FACTORMINER: A SELF-EVOLVING AGENT WITH SKILLS AND EXPERIENCE MEMORY FOR FINANCIAL ALPHA DISCOVERY

Yanlong Wang[1,2] **Jian Xu**[1] **Hongkang Zhang**[1] **Shao-Lun Huang**[1*]
Danny Dongning Sun[2*] **Xiao-Ping Zhang**[1*]

[1]Tsinghua University [2]Peng Cheng Laboratory

ABSTRACT

Formulaic alpha factor mining is a critical yet challenging task in quantitative investment, characterized by a vast search space and the need for domain-informed, interpretable signals. However, finding novel signals becomes increasingly difficult as the library grows due to high redundancy. We propose FactorMiner, a lightweight and flexible self-evolving agent framework designed to navigate this complex landscape through continuous knowledge accumulation. FactorMiner combines a modular skill architecture that encapsulates systematic financial evaluation into executable tools with a structured experience memory that distills historical mining trials into actionable insights (successful patterns and failure constraints). By instantiating the Ralph Loop paradigm—retrieve, generate, evaluate, and distill—FactorMiner iteratively uses memory priors to guide exploration, reducing redundant search while focusing on promising directions. Experiments on multiple datasets across different assets and markets show that FactorMiner constructs a diverse library of high-quality factors with competitive performance, while maintaining low redundancy among factors as the library scales. Overall, FactorMiner provides a practical approach to scalable discovery of interpretable formulaic alpha factors under the "Correlation Red Sea" constraint.

1 INTRODUCTION

Discovering predictive alpha factors is central to quantitative trading and portfolio construction. In practice, automated factor discovery faces three fundamental challenges: (i) vast search complexity—the space of formulaic expressions grows combinatorially with operator compositions and parameters; (ii) poor knowledge accumulation—traditional search methods (genetic programming, reinforcement learning) fail to retain and reuse insights across exploration sessions, leading to repetitive trials; (iii) interpretability constraints—unlike black-box neural predictors, financial practitioners require transparent, auditable formulas with explicit financial logic for regulatory compliance and risk management.

Traditional approaches rely heavily on domain experts who manually craft factors based on financial intuition (Kakushadze, 2016; Fama & French, 1993). More recently, machine learning methods have been applied to asset pricing (Gu et al., 2020; Feng et al., 2020), demonstrating strong predictive power but often sacrificing interpretability. Genetic programming (Koza, 1992) and reinforcement learning (Yu et al., 2023; Zhao et al., 2025) offer automated search but suffer from knowledge forgetting: as the factor library grows and correlation constraints tighten, these methods lack mechanisms to accumulate structural knowledge about what works and what fails. Moreover, existing methods treat each mined factor in isolation without considering the global factor library perspective, they optimize individual factor quality but ignore how new factors interact with the existing library, leading to redundant discoveries and inefficient exploration.

We study formulaic factor mining as a task for self-evolving AI agents (Yao et al., 2022; Shinn et al., 2023). Each factor is an explicit expression over market fields (e.g., close, volume, VWAP)

*Corresponding authors.

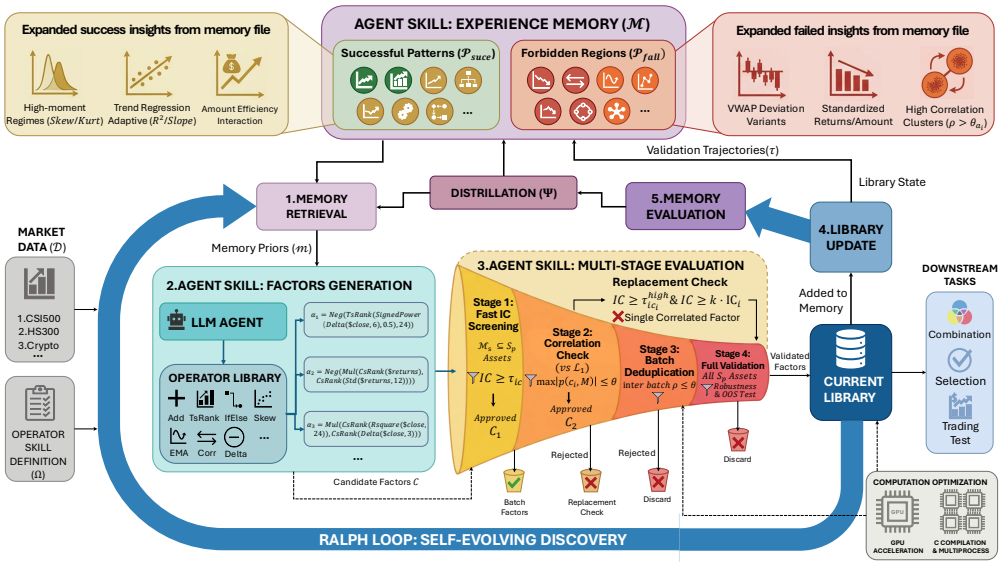


Figure 1: FactorMiner System Architecture. The Ralph Loop framework integrates three key components: (1) Experience Memory that stores successful patterns and forbidden regions from past mining sessions; (2) Agent Skill that encapsulates the multi-stage validation pipeline (IC screening, correlation checking, deduplication, and full validation); (3) Factor Library that grows dynamically while maintaining orthogonality constraints. The agent iteratively retrieves memory priors, generates candidates through the skill, and distills outcomes back into memory for improved future exploration.

composed from a library of 60+ operators. Unlike end-to-end neural approaches, formulaic factors enable human-in-the-loop auditing and compositional generalization across market regimes. The key research question is: how can an agent efficiently explore this vast program space while maintaining a global view of the factor library and autonomously accumulating structural knowledge across exploration sessions?

We propose FactorMiner, a self-evolving agent framework that addresses these challenges through two synergistic mechanisms: a compositional skill architecture and experience memory. First, we design factor mining as a reusable agent skill that can be invoked on-demand by a language model agent (Schick et al., 2023; Yao et al., 2022). The skill encapsulates domain knowledge—a curated operator library with 60+ financial operators, a multi-stage validation pipeline with IC thresholds and correlation checks, and standardized evaluation protocols—enabling flexible task decomposition and independent upgrades without retraining the agent. Second, we introduce experience memory that enables the agent to self-evolve through accumulated knowledge (Shinn et al., 2023; Park et al., 2023). The memory stores distilled structural patterns from historical mining sessions: successful patterns (factor templates that consistently pass quality thresholds) and forbidden regions (factor families with high mutual correlation to existing library members). Crucially, this memory maintains a global factor library perspective: the agent decides mining directions by considering how candidate factors complement the existing library, rather than optimizing individual factors in isolation.

The agent adopts the Ralph Loop paradigm for self-evolution: retrieve relevant patterns from experience memory, generate candidate factors by invoking the mining skill with retrieved priors, evaluate candidates through parallel validation, and distill outcomes back into memory. This creates a positive feedback cycle where each mining session improves future exploration efficiency, enabling the agent to continuously refine its search strategy.

To ensure computational efficiency, we build a lightweight yet high-performance system leveraging GPU-accelerated operators, multi-process parallelization, and C-compiled efficient numerical operations. This yields significant speedups over standard Python implementations (e.g., NumPy/Pandas), making large-scale iterative evaluation computationally feasible. We summarize our main contributions as follows:

- Experience memory for agent-based factor mining. We introduce structured experience memory into agent-based factor discovery, enabling self-evolution through accumulated knowledge. The memory stores reusable patterns that guide exploration and reduce redundant search compared to memoryless baselines.
- Modular skill architecture for on-demand invocation. We design factor mining as a compositional, reusable skill that encapsulates domain knowledge in a standalone module. This enables flexible agent invocation, independent skill upgrades, and clear separation between agent reasoning and skill execution.
- Lightweight and efficient mining system. We build a high-performance factor evaluation engine using GPU accelerated operators, multi-process parallelization, and C-compiled efficient numerical operations, enabling large-scale factor mining. This supports rapid iteration and large-scale factor exploration while adapting to different server configurations and compute budgets.
- Global factor library perspective. We incorporate factor library admission mechanisms into the mining loop, enabling the agent to make mining decisions from a global library perspective. The agent considers how candidate factors complement the existing library, rather than optimizing individual factors in isolation.
- Open factor library with research and practical value. We provide 110 A-share equity factors with explicit formulaic expressions, validated on real market data. These interpretable factors serve as diagnostic tools to understand cross-market behavioral anomalies and market microstructure inefficiencies, offering both research insights and practical value for quantitative trading.

2 METHODOLOGY

2.1 PROBLEM FORMULATION

We define the alpha factor discovery task over a universe of M assets and a time horizon T. The raw market input is represented by a tensor $\mathcal{D} \in \mathbb{R}^{M\times T\times F}$, where each entry $d_{m,t}^{(j)}$ denotes the value of feature j (e.g., price, volume) for asset m at time t. Symbolic factor space. A symbolic alpha factor α is a computational program composed from an operator set Ω that transforms market states into a cross-sectional predictive signal $\mathbf{s}_t \in \mathbb{R}^M$:

$$\alpha : \mathcal{D}_{:,:t,:} \to \mathbf{s}_t, \quad s_t^{(m)} = \alpha(\mathbf{d}_{m,:t}) \tag{1}$$

where $s_t^{(m)}$ represents the predictive score for asset m relative to its peers. Each operator $o \in \Omega$ has a fixed arity and a typed signature (e.g., time-series $\to$ time-series, cross-section $\to$ cross-section), and programs $\alpha \in \mathcal{P}(\Omega)$ correspond to expression trees composed from Ω with admissible parameters; detailed definitions and categories of Ω are provided in Appendix B and Table 2.

The effectiveness of α is quantified by the Information Coefficient (IC), defined as the cross-sectional Spearman rank correlation between the signal $\mathbf{s}_t$ and subsequent returns $\mathbf{r}_{t+1} \in \mathbb{R}^M$:

$$\mathrm{IC}_t(\alpha) = \mathrm{Corr}_{\text{rank}}\big(\mathbf{s}_t, \mathbf{r}_{t+1}\big) \tag{2}$$

Consistency over time is measured by the Information Ratio (ICIR): $\mathrm{ICIR}(\alpha) = \mu(\mathrm{IC}_t)/\sigma(\mathrm{IC}_t)$.

Correlation metric. To enforce library diversity, we measure redundancy between two factors α and β by the time-average cross-sectional Spearman correlation of their realized signals:

$$\rho(\alpha,\beta) = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}} \mathrm{Corr}_{\text{rank}}\big(\mathbf{s}_t(\alpha), \mathbf{s}_t(\beta)\big), \tag{3}$$

where $\mathcal{T}$ denotes the set of evaluation timestamps. This definition matches our current implementation, but the framework can accommodate alternative dependence measures (e.g., time-series correlation, partial correlation, or non-linear dependence metrics) by replacing $\rho(\cdot,\cdot)$ accordingly.

Objective: Orthogonal Library Synthesis. FactorMiner treats the problem as the iterative construction of a diverse factor library $\mathcal{L} = \{\alpha_1, \ldots, \alpha_K\}$. The goal is to maximize the aggregate predictive quality of the library subject to a global redundancy constraint:

$$\mathcal{L}^* = \arg\max_{\mathcal{L}\subset\mathcal{P}} \sum_{\alpha\in\mathcal{L}} \Phi(\alpha) \quad \text{s.t.} \quad \forall \alpha_i \neq \alpha_j \in \mathcal{L} : |\rho(\alpha_i,\alpha_j)| < \theta \tag{4}$$

where $\mathcal{P}$ is the infinite space of constructible programs, $\Phi(\cdot)$ is a fitness metric, and θ is the correlation budget.

The Correlation Red Sea. As the library $\mathcal{L}$ populates, the feasible region for new orthogonal factors—$\mathcal{P}_{\text{orth}} = \{\alpha \in \mathcal{P} : \max_{g \in \mathcal{L}} |\rho(\alpha, g)| < \theta\}$—shrinks rapidly as diversity constraints tighten. Standard search methods (e.g., GP or RL) often get trapped in this correlation red sea because they lack mechanisms to track explored regions or failure patterns.

Decision-Theoretic Memory Formulation. To navigate the correlation red sea, we reformulate the discovery process as a sequential decision task over an evolving internal knowledge state $S_t = (\mathcal{L}_t, \mathcal{M}_t)$, where $\mathcal{M}_t$ represents the persistent experience memory. The agent first retrieves a context-dependent memory signal m_t from $\mathcal{M}_t$ and $\mathcal{L}_t$, and then samples candidates from a memory-conditioned policy:

$$\alpha \sim \pi(\alpha \mid m_t), \quad m_t = R(\mathcal{M}_t, \mathcal{L}_t) \tag{5}$$

The role of $\mathcal{M}$ is to induce a probability measure contraction over the program space $\mathcal{P}$. By distilling historical trajectories $\tau = \{(\alpha_i, R_i)\}_{i=1}^{B}$ into structured patterns, $\mathcal{M}$ shifts the sampling mass toward the orthogonal manifold $\mathcal{P}_{\text{orth}}$. Mathematically, the evolution of memory is governed by a distillation operator Ψ:

$$\mathcal{M}_{t+1} = \Psi(\mathcal{M}_t, \tau_t) \tag{6}$$

which updates the agent's belief distribution so that future exploration is steered toward higher-utility and lower-redundancy regions of $\mathcal{P}$.

