AQuA: Recursively Self-Improving Quantitative Trading Research Agents

Jiacheng Guo[1*], Suozhi Huang[1*], Yunlong Gao[2*], Zihao Li[1], Jian Ge, Xu Kuang[3], Mengdi Wang[1]

[1]Princeton University [2]Ant Group [3]Stanford University

We study recursive self-improvement at the level of quantitative-investment research: whether an autonomous system can use evidence from earlier experiments to improve the hypotheses and candidates proposed in later iterations. We present **AQuA**, which comprises two separate language-model-driven research systems: one for symbolic factor discovery and one for trainable model development. The two systems do not share agents, memories, candidate spaces, or research state. Instead, each independently closes its own research loop by retaining validated evidence and using it to guide subsequent proposals. In this bounded sense, both systems implement recursive self-improvement at the level of the research process. Each system also uses its own sealed sandbox, which fixes the data splits, feature and label definitions, and evaluator while allowing the model to act only through constrained factor expressions or configuration diffs. The factor system, a manager-mediated multi-agent pipeline, discovers and combines factors into a signal that reaches a combined information coefficient of about 0.190 on a crypto universe. The model system, a config-driven loop over a hybrid time-series architecture, reaches a per-stock information coefficient of +0.0843 on US equities and converts it into a threshold long/short strategy with a held-out Sharpe of up to +2.50 at a two-leg cost. The strategy is positive in every year from 2021 to 2025.

1. Introduction

Quantitative-investment research searches over a large space of factors and models, and small methodological errors can turn into convincing but non-reproducible backtests. A feature that reads future information, a strategy selected on the test set, or a result confined to one favorable regime may all fail out of sample Bailey et al. (2017, 2014). Quantitative research therefore relies on frozen data splits, held-out evaluation, and skepticism toward results that look unusually strong Harvey et al. (2016).

Large language models can now propose hypotheses, write experiments, and revise their search from empirical feedback. Existing quantitative agents, however, generally focus on either factor discovery Chen and Kawashima (2025); Shi et al. (2026b) or model development Kabir et al. (2025); Song et al. (2025). More importantly, an unconstrained agent can corrupt the evidence on which its later iterations depend.

A code-generating agent may inadvertently introduce a temporal-alignment or preprocessing error that uses information unavailable at prediction time. A reviewing agent may miss the bug because the code appears semantically plausible. If the resulting leakage produces a high score, the experiment may be stored as a successful precedent and propagated through later iterations. Recursive improvement can therefore amplify an undetected error as readily as a genuine discovery.

Prompt-level instructions and model-based review do not provide a reliable integrity boundary: repeated access to a fixed holdout can cause adaptive overfitting Blum and Hardt (2015), and language-model agents have been observed exploiting misspecified objectives, tests, evaluators, or reward mechanisms Atinafu and Cohen (2026); Baker et al. (2025); Denison et al. (2024).

AQuA instead makes leakage-inducing actions unavailable to the agent. Each system fixes its data

