set of top-K tickers by market cap at time t (with K configurable). The market-cap–weighted index uses weights normalized over active names:

$$w_{t,i}^{\text{mkt}} = \frac{M_{t,i}}{\sum_{j \in U_t} M_{t,j}}, \qquad i \in U_t. \tag{47}$$

The simulator records, per month, the price panel $\{P_{t,i}\}$, market caps $\{M_{t,i}\}$, weights $\{w_{t,i}^{\text{mkt}}\}$, and constituent sets I_t.

9.5 Per-constituent weight capping algorithm

We optionally enforce a per-constituent maximum weight $\bar{w} \in (0, 1]$ on market-cap–normalized weights to control concentration. For each month t, let $\boldsymbol{w}_t \in \mathbb{R}^N$ denote the row-normalized weights over the set of active tickers $A_t \subseteq U_t$ (inactive tickers are treated as missing). The goal is to produce adjusted weights $\tilde{\boldsymbol{w}}_t$ that satisfy:

$$\text{(i)} \quad \tilde{w}_{t,i} \geq 0 \quad \forall i \in A_t, \qquad \text{(ii)} \quad \sum_{i \in A_t} \tilde{w}_{t,i} = 1, \tag{48}$$

$$\text{(iii)} \quad \tilde{w}_{t,i} \leq \bar{w} \quad \forall i \in A_t, \qquad \text{(iv)} \quad \tilde{w}_{t,i} = \text{NaN for } i \notin A_t. \tag{49}$$

A necessary and sufficient feasibility condition is

$$\bar{w} \cdot |A_t| \;\geq\; 1, \tag{50}$$

otherwise the cap is infeasible for period t.
We use an iterative clip-and-redistribute scheme that preserves the relative proportions among uncapped names at each step:

Initialization. Set $\tilde{\boldsymbol{w}}_t^{(0)} = \boldsymbol{w}_t$ on A_t and define the active receiver set $R_t^{(0)} = \{i \in A_t : \tilde{w}_{t,i}^{(0)} < \bar{w}\}$.

Iteration. For $k = 0, 1, 2, \ldots$:

1. Identify violators $V_t^{(k)} = \{i \in A_t : \tilde{w}_{t,i}^{(k)} > \bar{w}\}$. If $V_t^{(k)} = \varnothing$, stop and return $\tilde{\boldsymbol{w}}_t^{(k)}$.
2. Compute total excess
$$E_t^{(k)} \;=\; \sum_{i \in V_t^{(k)}} \big(\tilde{w}_{t,i}^{(k)} - \bar{w}\big). \tag{51}$$
Clip violators: set $\tilde{w}_{t,i}^{(k+1)} = \bar{w}$ for $i \in V_t^{(k)}$ and $\tilde{w}_{t,i}^{(k+1)} = \tilde{w}_{t,i}^{(k)}$ for $i \notin V_t^{(k)}$.
3. Determine receivers (currently uncapped and positive):
$$R_t^{(k)} \;=\; \{i \in A_t : \tilde{w}_{t,i}^{(k+1)} < \bar{w},\ \tilde{w}_{t,i}^{(k+1)} > 0\}. \tag{52}$$
If $R_t^{(k)} = \varnothing$ and $E_t^{(k)} > 0$, feasibility is violated; abort.
4. Redistribute the excess proportionally to receivers:
$$S_t^{(k)} \;=\; \sum_{j \in R_t^{(k)}} \tilde{w}_{t,j}^{(k+1)}, \qquad \tilde{w}_{t,i}^{(k+1)} \;\leftarrow\; \tilde{w}_{t,i}^{(k+1)} \;+\; E_t^{(k)} \cdot \frac{\tilde{w}_{t,i}^{(k+1)}}{S_t^{(k)}} \quad \forall i \in R_t^{(k)}. \tag{53}$$

10 Backtesting Protocols

10.1 Static scripts

1. Basis selection from all data to date: compute $\boldsymbol{D}$, $\mathcal{T}$, and B via max-spread or hub-branch.
2. Local Ridge: fit $\widehat{\beta}_{i,b}$ using neighbors from $\boldsymbol{D}$.
3. Weight mapping: compute $\boldsymbol{A}$ and $\boldsymbol{w}^B$ (optionally with turnover penalty).

10.2 Time-series analysis

At each date t:

1. Compute $\boldsymbol{D}_t$ from data up to t and select basis B_t ($|B_t| = k$).
2. Fit local Ridge mapping coefficients $\widehat{\beta}^{(t)}$ using data up to t.
3. Map $\boldsymbol{w}_t^{\text{idx}} \mapsto \boldsymbol{w}_t^B$ using the pivot at date t; optionally solve (31).
4. Evaluate with next-period returns using lagged weights.

10.3 Baseline strategy

Top-N baseline replicator: at each t, take the N_0 largest index weights and renormalize:

$$\tilde{w}_{t,i} = \begin{cases} \dfrac{w_{t,i}^{\text{idx}}}{\sum_{j \in \text{Top-}N_0} w_{t,j}^{\text{idx}}}, & i \in \text{Top-}N_0, \\ 0, & \text{otherwise.} \end{cases} \tag{54}$$

Replicate returns use $\tilde{\boldsymbol{w}}_{t-1}$ and full-universe returns.

11 Conclusions

SPC integrates robust covariance estimation, tree geometry, greedy max-spread and hub-branch selections, local regressions, and turnover-aware mapping to compress index exposures into a compact, tradable basis with controlled tracking error and turnover.