2.2 Factor Mining Skill Architecture

Unlike traditional RL or monolithic agent approaches where domain logic is often hard-coded or entangled with the agent's reasoning loop, FactorMiner adopts a modular skill-based architecture. Inspired by the tool-augmented paradigm (Schick et al., 2023), we encapsulate the entire factor mining process as a standalone, reusable Agent Skill—a standardized interface that exposes high-level capabilities to the LLM while abstracting away low-level execution details. The overall interaction pattern is illustrated in Figure 1.

Compositional Design. The skill is structured as a hierarchical library of tools:

- Operator Layer: A curated set of 60+ financial operators (e.g., `TsRank`, `Rsquare`) implemented with GPU-accelerated backends. This ensures that the agent's symbolic proposals are executable and computationally efficient.
- Validation Pipeline: A rigorous, standardized protocol for factor assessment (`check_ic` $\rightarrow$ `check_correlation` $\rightarrow$ `admit`). By decoupling validation from generation, we ensure that the agent's "creativity" is bounded by strict quantitative constraints, preventing hallucinated or scientifically invalid discoveries.

Advantages of the Skill-Based Approach. This modular design offers three distinct advantages over monolithic architectures:

1. Prevention of Calculation Hallucination: Large language models often struggle with precise arithmetic and algorithmic execution. By offloading the evaluation to a deterministic, code-based skill, FactorMiner eliminates the risk of "hallucinated metrics," ensuring that every reported IC and Sharpe ratio is mathematically rigorous.
2. Cross-Domain Transferability: The skill is parameterized to support multiple markets (e.g., A-shares vs. Crypto) via configuration files. The same high-level agent reasoning logic can be applied to different financial domains simply by switching the underlying skill context, as demonstrated in our cross-market experiments (Section 3.2.2).
3. Independent Optimization: The skill's execution backends and evaluation pipeline can be optimized independently of the agent's reasoning model, improving throughput without retraining the LLM backbone.

2.3 EXPERIENCE MEMORY

A key component of FactorMiner is the experience memory $\mathcal{M}$, a structured knowledge base that accumulates insights from historical mining sessions. We formalize the dynamics of the memory system through three conceptual operators: Formation, Evolution, and Retrieval.

Memory Formation. At the end of each mining batch t, the agent analyzes the mining trajectory $\tau_t = \{(\alpha_i, R_i)\}_{i=1}^{B}$, where R_i represents the evaluation feedback (IC, correlation, etc.). A formation operator F selectively extracts informational artifacts:

$$\mathcal{M}_{t+1}^{\text{form}} = F(\mathcal{M}_t, \tau_t) \tag{7}$$

This process distills raw data into symbolic patterns, categorizing them into successful patterns $\mathcal{P}_{\text{succ}}$ (those that pass admission) and forbidden regions $\mathcal{P}_{\text{fail}}$ (those rejected due to high correlation).

Memory Evolution. Formed memory candidates are integrated into the existing knowledge base through an evolution operator E:

$$\mathcal{M}_{t+1} = E(\mathcal{M}_t, \mathcal{M}_{t+1}^{\text{form}}) \tag{8}$$

This operator consolidates redundant entries and discards low-utility information. For instance, if a specific `VWAP_Deviation` variant is admitted but shows 0.82 correlation with existing factors, it is reclassified into $\mathcal{P}_{\text{fail}}$ to prevent further redundant exploration.

Memory Retrieval. During the factor generation phase, the agent retrieves a context-dependent memory signal m_t via a retrieval operator R:

$$m_t = R(\mathcal{M}_t, \mathcal{L}_t) \tag{9}$$

The signal m_t serves as a prompt-level constraint for the LLM policy, effectively shaping the sampling distribution $\pi(\alpha \mid m_t)$. In practice, we store experience as compact natural-language templates with canonical examples (e.g., "recommended directions" and "forbidden directions"), and retrieve them by matching against the current library diagnostics and recent rejection reasons; examples are provided in Appendices H and I.

Memory Content. The resulting memory $\mathcal{M}$ maintains a persistent record of the evolving mining landscape:

(1) Mining State ($\mathcal{S}$): Tracks the global evolution of the factor library, including current library size $|\mathcal{L}|$, recent admission logs, and saturation metrics that signal when specific logical domains (e.g., price-volume reversal) are becoming overpopulated.

(2) Structural Experience ($\mathcal{P}$): This is the core of the agent's guidance system, categorized into:

- Recommended Directions ($\mathcal{P}_{\text{succ}}$): High-success logical templates distilled from recent batches, such as higher moment regimes (using Skew/Kurt for environment switching) and robust efficiency interaction.
- Forbidden Directions ($\mathcal{P}_{\text{fail}}$): Regions identified as "Red Seas" due to persistent high correlation with the existing library, such as simple VWAP Deviations or standardized returns.

(3) Strategic Insights ($\mathcal{I}$): High-level lessons learned from the mining process, such as the observation that non-linear combination strategies (e.g., XGBoost-based synthesis) significantly outperform linear ones, or specific operator warnings (e.g., the instability of high-order moments in high-frequency data).

2.4 RALPH LOOP: SELF-EVOLVING FACTOR DISCOVERY

FactorMiner adopts the Ralph Loop paradigm—an iterative refinement philosophy where agents accumulate experience and self-evolve through repeated interaction. We instantiate this paradigm for factor mining by integrating experience memory into the search process (Algorithm 1). The key insight is that memory enables the agent to learn how to search—avoiding redundant exploration while focusing on promising regions.
Our instantiation of the Ralph Loop for factor mining has four key properties:

Algorithm 1 Ralph Loop: Self-Evolving Factor Discovery

Input: Operator library Ω, experience memory $\mathcal{M}$, target library size K
Output: Factor library $\mathcal{L}$
Initialize $\mathcal{L} \leftarrow \emptyset$
repeat
  Step 1: Memory Retrieval
    Retrieve memory signal $m \leftarrow R(\mathcal{M}, \mathcal{L})$
  Step 2: Guided Generation
    Sample batch $\mathcal{C} \sim \pi(\alpha \mid m)$ using Ω
  Step 3: Multi-Stage Evaluation
    Stage 1: Fast IC screening (M_{fast} assets)
      $\mathcal{C}_1 \leftarrow \{\alpha \in \mathcal{C} : |\text{IC}(\alpha)| \geq \tau_{\text{IC}}\}$
    Stage 2: Correlation check against $\mathcal{L}$
      $\mathcal{C}_2 \leftarrow \{\alpha \in \mathcal{C}_1 : \max_{g \in \mathcal{L}} |\rho(\alpha, g)| < \theta\}$
    Stage 2.5: Replacement check
      For $\alpha \in \mathcal{C}_1 \setminus \mathcal{C}_2$, let $g^\star = \arg\max_{g \in \mathcal{L}} |\rho(\alpha, g)|$
      Replace $g^\star$ with α if $|\rho(\alpha, g^\star)| \geq \theta$,
      $\max_{g \in \mathcal{L} \setminus \{g^\star\}} |\rho(\alpha, g)| < \theta$, and $\Phi(\alpha) \geq \Phi(g^\star) + \Delta$
    Stage 3: Batch deduplication (intra-batch $\rho < \theta$)
    Stage 4: Full validation (M_{full} assets) and trajectory τ collection
  Step 4: Library Update
    Admit validated factors: $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{C}_{\text{admitted}}$
  Step 5: Memory Evolution
    Update strategy: $\mathcal{M} \leftarrow E(\mathcal{M}, F(\tau))$
until $|\mathcal{L}| \geq K$ or budget exhausted

Global library perspective. Unlike methods that optimize individual factors in isolation, our approach considers how each candidate complements the existing library $\mathcal{L}$. The correlation constraint (Stage 2) ensures diversity, while the replacement mechanism (Stage 2.5) allows high-quality factors to replace inferior ones.

Memory-guided exploration. By maintaining $\mathcal{P}_{\text{succ}}$ and $\mathcal{P}_{\text{fail}}$, the agent avoids redundant exploration of known failure regions while focusing on promising structural patterns.

Multi-stage evaluation. The multi-stage pipeline balances efficiency and accuracy: Stage 1 uses a small asset subset for fast screening, Stages 2–3 enforce correlation constraints (inter-factor and intra-batch), and Stage 4 performs full validation only on surviving candidates.

Self-evolution. After each iteration, the memory is updated via the evolution operator E (integrating insights from F), creating a feedback loop where the agent continuously improves its search strategy. This enables continual learning: knowledge accumulated in early sessions benefits later exploration.

Trajectory semantics. The trajectory τ_t records, for each evaluated candidate, its formula α, quality statistics (e.g., IC/ICIR), redundancy diagnostics (e.g., $\max_{g \in \mathcal{L}} |\rho(\alpha, g)|$), and the rejection/admission outcome (including whether a replacement was triggered). These fields are the inputs to the formation operator F and support experience distillation across batches.

Detailed specifications of the operator library, admission criteria, computational efficiency optimizations, and factor combination methods are provided in Appendices B, C, E, and F.

To illustrate the diversity structure of the resulting library, we visualize the correlation structure of the released full A-share factor library (110 admitted factors) in Appendix Figure 9. The heatmap is computed using Spearman correlations on cross-sectionally standardized realized factor signals over the common time–asset panel, and indicates that most factor pairs are weakly to moderately correlated, with only a few localized clusters of higher dependence.

3 EXPERIMENTS

3.1 EXPERIMENTAL SETUP

Datasets. We evaluate FactorMiner across A-share equities and the cryptocurrency market to assess its discovery efficiency and cross-asset generalization. For A-share equities, we utilize intraday 10-minute bars of three index universes: the CSI 500 and CSI 1000 index constituents for mid-cap and broader small-/mid-cap coverage, and the HS300 index constituents for large-cap representation, with over 25 million data points in aggregate. For the Cryptocurrency market, we use 10-minute bars of 64 major assets from Binance. All datasets cover a training period from 2024-Q1 to 2024-Q4 and a held-out test period in 2025. The prediction target is the next 10-minute open-to-close price change ratio.

Baselines and Metrics. We compare against five representative methods: (1) Alpha101 (Classic) (Kakushadze, 2016), a static library of hand-crafted formulas; (2) Alpha101 (Adapted), with parameters tuned for high-frequency data; (3) Random Formula Exploration (RF), randomly sampling type-correct expression trees from $\mathcal{P}(\Omega)$ under a bounded depth/size distribution; (4) GPLearn (Stephens & contributors), an enhanced genetic programming approach; and (5) AlphaAgent (Tang et al., 2025), an LLM-driven proposal-refinement framework. For a fair comparison, we apply the same admission rules to each method and evaluate an equal-sized factor set; if a method admits fewer factors, we complete the set by selecting the remaining candidates with the best IC. We also include a No Memory variant as an internal baseline in Section 3.3. Performance is quantified using standard predictive metrics: rank information coefficient for forecasting precision, and its ratio (ICIR) for stability across time. Unless otherwise stated, we use IC to denote this Spearman correlation based IC. When applicable, all baselines share the same operator library Ω, data fields, and evaluation/admission protocol, and are scored with a unified evaluation engine. For methods that require LLM-based proposal generation, we use Gemini 3.0 Flash unless otherwise stated. Implementation details are provided in Section D.

3.2 MAIN RESULTS

3.2.1 FACTOR QUALITY AND DIVERSITY

Table 1 reports 2025 out-of-sample results under a strict protocol: Top-40 factors are selected once on CSI500 (2024) and then frozen for evaluation on multiple datasets. Under this protocol, FactorMiner performs best among the compared methods across all four markets. In the Factor Library setting, it achieves IC/ICIR of 8.25%/0.77 on CSI500, with similar competitive improvements on the other markets (see Table 1).

In terms of redundancy, FactorMiner's selected factor set exhibits moderate pairwise dependence. As summarized by Avg $|\rho|$ in Table 1, the average pairwise absolute correlation is 0.30–0.31 on A-shares and 0.25 on Crypto. The correlation distribution further shows a controlled tail ($|\rho| \approx$ 0.44–0.45 on A-shares and 0.42 on Crypto), suggesting that the performance gains are not driven by a few near-duplicate signals. We further visualize the correlation structure of the full admitted factor library in Appendix Figure 9. The complete factor formulas and additional downstream analyses are provided in Appendix Sections K to M and S.

3.2.2 ROBUSTNESS ACROSS HETEROGENEOUS MARKETS

The cross-market evaluation reveals the generalization capability of the discovered factors. While the Cryptocurrency market represents a fundamentally different microstructure (24/7 trading, no price limits) compared to A-shares, FactorMiner maintains competitive performance on Crypto, with IC/ICIR of 3.82%/0.28 in the single-factor library evaluation and 9.48%/0.62 under a simple IC-weighted combination (see Table 1).

This robustness suggests that FactorMiner captures fundamental price-volume dynamics (e.g., liquidity-constrained reversals, volatility clustering) that are invariant across asset classes, rather than overfitting to the idiosyncrasies of a single market.