*Equal contribution

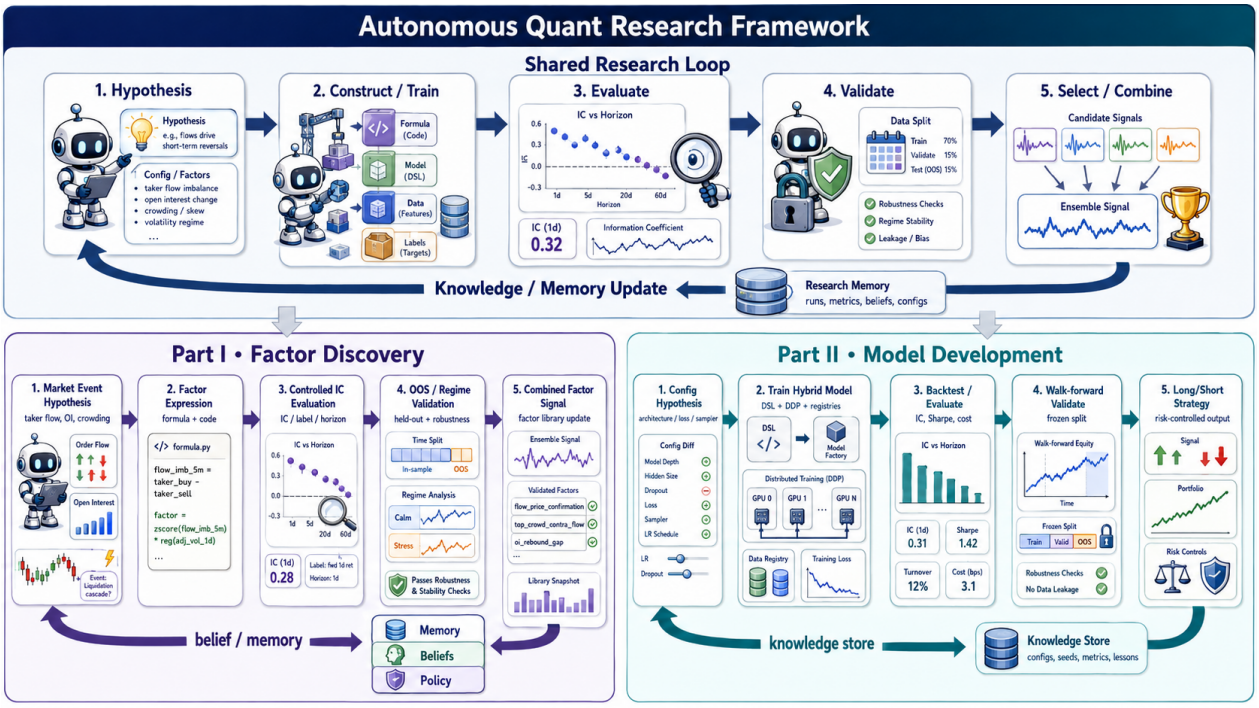

Figure 1 | Overview of the two AQuA systems. Factor discovery (Part I) and model development (Part II) use separate agents, memories, search spaces, and outputs. Each closes its own loop through hypothesis, construct/train, evaluate, validate, select/combine, and a persistent update that guides the next iteration.

pipeline, splits, labels, and evaluator before autonomous iteration begins. The agent cannot write arbitrary experimental code; it can only emit a program in a restricted domain-specific language (DSL), which contains no operation for modifying or bypassing the sealed data path or evaluator.

We build **AQuA**, comprising two separate recursively self-improving research systems: one for factor discovery and one for model development. They do not share agents, memories, candidate spaces, or outputs. In each system, validated experiments update a local research state that guides later proposals, while the experimental contract used to judge those proposals remains fixed. Thus, what improves is the research process, not the definition of success.

The key design is asymmetric freedom: the agent remains free to explore within its DSL, but the evaluator is outside the adaptive surface. Part I implements this principle with a manager-mediated multi-agent pipeline that proposes falsifiable economic hypotheses, evaluates factors, combines surviving signals, and carries beliefs across runs. Part II implements it with a config-driven loop over a hybrid time-series model; each configuration diff defines one comparable model variant, and its outcome updates the knowledge used to propose the next variant.

Both systems produce out-of-sample signal. On a crypto universe, Part I reaches a combined signal information coefficient of about 0.190. On US equities, Part II reaches a per-stock information coefficient of +0.0843, versus +0.0613 for the strongest baseline, a GRU, under the same evaluator—an absolute improvement of +0.0230 and a relative improvement of 37.5%. The resulting threshold long/short strategy reaches a held-out Sharpe of +2.50 at a two-leg cost of 2 bps and is positive in every year from 2021 to 2025. A stricter walk-forward evaluation, in which every model and strategy parameter is fixed using only data available before the next test segment, retains a Sharpe of about +2.0.