Table 1: Out-of-Sample performance comparison with a stricter protocol. Top-40 factors are selected on CSI500 (**2024**) and evaluated on **2025** across datasets. For Alpha101, Classic is restricted to the same candidate set as Adapted. All reported IC and ICIR use the paper's absolute-IC summary: $|\mathbb{E}[IC_t]|$ and $|\mathbb{E}[IC_t]|/\mathrm{std}(IC_t)$. Factor Combination uses the frozen Top-40 with EW and ICW (weights/signs determined on 2024). Factor Selection trains on 2024 and tests on 2025 using Lasso and XGBoost.

Dataset	Method	Factor Library (Top-40)			Factor Combination (Top-40)				Factor Selection (Train'24/Test'25)			
		IC (%)	ICIR	Avg $\lvert\rho\rvert$	EW IC	EW ICIR	ICW IC	ICW ICIR	Las. IC	Las. ICIR	XGB. IC	XGB. ICIR
CSI500	RF‡	2.68	0.25	**0.13**	6.98	0.41	12.02	0.90	13.81	1.15	8.99	0.90
	Alpha101 (Classic)	4.49	0.42	0.19	10.85	0.88	12.89	0.99	14.09	**1.27**	12.07	1.21
	Alpha101 (Adapted)	5.06	0.43	0.21	11.53	0.86	14.71	1.13	13.70	1.08	13.76	1.20
	GPLearn	6.04	0.43	0.44	10.30	0.62	13.38	1.00	12.44	1.17	9.86	0.95
	AlphaForge	4.48	0.38	0.36	7.12	0.60	11.13	0.93	10.49	0.87	11.30	1.25
	AlphaAgent	5.90	0.46	0.32	10.99	0.92	11.86	0.99	13.87	1.19	11.93	1.24
	FactorMiner (Ours)	**8.25**	**0.77**	0.31	**14.95**	**1.29**	**15.11**	**1.31**	**14.59**	1.21	**14.03**	**1.29**
CSI1000	RF‡	2.88	0.30	**0.13**	7.48	0.49	12.28	1.00	13.72	1.24	9.54	1.03
	Alpha101 (Classic)	4.86	0.50	0.19	11.37	1.02	13.14	1.08	**14.64**	**1.42**	11.11	1.17
	Alpha101 (Adapted)	5.32	0.49	0.21	11.95	0.98	**14.78**	1.21	13.08	1.09	**13.88**	1.26
	GPLearn	5.86	0.48	0.44	11.10	0.73	13.66	1.11	12.87	1.23	9.53	0.96
	AlphaForge	4.64	0.42	0.35	7.60	0.71	11.25	1.02	10.42	0.93	12.20	**1.34**
	AlphaAgent	6.21	0.51	0.32	11.17	1.04	12.00	1.12	13.73	1.27	11.42	1.22
	FactorMiner (Ours)	**7.78**	**0.76**	0.30	**14.62**	**1.37**	14.76	**1.39**	14.25	1.25	12.42	1.30
HS 300	RF‡	1.94	0.15	**0.13**	5.46	0.30	9.49	0.61	9.98	0.63	5.55	0.46
	Alpha101 (Classic)	3.44	0.26	0.18	8.55	0.57	10.31	0.65	11.84	0.85	9.70	0.74
	Alpha101 (Adapted)	4.00	0.28	0.20	9.31	0.60	12.03	0.76	12.04	0.77	**11.81**	0.90
	GPLearn	4.12	0.16	0.45	8.01	0.44	10.64	0.66	9.59	0.58	7.92	0.61
	AlphaForge	3.53	0.25	0.36	5.19	0.35	8.54	0.57	6.79	0.43	10.89	0.91
	AlphaAgent	4.69	0.30	0.33	8.83	0.59	9.65	0.65	**12.26**	**0.86**	9.66	0.84
	FactorMiner (Ours)	**7.46**	**0.38**	0.31	**12.49**	**0.88**	**12.66**	**0.88**	11.23	0.82	11.33	**0.94**
Crypto	RF‡	1.45	0.09	**0.07**	3.31	0.19	7.91	0.48	8.04	0.51	3.50	0.24
	Alpha101 (Classic)	2.11	0.14	0.14	5.91	0.41	8.05	0.53	**9.63**	**0.54**	5.61	0.37
	Alpha101 (Adapted)	2.40	0.15	0.15	6.09	0.41	9.21	**0.62**	8.45	0.48	4.51	0.30
	GPLearn	2.50	0.15	0.38	4.44	0.25	7.62	0.42	9.07	0.48	4.04	0.27
	AlphaForge	2.52	0.16	0.31	4.63	0.29	7.44	0.44	8.39	0.42	5.00	0.33
	AlphaAgent	2.86	0.17	0.27	7.94	0.34	8.40	0.36	7.48	0.41	2.35	0.17
	FactorMiner (Ours)	**3.82**	**0.28**	0.25	**9.48**	**0.61**	**9.48**	**0.62**	8.98	0.51	**6.82**	**0.47**

Note: IC is computed as Spearman rank correlation per bar and summarized as $|\mathbb{E}[IC_t]|$. Las.=Lasso, XGB.=XGBoost. ‡RF=Random Exploration.

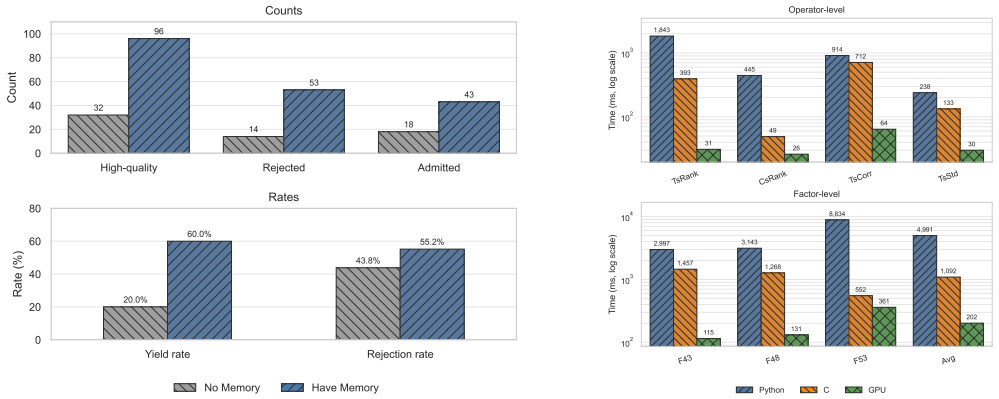
3.2.3 Ensembles vs. Learned Selection

Beyond individual factors, we evaluate the utility of the mined libraries for downstream portfolio construction via two families of methods: simple factor combinations (equal-weight and IC-weighted) and learned selection models (Lasso and XGBoost trained on 2024 and evaluated on 2025). Across most baselines, learned models provide a clear uplift over simple ensembles, consistent with the presence of heterogeneous yet partially redundant signals in the candidate libraries. For FactorMiner, however, learned selection offers limited additional gains and can be slightly negative on some datasets (e.g., CSI500: EW and ICW ICIR = 1.52/1.54 vs. Lasso and XGBoost = 1.42/1.52), indicating that simple ensemble signals already capture most of the exploitable predictive power under the Train'24/Test'25 protocol (see Table 1).

3.3 Effect of Experience Memory

To isolate the contribution of the Experience Memory ($\mathcal{M}$), we conducted an ablation study comparing FactorMiner against a "No Memory" variant where operators F, E, R were disabled, reducing the system to standard LLM-based evolution (see Figure 2a for a summary). Since this experiment is designed as a controlled mechanism ablation, we adopt relatively relaxed screening thresholds to ensure sufficient samples for comparison (IC threshold $|\text{IC}| > 0.02$; redundancy threshold $\theta = 0.85$ for this ablation setting).Figure 2a highlights the impact of memory guidance:

- Precise Navigation (High Yield): The Have Memory variant generates 96 high-quality candidates (60.0% yield), whereas the No Memory baseline produces only 32 (20.0% yield). This demon-



(a) Experience memory ablation.

(b) Efficiency benchmark.

Figure 2: Ablation and efficiency results. Left: ablation comparison between Have Memory and No Memory (high-quality candidates pass $|\text{IC}| > 0.02$). Right: computation time (log scale) for operator-level and factor-level benchmarks (lower is better).

strates that the retrieval operator effectively maps the search space, allowing the agent to guiding exploration toward regions with higher expected yield rather than exploring randomly.

- Aggressive Filtration (High Diversity): Despite generating $3.0\times$ more valid signals, the Have Memory variant actively rejects a higher proportion of them for redundancy (55.2% vs 43.8%). This confirms that the evolution operator (E) functions as a strategic filter, prioritizing unique signal discovery over mere quantity.

3.4 Mining Efficiency Analysis

To demonstrate the practical advantage of the factor mining skill, we conduct rigorous benchmarks comparing three execution backends supported by our framework: (1) standard Python (Pandas), (2) C-compiled (Bottleneck) for efficient CPU execution, and (3) GPU-accelerated for massive parallelism. This multi-backend support allows the agent to remain lightweight while achieving industrial-grade efficiency through on-demand acceleration.

Results. Figure 2b presents both operator-level and factor-level benchmarks on the real CSI500 dataset (12,610 $\times$ 500), measuring pure computation time excluding I/O. At the operator level, the GPU backend achieves 8–59$\times$ speedups over Pandas and 2–13$\times$ over the optimized C backend. The most dramatic improvement is on `TsRank`: 1,843ms (Pandas) $\rightarrow$ 393ms (C) $\rightarrow$ 31ms (GPU).

For end-to-end factor evaluation, rank-intensive factors (F43, F48, F53) show substantial 23–27$\times$ speedups. Notably, even the highly-optimized C implementation is 5.4$\times$ slower than the GPU backend on average (1,092ms vs. 202ms). This hybrid acceleration strategy demonstrates that our factor mining skill is both flexible and high-performance: by offloading computationally intensive primitives to GPU/C while leveraging multi-process parallelism for batch evaluation, FactorMiner makes large-scale iterative discovery feasible on commodity hardware.

4 Conclusion

FactorMiner provides a lightweight self-evolving agent framework for interpretable high-frequency alpha discovery by combining a modular mining skill with experience memory. The resulting library of 110 formulaic factors and a standardized evaluation protocol form a reproducible discovery artifact for hypothesis-driven analysis of market microstructure. Future work will incorporate transaction-cost-aware backtesting, extend to broader assets and frequencies, and develop online memory updates for non-stationary markets.

REFERENCES

Franklin Allen and Risto Karjalainen. Using genetic algorithms to find technical trading rules. *Journal of Financial Economics*, 51(2):245–271, 1999. doi: 10.1016/S0304-405X(98)00052-X.

Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z. Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control. *arXiv preprint arXiv:1606.04460*, 2016. doi: 10.48550/arXiv.1606.04460. URL https://arxiv.org/abs/1606.04460.

Eugene F. Fama and Kenneth R. French. Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33(1):3–56, 1993. doi: 10.1016/0304-405X(93)90023-5. URL https://doi.org/10.1016/0304-405X(93)90023-5.

Guanhao Feng, Stefano Giglio, and Dacheng Xiu. Taming the factor zoo: A test of new factors. *The Journal of Finance*, 75(3):1327–1370, 2020. doi: 10.1111/jofi.12883.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, pp. 1126–1135. PMLR, 2017. URL https://proceedings.mlr.press/v70/finn17a.html.

Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5):2223–2273, 2020. doi: 10.1093/rfs/hhaa009.

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. *arXiv preprint arXiv:1802.07245*, 2018. doi: 10.48550/arXiv.1802.07245. URL https://arxiv.org/abs/1802.07245.

Zura Kakushadze. 101 formulaic alphas. *Wilmott*, 2016(84):72–81, 2016. doi: 10.1002/wilm.10525. URL https://doi.org/10.1002/wilm.10525. arXiv:1601.00991.

Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, et al. MRKL systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. *arXiv preprint arXiv:2205.00445*, 2022. doi: 10.48550/arXiv.2205.00445.

John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992.

Serhiy Kozak, Stefan Nagel, and Shrihari Santosh. Shrinking the cross-section. *Journal of Financial Economics*, 135(2):271–292, 2020. doi: 10.1016/j.jfineco.2019.06.008.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-bank: A comprehensive benchmark for tool-augmented LLMs. *arXiv preprint arXiv:2304.08244*, 2023. doi: 10.48550/arXiv.2304.08244. EMNLP 2023.

Alberto Moraglio, Krzysztof Krawiec, and Colin G. Johnson. Geometric semantic genetic programming. In *Parallel Problem Solving from Nature – PPSN XII*, volume 7491 of *Lecture Notes in Computer Science*, pp. 21–31. Springer, 2012. doi: 10.1007/978-3-642-32937-1_3.

Christopher J. Neely, Paul A. Weller, and Rob Dittmar. Is technical analysis in the foreign exchange market profitable? A genetic programming approach. *Journal of Financial and Quantitative Analysis*, 32(4), 1997. doi: 10.2307/2331231. JSTOR:2331231.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018. doi: 10.48550/arXiv.1803.02999. URL https://arxiv.org/abs/1803.02999.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2024. doi: 10.48550/arXiv.2310.08560.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. doi: 10.1016/j.neunet.2019.01.012.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*, pp. 1–22, 2023. doi: 10.1145/3586183.3606763.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. *arXiv preprint arXiv:2305.15334*, 2023. doi: 10.48550/arXiv.2305.15334.

Riccardo Poli, William B. Langdon, and Nicholas F. McPhee. *A Field Guide to Genetic Programming*. Lulu.com, 2008. ISBN 978-1-4092-0073-4. URL http://www.gp-field-guide.org.uk/.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. *arXiv preprint arXiv:2307.16789*, 2023. doi: 10.48550/arXiv.2307.16789.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019. doi: 10.1038/s42256-019-0048-x. URL https://doi.org/10.1038/s42256-019-0048-x.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023. doi: 10.48550/arXiv.2302.04761. URL https://arxiv.org/abs/2302.04761.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving AI tasks with chatgpt and its friends in hugging face. *arXiv preprint arXiv:2303.17580*, 2023. doi: 10.48550/arXiv.2303.17580.

Hao Shi, Weili Song, Xinting Zhang, Jiahe Shi, Cuicui Luo, Xiang Ao, Hamid Arian, and Luis Seco. AlphaForge: A framework to mine and dynamically combine formulaic alpha factors. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(12):12524–12532, 2025. doi: 10.1609/aaai.v39i12.33365.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023. doi: 10.48550/arXiv.2303.11366. URL https://arxiv.org/abs/2303.11366.

Trevor Stephens and contributors. gplearn: Genetic programming in python. GitHub repository. URL https://github.com/trevorstephens/gplearn. Accessed 2026-02-08.

Ziyi Tang, Zechuan Chen, Jiarui Yang, Jiayao Mai, Yongsen Zheng, Keze Wang, Jinrui Chen, and Liang Lin. AlphaAgent: LLM-driven alpha mining with regularized exploration to counteract alpha decay. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2813–2822. ACM, 2025. doi: 10.1145/3711896.3736838.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. doi: 10.48550/arXiv.2305.16291.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022. doi: 10.48550/arXiv.2210.03629. URL https://arxiv.org/abs/2210.03629. ICLR 2023.

Shuo Yu, Hongyan Xue, Xiang Ao, Feiyang Pan, Jia He, Dandan Tu, and Qing He. Generating synergistic formulaic alpha collections via reinforcement learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 5476–5486. ACM, 2023. doi: 10.1145/3580305.3599831.

Junjie Zhao, Chengxi Zhang, Min Qin, and Peng Yang. Quantfactor REINFORCE: Mining steady formulaic alpha factors with variance-bounded REINFORCE. *IEEE Transactions on Signal Processing*, 73:2448–2463, 2025. doi: 10.1109/TSP.2025.3576781.

A Related Work

A.1 Automated Alpha Factor Discovery

Traditional factor discovery relies on domain expertise to manually craft interpretable signals. Kakushadze (2016) presents 101 formulaic alphas with explicit expressions and low mutual correlations (15.9%), demonstrating that hand-crafted factors can be both diverse and effective; Alpha191 extends this with 191 formulas covering additional microstructure patterns. However, these classical approaches are limited to daily-frequency data, and despite explicit formulaic expressions, their economic interpretability remains opaque for complex nested structures. Consequently, high-frequency markets currently lack comparable curated factor libraries for systematic research.

To automate factor discovery, evolutionary program search methods such as genetic programming represent factor formulas as executable programs and evolve them via crossover and mutation (Koza, 1992; Neely et al., 1997; Allen & Karjalainen, 1999). In practice, vanilla genetic programming can explore the expression space inefficiently, exhibiting slow convergence and limited semantic guidance because genetic operators primarily act on program syntax rather than program behavior (Poli et al., 2008; Moraglio et al., 2012). More recently, machine learning methods have been applied to empirical asset pricing and high-dimensional model selection (Gu et al., 2020; Kozak et al., 2020; Feng et al., 2020). While these methods achieve strong predictive performance, they are often less transparent than formulaic signals and can be difficult to interpret or audit in high-stakes settings (Rudin, 2019).

Reinforcement learning has been used to navigate the discrete space of formulaic factor expressions by treating evaluation metrics (e.g., IC/ICIR) as rewards (Yu et al., 2023; Zhao et al., 2025), often at the cost of additional training and repeated evaluation overhead. In parallel, neural and LLM-driven frameworks generate and refine formulaic factors with exploration strategies designed to mitigate decay and redundancy (Shi et al., 2025; Tang et al., 2025). Despite these advances, these approaches still face the challenge of explicitly and persistently reusing structural patterns across mining sessions; as the factor library grows and redundancy/correlation constraints tighten, exploration can become increasingly repetitive (Feng et al., 2020; Kozak et al., 2020).

A.2 AI Agents with Skills and Memory

Recent language-model agents shift from one-shot text generation to closed-loop task execution via tool calls and feedback. Toolformer (Schick et al., 2023) trains LLMs to insert API calls in a self-supervised manner, while ReAct (Yao et al., 2022) interleaves reasoning traces with actions—the agent plans, invokes tools, observes outcomes, and iterates. This tool-augmented view motivates our skill-based design: factor mining is packaged as an executable skill that an agent can invoke on demand.

Memory and self-improvement mechanisms further let agents accumulate experience over time. Reflexion (Shinn et al., 2023) uses language-based self-reflection to store lessons in natural language, and generative agents (Park et al., 2023) maintain memory streams (e.g., observations, reflections, plans) to condition future behavior; more broadly, continual learning studies how to retain knowledge without catastrophic forgetting (Parisi et al., 2019). In our setting, we instantiate memory for symbolic program synthesis: instead of storing dense representations or replay buffers (Blundell et al., 2016), we retain reusable symbolic rules and structural patterns discovered during factor mining, together with summary statistics that help avoid redundant regions of the search space.

From a theoretical perspective, meta-learning and meta-RL formalize ”learning to learn” by extracting transferable learning biases from prior tasks or episodes (Finn et al., 2017; Nichol et al., 2018; Gupta et al., 2018). Analogously, a memory-guided mining process can be seen as acquiring search priors: beyond producing individual factors, the system accumulates structural knowledge that shapes future exploration.

Beyond learning to call tools, systems work has advocated modular agent architectures that route subproblems to specialized models, external knowledge sources, and discrete reasoning modules (e.g., MRKL (Karpas et al., 2022); HuggingGPT (Shen et al., 2023)). In parallel, recent benchmarks and training pipelines study how to ground LLM outputs into executable API calls at scale and reduce tool hallucination, including API-Bank (Li et al., 2023), ToolLLM/ToolBench (Qin et al.,

Table 2: Operator categories in the factor mining skill (representative operators).

Category	Operators	Description
Arithmetic	Add, Sub, Mul, Div, Neg, Abs, Log, SignedPower, Power, Inv, Sqrt, Square, Exp, Tanh	Element-wise transformations
Statistical	Mean, Std, Var, Skew, Kurt, Med, Sum, Product	Rolling window statistics
Time-series	Delay, Delta, TsRank, TsMax, TsMin, TsArgMax, TsArgMin, TsDecay	Temporal pattern capture
Cross-sectional	CsRank, Scale	Cross-asset transforms
Smoothing	SMA, EMA, WMA	Trend extraction
Regression	Slope, Rsquare, Resi	Trend strength and residuals
Logical	IfElse, Greater, Less, GreaterEqual, LessEqual, And, Or, Eq, Ne	Conditional regime switching

2023), and Gorilla/APIBench (Patil et al., 2023). These works collectively emphasize the importance of separating high-level planning from reliable execution, and of representing tools/skills in a form that supports retrieval and continual updates.

Another line of research addresses long-horizon agent behavior under limited context windows by introducing explicit long-term memory managers. MemGPT (Packer et al., 2024) proposes OS-inspired hierarchical memory with controlled paging between fast and slow memory, while Voyager (Wang et al., 2023) demonstrates an open-ended embodied agent that accumulates an executable skill library and reuses it to generalize to new tasks. Such settings are often effectively non-stationary: as an agent acquires new tools/skills (or as an external library grows), the feasible action space and redundancy patterns evolve, motivating experience summarization mechanisms that capture reusable patterns and avoid repeatedly exploring known dead ends.

B Factor Expression and Operator Library

Each symbolic factor $\alpha \in \mathcal{P}$ is represented as a symbolic expression tree $T(\alpha)$ constructed from an operator library Ω. Following Kakushadze (2016), we define factors using a domain-specific language where:

- **Leaf nodes** are raw feature references: `$open`, `$high`, `$low`, `$close`, `$volume`, `$amt`, `$vwap`, `$returns`
- **Internal nodes** are operators from Ω with specified parameters

We curate a library of 60+ operators organized into categories (Table 2): (1) *Arithmetic*: Add, Sub, Mul, Div, Neg, Log, SignedPower; (2) *Statistical*: Mean, Std, Skew, Kurt over rolling windows; (3) *Time-series*: Delta, TsRank, TsMax, TsMin, Delay; (4) *Cross-sectional*: CsRank (percentile rank across assets at each time); (5) *Smoothing*: SMA, EMA, WMA (moving averages); (6) *Trend regression*: Slope, Rsquare, Resi (linear regression statistics); (7) *Logical*: IfElse, Greater, And, Or (conditional branching).

Note. Table 2 lists representative operators for readability; the full operator registry in the FactorMiner skill contains 60+ typed operators.

C Admission Criteria and Factor Replacement

A symbolic factor α is admitted to the library $\mathcal{L}$ if it satisfies:

$$\text{IC}(\alpha) \geq \tau_{\text{IC}} \quad \wedge \quad \max_{g \in \mathcal{L}} |\rho(\alpha, g)| < \theta \tag{10}$$

where τ_{IC} and θ are the IC and correlation thresholds (default: $\tau_{\text{IC}} = 0.04$ and $\theta = 0.5$ for A-share mining unless otherwise specified).

Additionally, we introduce a **factor replacement mechanism** for high-quality factors that would otherwise be rejected due to correlation. If a new factor α satisfies:

$$\text{IC}(\alpha) \geq 0.10 \quad \wedge \quad \text{IC}(\alpha) \geq 1.3 \times \text{IC}(g) \quad \wedge \quad |\{g \in \mathcal{L} : \rho(\alpha, g) > \theta\}| = 1 \tag{11}$$

then α replaces the single correlated factor g in $\mathcal{L}$. This mechanism allows the library to evolve toward higher quality while maintaining the correlation constraint.

D Implementation Details

For each method, we first construct a fixed Top-40 factor set by ranking candidates on CSI500 using the **training year 2024**: we filter admitted factors by thresholding |IC| and |ICIR| (stock thresholds: $|\text{IC}| \geq 0.05$, $|\text{ICIR}| \geq 0.5$), select up to 40 from the admitted set by descending |IC|, and if fewer than 40 are admitted, fill the remaining slots with the next-best valid candidates by |IC|. We then freeze this Top-40 set (selected once on CSI500) and evaluate it on CSI500/CSI1000/HS300/Crypto for the full-year 2025 out-of-sample comparison in Table 1, to test generalization across time (2024 $\rightarrow$ 2025), universes, and markets. Alpha101 factors were originally designed for daily frequency data, we adapt them for 10-minute frequency by optimizing window parameters. For each of the Alpha101 factors, we generate up to 10 parameter variants and select the best-performing configuration.

FactorMiner is implemented in Python, utilizing NumPy and CuPy for GPU-accelerated operator computation. This achieves up to 26$\times$ speedup; see Section E. A 40-worker multiprocessing pool handles parallel candidate evaluation. The experience memory is maintained as a structured, human-readable knowledge base. We employ Gemini 3.0 Flash as the LLM backbone for symbolic program synthesis. The mining process follows an iterative curriculum using the default admission thresholds in Section C, adjusted when needed.

E Computational Efficiency

To enable rapid iteration, FactorMiner leverages three acceleration techniques:

GPU-accelerated operators. Core operators (CsRank, TsRank, rolling statistics) are implemented using PyTorch, achieving 6–26$\times$ speedup over CPU implementations (Table 3).

Multi-process parallelization. Factor evaluation is inherently parallel across candidates. We use a worker pool to evaluate batches concurrently, with each worker handling one candidate factor.

C-compiled numerical operations. Low-level numerical operations use optimized C implementations (e.g., bottleneck library) for efficient rolling window computations.

Table 3: GPU acceleration for core operators (single A100 GPU).

Operator	CPU (ms)	GPU (ms)	Speedup
CsRank	93	3.6	26$\times$
TsRank	97	6.0	17$\times$
Rolling Corr	76	11	6.8$\times$
Rolling Std	13	3.4	3.7$\times$
TsDecay	45	5.0	9$\times$

F Factor Combination and Selection

After library construction, we provide three combination strategies and three selection methods:

Combination strategies.

- **Equal-weight**: Simple average of all factor values.
- **IC-weighted**: Weight factors by their historical IC.
- **Orthogonal**: Gram-Schmidt orthogonalization before averaging.

Selection methods.

- **Lasso**: L1-regularized linear regression to identify sparse factor subsets.

- **Forward stepwise**: Greedy selection maximizing combined ICIR.
- **XGBoost**: Gradient boosting to capture nonlinear factor interactions.

G DISCUSSION

High-frequency markets can be treated as a data-rich complex adaptive system, and each formulaic factor can be viewed as an explicit, falsifiable hypothesis about market microstructure. Beyond predictive performance, FactorMiner yields a curated library of 110 interpretable high-frequency alpha factors together with a standardized evaluation protocol, enabling reproducible hypothesis testing, mechanistic inspection, and cross-market transfer studies.

Experience memory as continual learning. Our results confirm that memory-guided, skill-based agents can efficiently scale neural-symbolic program synthesis while retaining interpretability. The experience memory serves as a form of institutional knowledge that accumulates across mining sessions, enabling meta-learning: the system learns not just individual factors, but how to search more effectively. Key insights extracted from our memory (Appendix H) include: **successful patterns**, such as higher-order moment regimes via Skew/Kurt, trend-regression adaptivity via Rsquare/Slope/Resi, and amount-efficiency interactions, that consistently yield high-IC yet low-correlation factors; **failure patterns**, such as VWAP-deviation variants, standardized returns, and simple Delta reversals, that tend to be highly correlated with existing factors and should be avoided; and the **Correlation Red Sea** phenomenon, where discovering new orthogonal factors becomes increasingly difficult as the library grows without memory guidance.

H EXPERIENCE MEMORY: RECOMMENDED DIRECTIONS

The following table summarizes successful mining patterns extracted from our experience memory. These patterns have consistently yielded factors that pass the default admission criteria (Section C).

Table 4: Recommended mining directions from experience memory.

Pattern	Description	Success Rate
Higher Moment Regimes	Use Skew/Kurt as IfElse conditions to identify extreme asymmetric or fat-tail environments for reversal signals.	High
PV Corr Interaction	Combine price-volume correlation (Corr) with amount efficiency or trend operators to capture volume-price coordination.	High
Robust Efficiency	Use median (Med) and other robust statistics to smooth amount efficiency, filtering extreme noise.	High
Smoothed Efficiency Rank	Apply time-series smoothing (EMA) to amount efficiency before cross-sectional ranking.	High
Trend Regression Adaptive	Use Rsquare/Slope/Resi operators for adaptive trend regression. High $R^2 \rightarrow$ slope reversal; Low $R^2 \rightarrow$ residual reversal.	High
Logical 'Or' Extreme Regimes	Use Or operator to integrate multiple extreme indicators (volume/price) as environment switching conditions.	High
Kurtosis Regime	Use kurtosis to identify fat-tail environments and adaptively adjust reversal windows.	High
Amt Efficiency Rank Interaction	Combine amount efficiency time-series rank with other statistical features (e.g., kurtosis).	Medium

The recommended directions reveal several key insights about effective factor design in high-frequency intraday settings:

(1) Higher-order moments as regime indicators. Skewness and kurtosis emerge as powerful tools for identifying market regimes. When used as conditions in IfElse branching, they enable factors to adapt their logic based on distributional characteristics. For example, Factor 095 (Higher_Moment_Regime_Switch) achieves IC = 0.062 by switching between amount efficiency and slope reversal based on skewness thresholds. This pattern suggests that extreme distributional environments (high skew or fat tails) signal different underlying market dynamics requiring distinct trading logic.

(2) Trend regression operators provide orthogonal signals. The Rsquare/Slope/Resi operator family, introduced in our expanded operator library, enables capturing trend reliability and deviations. Factors using these operators (080–086) consistently achieve low correlation with existing VWAP-based factors while maintaining strong IC. The adaptive logic—using slope reversal when R^2 is high (indicating reliable trend) versus residual reversal when R^2 is low—aligns with financial intuition about trend-following vs. mean-reversion regimes.

(3) Amount efficiency as an underexplored dimension. Combining returns with transaction amount (Returns/Amount) produces signals orthogonal to pure price-based factors. The success of factors 092–099 demonstrates that this dimension, when properly smoothed and ranked, captures liquidity-adjusted momentum that complements traditional price signals.

I Experience Memory: Forbidden Directions

The following table summarizes patterns that consistently lead to high correlation with existing factors and should be avoided.

Table 5: Forbidden mining directions (high correlation risk).

Direction	Correlated Factors	Correlation
Standardized Returns/Amount	006, 008, 009	>0.6
VWAP Deviation variants	006, 009, 012, 013, 016	>0.5
Mean Reversion / Price Deviation	001, 002	>0.5
Simple Delta Reversal	023, 024	>0.5
Close-Position Location	028, 044	0.87+
Volatility + Price Position Branch	046, 064	>0.6
High R^2 Trend Following	081, 083	>0.6
Rsquare Weighted Momentum	002, 023	>0.7
WMA/EMA Smoothed Efficiency	092	>0.9

The forbidden directions encode critical "negative knowledge" that prevents wasteful exploration:

(1) The VWAP cluster dominance. Factor 006 (VWAP Deviation) and its variants form the most densely populated region of our correlation space. Any new factor involving close-to-VWAP distances, standardized by volatility or transaction volume, will almost certainly correlate > 0.5 with this cluster. This explains why 40+ candidate factors were rejected during mining—they inadvertently rediscovered VWAP logic in different algebraic forms.

(2) The "Correlation Red Sea" phenomenon. As the library grew beyond 70 factors, we observed that high-IC candidates (IC > 0.08) increasingly correlated with existing factors. For example, in Batch 122, factors like SignedPower_Returns (IC = 0.10) and High_Log_Ratio (IC = 0.105) were rejected due to correlations of 0.69–0.74 with factors 083 and 029. This suggests that the "easy" signal space has been exhausted, and future mining must target genuinely novel dimensions.

(3) Mathematical equivalence traps. Several forbidden directions represent mathematically equivalent or highly similar computations. For instance, the two expressions below appear structurally different but capture essentially the same VWAP deviation dynamic:

```
Neg(TsRank(Div(Sub(close,vwap),vwap),24))
```

```
Neg(CsRank(Delta(Sub(close,vwap),3)))
```

Recording these equivalences prevents the agent from "rediscovering" the same signal.

J Mining Log Analysis

This section provides detailed analysis of representative mining batches, illustrating the Ralph Loop in action.

J.1 BATCH 126: MEDIAN/LOG/RESI EXPLORATION (40 WORKERS)

Setup. 40 parallel workers evaluated 40 candidate factors generated from Median/Log/Resi combinations. IC threshold: 0.05; correlation threshold: 0.5.

Stage 1 Results. 7 candidates passed IC screening:

- Open_Close_ResiCross (IC = 0.071)
- Median_Adjusted_Returns (IC = 0.068)
- High_Log_CloseSwitch (IC = 0.062)
- Open_Resi_Skew_Hybrid (IC = 0.062)
- Median_LogSwap (IC = 0.057)
- Low_Log_RangeSwitch (IC = 0.089)
- Low_Log_SignedPower (IC = 0.101)

Stage 2 Bottleneck. 5 of 7 candidates were blocked by correlation with existing factors:

- Low_Log_SignedPower: corr = 0.74 with Factor 028 (Close-Low $\times$ Volume)
- Median_Adjusted_Returns: corr = 0.68 with Factor 083 (Rsquare Filtered Momentum)
- High_Log_CloseSwitch: corr = 0.61 with Factor 101 (Median Returns Switch)
- Open_Close_ResiCross: corr = 0.58 with Factor 006 (VWAP Deviation)
- Low_Log_RangeSwitch: corr = 0.52 with Factor 028

Final Admission. 2 factors admitted:

- **Factor 103** (Open_Resi_Skew_Hybrid): IC = 0.062, ICIR = 0.62, max_corr = 0.41
- **Factor 104** (Median_LogSwap): IC = 0.057, ICIR = 0.59, max_corr = 0.49

Lessons Learned.

1. **High IC $\neq$ admission**: The highest-IC candidate (Low_Log_SignedPower, IC = 0.101) was rejected due to correlation.
2. **Residual combinations effective**: The Resi operator combined with Skew branching produced orthogonal signals.
3. **Memory update**: Added "Low-level log/close combinations" to forbidden directions (correlated with 028).

K FACTOR COMBINATION DETAILED ANALYSIS

This section provides in-depth analysis of the three factor combination strategies.

Table 6: Detailed combination analysis (110 factors).

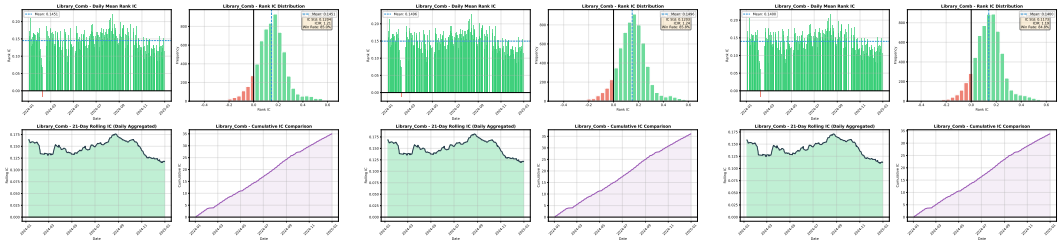
Metric	Equal-Weight	IC-Weighted	Orthogonal
IC Mean	0.1451	0.1496	0.1400
ICIR	1.2053	1.2430	1.1933
IC Win Rate	85.0%	85.8%	84.8%
Q1 Return	−0.0422%	−0.0441%	−0.0406%
Q5 Return	0.0603%	0.0619%	0.0564%
L-S Return	0.0513%	0.0531%	0.0486%
L-S Cumulative	23.72	26.67	19.84
Monotonicity	1.0	1.0	1.0
Avg Turnover	20.14%	20.43%	19.67%

Figures Figures 3 and 4 visualize the temporal IC stability and return profiles of the three combination strategies.

(1) IC-weighted combination achieves highest cumulative returns. By weighting factors proportionally to their historical IC, the IC-weighted method concentrates exposure on the most predictive signals. The 12.4% improvement in cumulative returns (26.67 vs. 23.72) over equal-weight comes at minimal cost in turnover (+1.4%).

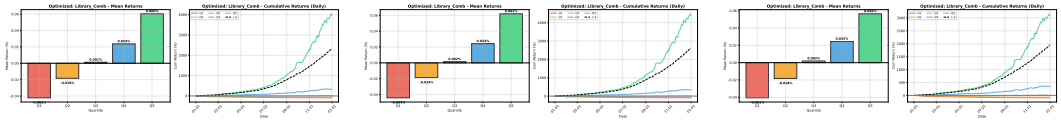
(2) Orthogonal combination underperforms expectations. Despite its theoretical appeal (removing redundant information), orthogonalization actually reduces both IC and cumulative returns. This counterintuitive result suggests that the correlation structure among factors contains useful information—highly correlated factors may be capturing the same underlying signal with different noise characteristics, and averaging them provides noise reduction benefits.

(3) Perfect monotonicity across all methods. All three methods achieve monotonicity = 1.0, indicating perfect rank ordering of quintile returns. This confirms that the combined factors produce consistent cross-sectional rankings, essential for practical portfolio construction.



(a) Equal-weight IC (b) IC-weighted IC (c) Orthogonal IC

Figure 3: IC time-series analysis for three combination methods. All methods show stable positive IC throughout the evaluation period, with IC-weighted exhibiting slightly higher peaks.



(a) Equal-weight quintiles (b) IC-weighted quintiles (c) Orthogonal quintiles

Figure 4: Quantile returns for three combination methods. All methods show perfect monotonicity with Q5 (highest factor value) consistently outperforming Q1 (lowest).

L Factor Selection Detailed Analysis

This section provides comprehensive analysis of the three factor selection methods.

Table 7: Detailed selection analysis.

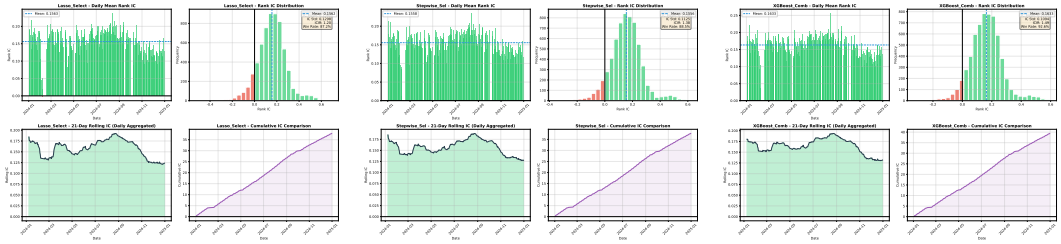
Metric	Lasso	Stepwise	XGBoost
# Selected Factors	8	18	110 (all)
IC Mean	0.1562	0.1556	0.1633
ICIR	1.2039	1.3827	1.4929
IC Win Rate	87.2%	88.5%	92.6%
Q1 Return	−0.0604%	−0.0485%	−0.0609%
Q5 Return	0.0678%	0.0625%	0.0804%
L-S Return	0.0642%	0.0556%	0.0708%
L-S Cumulative	54.69	31.51	82.63
Monotonicity	1.0	1.0	1.0
Avg Turnover	19.92%	20.02%	19.32%

Interpretation. Figures Figures 5 and 6 summarize the IC stability and return profiles for the three selection methods.

(1) XGBoost achieves superior performance through nonlinear interactions. The 24% improvement in ICIR (1.49 vs. 1.20) and 51% improvement in cumulative returns (82.6 vs. 54.7) over Lasso demonstrates the value of capturing nonlinear factor interactions. The regime-switching factors (IfElse-based) likely contribute significantly here, as tree-based models can naturally exploit their conditional logic.

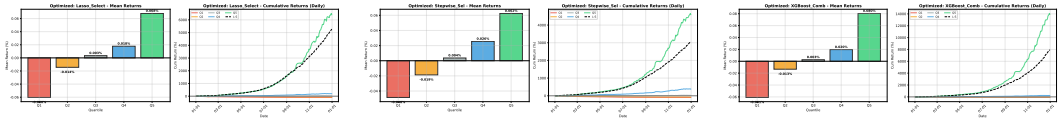
(2) Lasso achieves extreme sparsity. With only 8 factors, Lasso captures 95% of the IC improvement achievable by the full library. This suggests that most predictive information is concentrated in a small subset of factors, consistent with the Pareto principle observed in many financial applications.

(3) Stepwise provides interpretable middle ground. Forward stepwise selection balances sparsity (18 factors) with performance (ICIR = 1.38). The greedy selection trajectory (Table 9) reveals the diminishing returns of adding factors beyond the first 10.



(a) Lasso IC analysis (b) Stepwise IC analysis (c) XGBoost IC analysis

Figure 5: IC time-series analysis for three selection methods. XGBoost shows the most stable IC with highest win rate (92.6%).



(a) Lasso quintile returns (b) Stepwise quintile returns (c) XGBoost quintile returns

Figure 6: Quantile returns for three selection methods. XGBoost shows the widest Q5-Q1 spread, indicating strongest discriminative power.

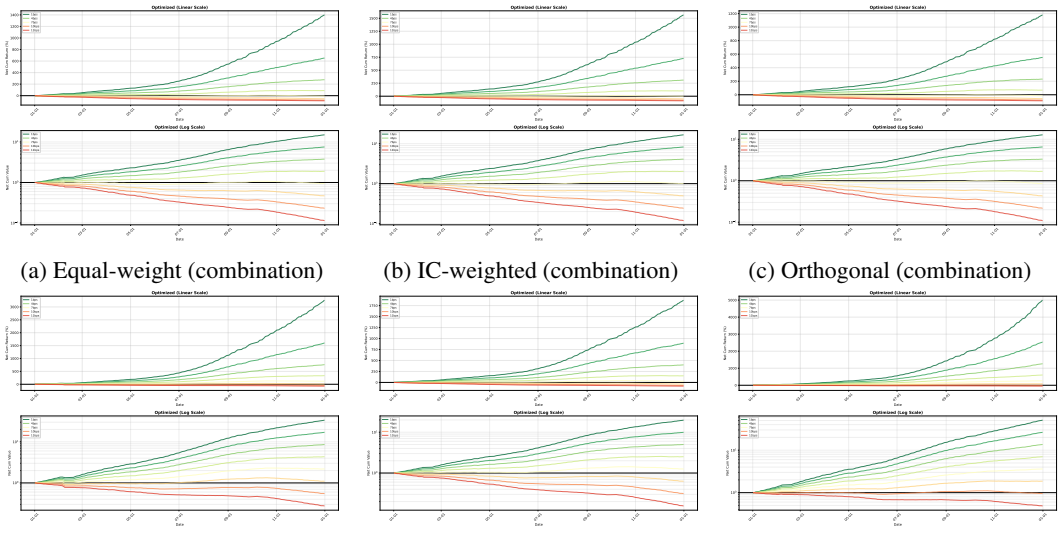
M Cost Pressure Stress Test

We further evaluate the robustness of downstream combination and selection methods under transaction cost pressure. Specifically, we test five transaction cost settings (1, 4, 7, 10, and 11 bps) and report the resulting net cumulative performance trajectories. Figure 7 summarizes the cost-pressure analyses for three factor combination strategies and three factor selection strategies. For ease of comparison across cost regimes and performance scales, each panel is shown with both a linear y-axis and a log-scale y-axis.

N Lasso Selected Factors Analysis

Interpretation. The Lasso-selected factors represent a diverse minimal spanning set of the factor space:

Factor 006 (VWAP Deviation) dominates with coefficient 4.5x larger than the second factor. This confirms VWAP as the most powerful single signal for intraday reversal prediction.



(a) Equal-weight (combination) (b) IC-weighted (combination) (c) Orthogonal (combination)

(d) Lasso (selection) (e) Stepwise (selection) (f) XGBoost (selection)

Figure 7: Cost pressure stress tests for three factor combination methods (Equal-weight, IC-weighted, Orthogonal; top row) and three factor selection methods (Lasso, Stepwise, XGBoost; bottom row). For each method, we evaluate performance under five transaction cost settings: 1, 4, 7, 10, and 11 bps. Each panel reports the net cumulative performance over time using both a linear and a log-scale y-axis to facilitate comparison across cost regimes and performance scales.

Table 8: Factors selected by Lasso (8 factors, sorted by coefficient magnitude).

ID	Name	Coefficient	Role
006	VWAP Deviation	3.23×10^{-4}	Core anchor
002	EMA Deviation	7.23×10^{-5}	Trend baseline
079	Regime_Vol_Range_Pos	2.58×10^{-5}	Regime switch
040	PricePos_Skew_Slope	1.14×10^{-5}	Distribution
011	Price-VWAP Momentum	8.18×10^{-6}	VWAP variant
045	Kurtosis-Regime Range	7.18×10^{-6}	Higher moment
009	Vol-Filtered Return	4.28×10^{-6}	Risk filter
022	Lower-Shadow Ratio	2.59×10^{-6}	Candlestick

Factor 002 (EMA Deviation) provides complementary trend information using exponential smoothing rather than volume-weighted averaging.

Factors 079, 045 (Regime-switching) introduce conditional logic that adapts to market state, providing nonlinear signal that linear Lasso partially captures through coefficient weighting.

Factors 040, 022 (Distribution/Candlestick) represent orthogonal signal sources—higher-order moments and price action patterns—that Lasso identifies as non-redundant.

O Stepwise Selection Trajectory

Interpretation. The stepwise trajectory reveals the structure of marginal contribution:

Steps 1–5: Rapid ICIR improvement (+0.133). The first five factors (006, 046, 079, 044, 041) provide the bulk of predictive improvement. These factors span different logical categories (VWAP, regime-switching, higher moments), confirming the value of diversity.

Steps 6–11: Diminishing returns. Adding factors 6–11 improves ICIR by only +0.020 total, despite each factor having respectable individual IC (> 0.07). This plateau indicates that the marginal information content decreases rapidly after the core signal space is covered.

Table 9: Forward stepwise selection trajectory (all 18 steps).

Step	Added Factor	Indiv. IC	Comb. IC	ICIR	ΔICIR
1	006 VWAP Deviation	0.129	0.150	1.163	–
2	046 Vol-Regime Reversal	0.109	0.145	1.184	+0.021
3	079 Regime Vol-Range	0.103	0.145	1.234	+0.050
4	044 Kurtotic Vol Intensity	0.100	0.145	1.271	+0.037
5	041 Price Range Skew	0.097	0.145	1.296	+0.025
6	040 PricePos Skew Slope	0.095	0.145	1.299	+0.003
7	013 Illiquidity VWAP	0.092	0.148	1.304	+0.005
8	107 PricePos Skew VolDelta	0.090	0.147	1.304	+0.000
9	004 High-Vol Weakness	0.082	0.150	1.308	+0.004
10	011 Price-VWAP Momentum	0.078	0.151	1.315	+0.007
11	074 Range Pos Vol Skew	0.073	0.150	1.316	+0.001
12	026 PV Covariance Long	0.066	0.153	1.357	+0.041
13	054 Amount Regime	0.063	0.153	1.366	+0.009
14	016 VWAP Vol Skew Switch	0.063	0.154	1.370	+0.004
15	005 Range-Pos Vol-Cov	0.055	0.154	1.370	+0.000
16	022 Lower-Shadow Ratio	0.055	0.154	1.378	+0.008
17	051 High Med Volume Switch	0.049	0.155	1.381	+0.003
18	076 Amt Velocity Volatility	0.047	0.156	1.383	+0.002

Step 12: Second-wave improvement. Factor 026 (PV Covariance Long) provides an unexpected +0.041 ICIR boost, suggesting that price-volume covariance captures a distinct signal dimension not well-represented by the first 11 factors.

Steps 13–18: Final refinement. The last six factors add only +0.026 ICIR total, confirming convergence to the ICIR ceiling.

P XGBoost Feature Importance Analysis

Table 10: XGBoost feature importance (top 20 factors).

ID	Name	Import.	Category
006	VWAP Deviation	6.04%	VWAP
061	Alpha101_12_Modern	4.06%	Classic
023	Normalized-Momentum TsRank	3.59%	Momentum
068	Skewness_Regime_PV_Div	3.55%	Regime
028	Close-Low × Volume	3.03%	Price range
070	Price_Pos_Vol_Interaction	2.27%	Interaction
057	TsRank_PV_Divergence	2.26%	Divergence
029	High-Close × Volume	2.15%	Price range
018	Range-Position Vol Regime	1.62%	Range
045	Kurtosis-Regime Range	1.57%	Higher moment
048	Vol-Price Rank Divergence	1.47%	Divergence
104	Median_LogSwap	1.46%	Median
053	Alpha101_1_V	1.44%	Classic
101	Median_Returns_Switch	1.41%	Median
004	High-Vol Relative Weakness	1.34%	Volume
055	Skew_Open_Close_Product	1.24%	Distribution
019	VWAP-Gap Regime Range	1.20%	VWAP
092	Amt_Efficiency_EMA_Smooth	1.16%	Efficiency
073	Amt_Velocity_Regime	1.16%	Efficiency
043	Skewed Volume Momentum	1.11%	Distribution

Interpretation. XGBoost importance reveals the nonlinear signal structure:

(1) VWAP remains dominant but not overwhelming. Factor 006 contributes 6.04% importance—significant but not dominant. In contrast, Lasso assigns 006 a coefficient 4.5x larger than the second

factor. This suggests XGBoost extracts value from many factors simultaneously rather than relying on a single anchor.

(2) Regime-switching factors are highly valued. Factors 068 (Skewness Regime) and 070 (Price-Vol Interaction) rank 4th and 6th despite lower individual IC than factors 046 or 079. This indicates XGBoost effectively exploits their conditional logic structure.

(3) Classic Alpha101 factors remain relevant. Factors 061 (Alpha101_12_Modern) and 053 (Alpha101_1_V) rank 2nd and 13th, demonstrating that modernized versions of classic formulas contribute unique signals even in a 110-factor library.

(4) Long-tail distribution of importance. The top 20 factors contribute 43.8% of total importance, while the remaining 90 factors contribute 56.2%. This confirms that XGBoost extracts value broadly across the library, justifying the construction of a diverse factor set.

Q Individual Factor Tear-Sheet Example

We present the complete evaluation report for Factor 046 (Volatility-Regime Reversal Divergence), one of the top-performing factors (see Figure 8).

Factor Profile.

- **ID**: 046
- **Name**: Volatility-Regime Reversal Divergence
- **Formula**: `IfElse(Greater(Std($returns, 12), Mean(Std($returns, 12), 48)), Neg(CsRank(Delta($close, 3))), Neg(CsRank(Div(Sub($close, $low), Add(Sub($high, $low), 0.0001)))))`
- **Category**: Regime-switching

Metric	Value
IC Mean	0.1087
ICIR	0.9422
IC Win Rate (daily)	80.1%
Q1 Return	−0.0380%
Q5 Return	0.0402%
L-S Return	0.0390%
L-S Cumulative	10.57
Monotonicity	1.0
Avg Turnover	15.69%

Performance Metrics. **Note.** IC is computed at 10-minute frequency. The win rate is reported at daily granularity by aggregating the 10-minute IC values within each day (daily mean) and counting the fraction of days with positive daily IC.

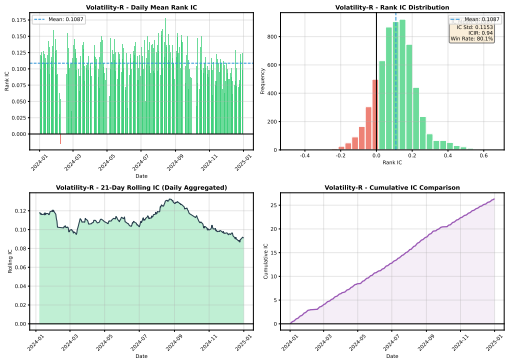
Financial Logic. This factor implements adaptive reversal logic based on volatility regime:

- **High volatility regime** (Std $>$ 48-period mean): Use volume-price divergence signal. In turbulent markets, divergence between volume momentum and price momentum signals exhaustion.
- **Low volatility regime**: Use VWAP deviation normalized by volatility. In calm markets, simple mean-reversion to VWAP is more reliable.

R Library Correlation Heatmap

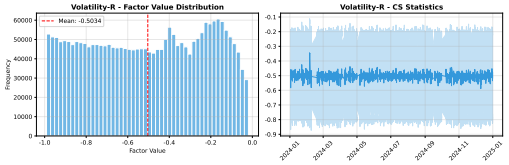
S Full Factor Library (110 factors)

We provide the complete factor library (ID, name, and formula).

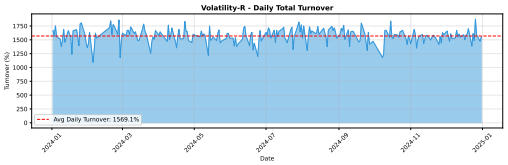


(a) IC time-series analysis

(b) Quantile returns



(c) Factor value distribution



(d) Turnover analysis

Figure 8: Complete tear-sheet for Factor 046 (Volatility-Regime Reversal Divergence). The factor shows stable IC with 80% daily win rate, perfect monotonicity in quintile returns, and moderate turnover (15.7%).

ID	Name	Formula
001	Intraday Range Position	`Neg(CsRank(Div(Sub($close, Min($close, 48)), Add(Sub(Max($close, 48), Min($close, 48)), 1e-8))))`
002	EMA Deviation	`Neg(Div(Sub($close, EMA($close, 10)), EMA($close, 10)))`
003	Vol-VWAP Divergence	`Sub(CsRank(Delta($volume, 1)), CsRank(Div(Sub($close, $vwap), $vwap)))`
004	High-Volume Relative Weakness	`Mul(CsRank(Div($volume, Mean($volume, 24))), CsRank(Neg($returns)))`
005	Range-Position Vol-Volume Cov Reversal	`Neg(Mul(TsRank(Div(Sub($close, TsMin($close, 12)), Add(Sub(TsMax($close, 12), TsMin($close, 12)), 1e-6)), 12), CsRank(Abs(Cov($returns, $volume, 12)))))`
006	VWAP Deviation	`Neg(Div(Sub($close, $vwap), $vwap))`
007	Price-Volume Mean Reversal	`Neg(Sub(CsRank(Div(Sub($close, $low), Add(Sub($high, $low), 0.0001))), CsRank(Delta(EMA($volume, 5), 5))))`
008	Normalized Money Strength Reversal	`Neg(Mul(CsRank(Div($returns, Add(Std($returns, 12), 0.0001))), CsRank(Div($volume, Mean($volume, 12)))))`
009	Volatility-Filtered Return Reversal	`Neg(Mul(CsRank($returns), CsRank(Std($returns, 12))))`
010	Price-Volume Divergence Reversal	`Neg(CsRank(Sub(CsRank(Div($close, Delay($close, 5))), CsRank(Div($volume, Delay($volume, 5))))))`
011	Price-VWAP Momentum Overextension	`Neg(Sub(CsRank(Delta($close, 6)), CsRank(Delta($vwap, 6))))`
012	Volume-Augmented VWAP Reversal	`Mul(CsRank(Neg(Div(Sub($close, $vwap), $vwap))), CsRank(Div($volume, EMA($volume, 12))))`
013	Illiquidity-Augmented VWAP Reversal	`Mul(CsRank(Neg(Div(Sub($close, $vwap), $vwap))), Sub(1, CsRank($volume)))`
014	Resilience-Momentum Blend Reversal	`Mul(Add(Mul(CsRank(Neg(Div(Sub($close, $vwap), $vwap))), 0.6), Mul(CsRank(Delta(Neg(Div(Sub($close, $vwap), $vwap)), 3)), 0.4)), CsRank(Delta(Div(Mean(Div(Abs($returns), Add($volume, 1)), 24), Add(Div(Abs($returns), Add($volume, 1)), 1e-6)), 3)))`
015	Volatility-Balanced Momentum Reversal	`Mul(CsRank(Neg(Delta($returns, 3))), Mul(Sub(1, CsRank(Std($returns, 12))), CsRank(Std($returns, 12))))`
016	VWAP-Deviation Acceleration Reversal	`Neg(Mul(CsRank(Delta(Div(Sub($close, $vwap), $vwap), 3)), CsRank(Div($volume, EMA($volume, 12)))))`
017	VWAP-Acceleration Volatility Reversal	`Neg(Mul(TsRank(Delta(Div(Sub($close, $vwap), $vwap), 3), 12), TsRank(Std($returns, 12), 12)))`
018	Range-Position Volume Regime Reversal	`Neg(Mul(TsRank(Div(Sub($close, TsMin($close, 12)), Add(Sub(TsMax($close, 12), TsMin($close, 12)), 1e-6)), 12), TsRank(Div($volume, EMA($volume, 12)), 12)))`
019	VWAP-Gap Regime Range Reversal	`IfElse(Greater(Abs(Div(Sub($close, $vwap), $vwap)), 0.01), Neg(CsRank(Sub($close, TsMax($close, 12)))), Neg(CsRank(Sub($close, TsMin($close, 12)))))`

(continued)

ID	Name	Formula
020	Volume-Price Delta Divergence TsRank	`TsRank(Sub(CsRank(Delta($volume, 3)), CsRank(Delta($close, 3))), 18)`
021	Volume-Shock VWAP-Momentum Reversal	`Neg(Mul(TsRank(Delta(Div($volume, EMA($volume, 12)), 3), 12), TsRank(Delta(Div(Sub($close, $vwap), $vwap), 3), 12)))`
022	Lower-Shadow Ratio Reversal	`Neg(CsRank(Div(Sub(Min2($open, $close), $low), Add(Sub($high, $low), 0.0001))))`
023	Normalized-Momentum TsRank Reversal	`Neg(TsRank(Div(Delta($close, 3), Mean(Abs(Delta($close, 3)), 12)), 12))`
024	Open-Close Momentum Divergence	`Sub(TsRank(Delta($open, 6), 12), TsRank(Delta($close, 6), 12))`
025	Open-Close Momentum Divergence V2	`Sub(TsRank(Delta($open, 9), 18), TsRank(Delta($close, 9), 18))`
026	Price-Volume Covariance Position Long	`Neg(Mul(TsRank(Cov(TsRank($close, 18), TsRank($volume, 18), 10), 18), TsRank(Div(Sub($close, $low), Sub($high, $low)), 18)))`
027	Open-Volume Covariance Position	`Neg(Mul(TsRank(Cov(TsRank($open, 12), TsRank($volume, 12), 5), 12), TsRank(Div(Sub($close, $low), Sub($high, $low)), 12)))`
028	Close-Low × Volume Ratio	`Neg(Mul(Sub($close, $low), Div($volume, Mean($volume, 12))))`
029	High-Close × Volume Ratio	`Mul(Sub($high, $close), Div($volume, Mean($volume, 12)))`
030	Delta(High-Close)3 × Volume Ratio	`Mul(Delta(Sub($high, $close), 3), Div($volume, Mean($volume, 12)))`
031	Price Trend Conditional Strategy	`IfElse(Greater(Delta($close, 1), 0), Neg(TsRank(Delta($close, 3), 12)), TsRank(Delta($close, 3), 12))`
032	Returns-Volume Delta Divergence	`Neg(TsRank(Sub(CsRank(Delta($returns, 2)), CsRank(Delta($volume, 2))), 12))`
033	Volume-Range Correlation Shock	`Neg(Mul(TsRank(Corr(TsRank(Div($volume, Mean($volume, 24)), 12), TsRank(Sub($high, $low), 12), 12), 24), CsRank(Delta($close, 2))))`
034	Price Trend Conditional Momentum V2	`IfElse(Greater(Delta($close, 2), 0), Neg(TsRank(Delta($close, 5), 15)), TsRank(Delta($close, 5), 15))`
035	High-Low Expansion vs Close Accel	`Neg(Mul(TsRank(Delta(Sub($high, $low), 2), 12), CsRank(Delta($close, 2))))`
036	Volatility Switch Candle Reversal	`IfElse(Greater(Std($returns, 12), Mean(Std($returns, 12), 48)), Neg(CsRank(Div(Sub(Min2($open, $close), $low), Add(Sub($high, $low), 0.0001)))), Neg(CsRank($returns)))`
037	Price-Volume Momentum Synchrony	`Neg(TsRank(Mul(CsRank(Delta($close, 3)), CsRank(Delta($volume, 3))), 18))`
038	Vol-Regime Conditional Divergence V2	`IfElse(Greater(Std($returns, 12), Mean(Std($returns, 12), 48)), Neg(TsRank(Sub(CsRank(Delta($volume, 3)), CsRank(Delta($close, 3))), 18)), Neg(TsRank(Div(Sub($close, $vwap), Add(Std($returns, 24), 0.0001)), 12)))`
039	Kurtosis-Filtered Momentum	`Neg(Mul(CsRank(Delta($close, 5)), CsRank(Kurt($returns, 24))))`
040	Price Intensity Skewness Blend	`Neg(Mul(CsRank(Div(Sub($close, $low), Add(Sub($high, $low), 0.0001))), CsRank(Skew($returns, 24))))`
041	Price Range Skewness Interaction	`Mul(CsRank(Div(Sub($high, $close), Add(Sub($high, $low), 1e-8))), CsRank(Neg(Skew($returns, 24))))`
042	Regime-Switching Skew Factor	`IfElse(Greater(Abs(Skew($returns, 24)), 1.0), Neg(CsRank($returns)), Neg(CsRank(Skew($returns, 24))))`
043	Skewed Volume Momentum	`Neg(TsRank(Mul(CsRank(Skew($volume, 24)), CsRank(Delta($close, 3))), 12))`
044	Kurtotic Volume Intensity	`Neg(Mul(CsRank(Kurt($volume, 24)), CsRank(Div(Sub($close, $low), Add(Sub($high, $low), 0.0001)))))`
045	Kurtosis-Regime Range Reversal	`IfElse(Greater(Kurt($returns, 24), 3.0), Neg(CsRank(Div(Sub($high, $close), Add(Sub($high, $low), 0.0001)))), Neg(CsRank(Div(Sub($close, $low), Add(Sub($high, $low), 0.0001)))))`
046	Volatility-Regime Reversal Divergence	`IfElse(Greater(Std($returns, 12), Mean(Std($returns, 12), 48)), Neg(CsRank(Delta($close, 3))), Neg(CsRank(Div(Sub($close, $low), Add(Sub($high, $low), 0.0001)))))`
047	Volume-weighted Alpha101-1 variant V2	`Neg(Mul(CsRank(TsRank(Delta(Log(Add($volume, 1)), 3), 6)), CsRank(Div(Delta($close, 6), $close))))`
048	Volatility-Price Rank Divergence TsRank V2	`TsRank(Sub(CsRank(Std($returns, 12)), CsRank(Delta($close, 6))), 18)`
049	Regime_C0_F10_F21	`IfElse(Greater(Skew($returns, 24), 0.5), CsRank(Delta($volume, 3)), Neg(CsRank(Std($returns, 12))))`
050	Regime_C0_F11_F20	`IfElse(Greater(Skew($returns, 24), 0.5), Neg(CsRank(Div(Sub($close, $low), Add(Sub($high, $low), 1e-6)))), Neg(CsRank(Delta($close, 6))))`
051	Regime_C1_F11_F21	`IfElse(Less(Skew($returns, 24), -0.5), CsRank(Delta($close, 5)), Neg(CsRank(Std($returns, 12))))`
052	Regime_C1_F12_F20	`IfElse(Less(Skew($returns, 24), -0.5), Neg(TsRank($returns, 12)), Neg(CsRank(Delta($close, 6))))`
053	Alpha101_1_V	`Neg(CsRank(TsArgMax(SignedPower(IfElse(Less($returns, 0), Std($returns, 20), $close), 2), 5)))`
054	Amount_Regime_Reversal	`IfElse(Greater($amt, Mean($amt, 24)), CsRank(Delta($close, 3)), Neg(CsRank(Div(Sub($close, $vwap), $vwap))))`
055	Return_Correlation_Regime	`IfElse(Greater(Corr($close, $volume, 12), 0.5), Neg(CsRank($returns)), CsRank(Delta($volume, 3)))`

(continued)

ID	Name	Formula
056	Kurtosis_Regime_Amount_Efficiency	IfElse(Greater(Kurt($returns, 24), 3.0), Neg(CsRank(Div($amt, Add($volume, 1e-6)))), Neg(CsRank(Delta($close, 3))))
057	TsRank_Price_Volume_Momentum_Divergence	Neg(Sub(TsRank(Delta($close, 6), 24), TsRank(Delta($volume, 6), 24)))
058	Regime_Amt_Efficiency_Switch_V2	IfElse(Greater($amt, Mean($amt, 48)), Neg(CsRank(Div($amt, Add($volume, 1e-6)))), Neg(TsRank($returns, 12)))
059	Regime_Triple_Vol_Amt_Skew	IfElse(Greater(Std($returns, 12), Mean(Std($returns, 12), 24)), Neg(CsRank(Skew($amt, 12))), Neg(CsRank(Delta($close, 3))))
060	Regime_Amt_Vol_Divergence_Switch	IfElse(Greater(Div($amt, Mean($amt, 24)), 1.2), Sub(CsRank($close), CsRank($volume)), Neg(CsRank(Delta($close, 3))))
061	Alpha101_12_Modern	Neg(Mul(TsRank(Delta($volume, 1), 12), TsRank(Delta($close, 1), 12)))
062	Amount_Stability_Adjusted_Returns	Neg(Div(CsRank($returns), Add(Std($amt, 12), 1e-6)))
063	Triple_Rank_Synchrony_V8	Neg(TsRank(Add(Add(CsRank(Delta($returns, 3)), CsRank(Delta($amt, 3))), CsRank(Kurt($volume, 12))), 12))
064	Regime_Price_Volume_Corr_Switch_V3	IfElse(Less(Corr($close, $volume, 12), -0.5), CsRank($returns), Neg(CsRank(Delta($close, 3))))
065	Regime_Volatility_Regime_Switch_V2	IfElse(Greater(Std($returns, 12), Mean(Std($returns, 12), 48)), Neg(CsRank(Delta($amt, 6))), Neg(CsRank(Delta($close, 1))))
066	Regime_Kurt_Amt_Flow_Switch	IfElse(Greater(Kurt($volume, 24), 3), Neg(CsRank(Delta($amt, 3))), Neg(CsRank($returns)))
067	Alpha101_54_Amt_Divergence	Neg(Mul(CsRank(Div(Sub($low, $close), Add(Sub($low, $high), 1e-6))), CsRank(Delta($amt, 6))))
068	Skewness_Regime_PV_Divergence_Fixed	Neg(IfElse(Less(Skew($returns, 24), -0.5), Neg(Sub(TsRank($close, 12), TsRank($volume, 12))), CsRank(Delta($returns, 3))))
069	Regime_Kurt_Volatility_Switch_Fixed	Neg(IfElse(Greater(Kurt($returns, 12), 3.0), Neg(CsRank(Std($returns, 6))), CsRank(Delta($returns, 3))))
070	Price_Pos_Vol_Interaction	Neg(Mul(CsRank(Div(Sub($close, $low), Add(Sub($high, $low), 0.0001))), TsRank(Std($returns, 24), 24)))
071	Range_Position_Vol_Stability_Interaction_Fixed	Mul(CsRank(Div(Sub($high, $close), Add(Sub($high, $low), 1e-6))), TsRank(Std($volume, 12), 12))
072	Kurt_Regime_PV_Corr_Switch	IfElse(Greater(Kurt($volume, 24), 3.0), Neg(Corr($close, $volume, 12)), Neg(TsRank($returns, 24)))
073	Amt_Velocity_Regime_Switch_V2	IfElse(Greater($amt, EMA($amt, 12)), Neg(TsRank(Delta($volume, 3), 12)), Neg(CsRank($returns)))
074	Range_Pos_Vol_Skew_Interaction_Fixed	Neg(Mul(CsRank(Div(Sub($close, $low), Add(Sub($high, $low), 1e-6))), TsRank(Skew($volume, 24), 12)))
075	Amt_Efficiency_Rank_Interaction	Neg(Mul(CsRank(Div($returns, $amt)), TsRank(Std($volume, 20), 20)))
076	Amt_Velocity_Volatility_Reversal	Neg(IfElse(Greater(Div($amt, Mean($amt, 24)), 2.0), Neg(Delta($close, 1)), SMA($returns, 6)))
077	Vol_of_Volume_Price_Reversal	Neg(Mul(TsRank(Std(Div($volume, Mean($volume, 24)), 24), 24), TsRank($returns, 12)))
078	Amt_Efficiency_Kurtosis_Interaction	Neg(Mul(CsRank(Div($returns, $amt)), TsRank(Kurt($volume, 20), 20)))
079	Regime_Vol_Range_Pos_Switch_Fixed	IfElse(Greater(Std($returns, 12), EMA(Std($returns, 12), 48)), Neg(Div(Sub($close, $low), Add(Sub($high, $low), 1e-6))), Div(Sub($high, $close), Add(Sub($high, $low), 1e-6)))
080	Rsquare_Resi_Adaptive	IfElse(Greater(Rsquare($close, 24), 0.7), Neg(CsRank(Slope($close, 24))), Neg(CsRank(Resi($close, 12))))
081	Regime_Trend_Vol_Corrected	IfElse(And(Greater(Rsquare($close, 24), 0.6), Greater(Std($returns, 12), Mean(Std($returns, 12), 48))), Slope($close, 12), Neg(SMA($returns, 6)))
082	Regime_Logical_Shadow_Slope_Follow	IfElse(Or(Greater(Div(Sub($high, Max2($open, $close)), Add(Sub($high, $low), 1e-6)), 0.6), Greater(Div(Sub(Min2($open, $close), $low), Add(Sub($high, $low), 1e-6)), 0.6)), Slope($close, 12), Neg(SMA($returns, 6)))
083	Rsquare_Stability_Filtered_Momentum	IfElse(And(Greater(Rsquare($close, 24), 0.5), Less(Std($returns, 12), Mean(Std($returns, 12), 48))), Slope($close, 12), Neg($returns))
084	Volatility_Sign_Agreement_Resi_Switch	IfElse(Eq(Sign(Delta(Std($returns, 12), 1)), Sign(Delta($returns, 1))), Neg(Resi($close, 12)), Neg(Slope($close, 24)))
085	Rsquare_Stability_Resi_Interaction	Neg(Mul(CsRank(Rsquare($close, 24)), CsRank(Resi($close, 12))))
086	Resi_Acceleration_Sign_Logic_Fixed	IfElse(Eq(Sign(Delta(Resi($close, 12), 1)), Sign(Resi($close, 12))), Neg(Resi($close, 6)), Neg(Slope($close, 12)))
087	Kurt_Regime_Resi_Momentum	IfElse(Greater(Kurt($returns, 24), 3.0), Neg(Resi($close, 6)), Neg(Resi($close, 24)))
088	Amt_Efficiency_Skew_Interaction	Neg(Mul(CsRank(Div($returns, Add($amt, 1e-6))), TsRank(Skew($returns, 24), 24)))
089	Extreme_Divergence_Or_Logic	IfElse(Or(Greater($volume, Mean($volume, 24)), Greater(Abs($returns), Std($returns, 24))), Neg(Resi($close, 6)), Neg(Slope($close, 12)))

(continued)

ID	Name	Formula
090	Trend_Reliability_Switch_Logic_V2	`IfElse(Greater(Rsquare($close, 24), 0.75), Neg(Slope($close, 12)), Neg(TsRank($returns, 12)))`
091	Amt_Efficiency_TsRank_Kurt_Interaction	`Neg(Mul(TsRank(Div($returns, Add($amt, 1e-6)), 24), TsRank(Kurt($returns, 24), 24)))`
092	Amt_Efficiency_EMA_Smooth_Rank	`Neg(CsRank(EMA(Div($returns, Add($amt, 1e-6)), 6)))`
093	Amt_Efficiency_Delta_TsRank	`Neg(TsRank(Delta(Div($returns, Add($amt, 1e-6)), 3), 24))`
094	And_HighVol_LowKurt_Switch	`IfElse(And(Greater(Std($returns, 12), Mean(Std($returns, 12), 60)), Less(Kurt($returns, 24), 2)), Neg(Delta($close, 3)), Neg(TsRank($returns, 24)))`
095	Higher_Moment_Regime_Switch	`IfElse(Or(Greater(Abs(Skew($returns, 24)), 1.5), Greater(Kurt($returns, 24), 4.0)), Neg(Resi($close, 6)), Neg(TsRank($returns, 24)))`
096	Amt_Efficiency_PV_Corr_Interaction	`Neg(Mul(TsRank(WMA(Div($returns, Add($amt, 1e-6)), 6), 24), TsRank(Corr($close, $volume, 24), 24)))`
097	Amt_Efficiency_Med_Smooth_Rank	`Neg(CsRank(Med(Div($returns, Add($amt, 1e-6)), 6)))`
098	Smoothed_Efficiency_Acceleration	`Neg(CsRank(Delta(WMA(Div($returns, Add($amt, 1e-6)), 6), 3)))`
099	Amt_Efficiency_Double_EMA_Cross	`Neg(Sub(EMA(Div($returns, Add($amt, 1e-6)), 6), EMA(Div($returns, Add($amt, 1e-6)), 24)))`
100	Residual_of_Residual_Acceleration	`Neg(Resi(Delta(Resi($close, 24), 3), 12))`
101	Median_Returns_Switch	`Neg(IfElse(Greater(Abs(Skew($returns, 24)), 1.0), TsRank(Div(Sub($returns, Med($returns, 24)), Add(Std($returns, 24), 1e-6)), 24), CsRank(Resi($close, 12))))`
102	High_Log_Resi_Divergence	`IfElse(Greater(Skew($high, 24), 0), TsRank(Log(Div($high, Add($close, 1e-6))), 24), CsRank(Resi($low, 24)))`
103	Open_Resi_Skew_Hybrid	`Neg(IfElse(Greater(Skew($open, 24), 0), TsRank(Resi($open, 24), 24), CsRank(SignedPower($returns, 0.5))))`
104	Median_LogSwap	`Neg(IfElse(Greater(Med($returns, 24), 0), TsRank(Log(Div($close, Add($open, 1e-6))), 24), CsRank(Sign(Resi($close, 24)))))`
105	Median_Skew_Open	`Neg(IfElse(Greater(Skew($returns,24),0.4), CsRank(Sign(Resi($open,24))), TsRank(SignedPower($returns,0.6),24)))`
106	Median_Volatility_Extreme	`Neg(IfElse(Greater(Std($volume,24),Mean(Std($volume,24),48)), CsRank(Div($amt, Add($volume,1e-6))), TsRank(Div($returns, Add(Std($returns,24),1e-6)),24)))`
107	VWAP_Vol_HighVol_Switch	`IfElse(Greater(Std($returns, 12), Med(Std($returns, 12), 48)), Mul(CsRank(Neg(Div(Sub($close, $vwap), $vwap))), CsRank(Div($volume, EMA($volume, 12)))), Neg(TsRank(Div(Sub($close, $vwap), $vwap), 24)))`
108	VWAP_Vol_Skew_Switch	`Neg(IfElse(Less(Skew($returns, 24), -0.3), Mul(CsRank(Neg(Div(Sub($close, $vwap), $vwap))), CsRank(Div($volume, EMA($volume, 12)))), TsRank(Div(Sub($close, $vwap), $vwap), 12)))`
109	VWAP_Vol_Rsquare_Switch	`IfElse(Greater(Rsquare($close, 24), 0.6), Neg(CsRank(Slope($close, 24))), Mul(CsRank(Neg(Div(Sub($close, $vwap), $vwap))), CsRank(Div($volume, EMA($volume, 12)))))`
110	PricePos_Skew_Slope	`Neg(Mul(CsRank(Div(Sub($close, $low), Add(Sub($high, $low), 1e-6))), CsRank(Mul(Skew($returns, 24), Slope($close, 12)))))`

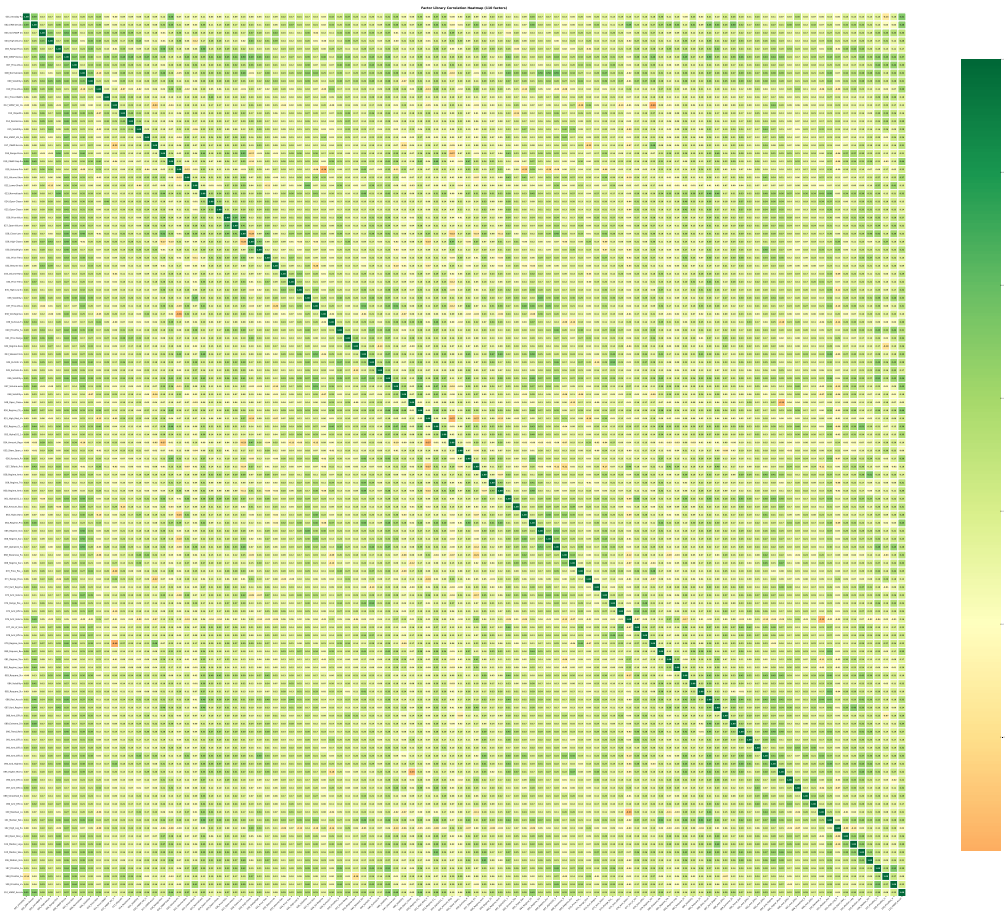


Figure 9: Pairwise Spearman correlation heatmap of the released A-share factor library (110 admitted factors), computed from cross-sectionally standardized realized factor signals over the common time–asset panel. The average off-diagonal absolute correlation is Avg $|\rho| = 0.203$.