Our main contributions are:

- **We instantiate recursive self-improvement in two separate components of quantitative research.** In both factor discovery and model development, validated experiments improve later research decisions without coupling the two systems.
- **We build an autonomous factor-discovery system.** A manager-mediated multi-agent pipeline proposes, evaluates, and combines factors while carrying empirical beliefs across runs.
- **We design a hybrid time-series model and an autonomous development loop.** A config-driven agent trains directly comparable variants under a sealed evaluation sandbox.
- **We demonstrate out-of-sample predictive and trading performance.** The two systems produce positive signals in crypto and US equities, with the equity strategy remaining profitable under a fully causal walk-forward evaluation.

Section 3 describes the shared high-level pattern and the two sealed sandboxes; Sections 4 and 5 present factor discovery and model development.

2. Related Work

LLM-driven alpha mining. A fast-growing line of work uses large language models to mine formulaic alpha factors. It builds on operator-based factor search by genetic programming Cui et al. (2021); Zhang et al. (2020) and reinforcement learning Yu et al. (2023); Zhang et al. (2026); Zhao et al. (2025a,b), and now spans evolutionary and agentic search Han et al. (2026); Huang et al. (2026); Liu et al. (2025); Tang et al. (2025, 2026); Wu et al. (2026); Yi et al. (2026); Yu et al. (2026b), program-level synthesis Lin et al. (2026), graph-structured evolution Guo et al. (2026), self-evolving agents with experience memory Wang et al. (2026); Yu et al. (2026a), safety- and reproducibility-constrained generation Shi et al. (2026a), market-logic modeling Weng et al. (2026), and standardized benchmarks Luo et al. (2026). Newer systems add tree search and chain-of-thought prompting Cao (2025); Shi et al. (2026b), multi-agent generation-and-selection pipelines Chen and Kawashima (2025); Vu et al. (2026), and the fusion of formulaic factors with textual newsflow Guo and Hauptmann (2025). The classic formulaic-alpha vocabulary Kakushadze (2016) underlies all of these. Our Part I sits within this line and shares its proposal-first, memory-driven design. AQuA places this factor-discovery system alongside a separate autonomous model-development system. The two do not share agents, memory, or search state; rather, each independently uses prior experimental outcomes to improve subsequent proposals in its own domain.

Autonomous research agents. Beyond finance, language-model agents have been built to run the scientific process end to end Lu et al. (2024); Romera-Paredes et al. (2024); Xin et al. (2026); Zhou et al. (2025). The same agentic pattern has moved into trading, where multi-agent teams of language models propose and act on investment decisions Miyazaki et al. (2026); Singhi (2025). We adopt the same ambition of an autonomous research loop, but study recursive self-improvement separately in two quantitative settings: factor discovery and model development. In each setting, validated evidence from one iteration is retained and used to improve later research decisions. We additionally target the specific failure mode of quantitative work, data leakage, by making the data path and the evaluator unreachable from the agent rather than relying on its judgment.

Deep models for financial time series. The model in Part II draws on standard sequence-modeling components, including convolutional networks for sequences Bai et al. (2018), state-space models Gu and Dao (2023), and attention Vaswani et al. (2017), and on deep architectures designed for financial

data Gu et al. (2020); Zhang et al. (2019). A recent wave applies hybrid recurrent, convolutional, and transformer models, often combined with reinforcement learning, graph, or foundation-model components, to stock-return and portfolio prediction Asher Marconi (2026); Ashrafzadeh et al. (2025); Kabir et al. (2025); Kirtac and Germano (2025); Liu (2026); Song et al. (2025); Wang (2025). Our contribution is not a new primitive but a separate autonomous loop that lets the agent compose these primitives into models and accumulate evidence across variants. Its relation to Part I is conceptual—both systems learn from prior experiments—rather than architectural.

3. Method Overview

Part I and Part II of AQuA are separate research systems. They do not share agents, memories, candidate spaces, or research state. We describe them together here only because both exhibit the same high-level pattern: each system uses validated evidence from earlier experiments to guide later research decisions in its own domain.

Within each part, an iteration proceeds through five stages. It begins with a *hypothesis*: a candidate factor in Part I, or a model configuration in Part II. The hypothesis is *constructed or trained* into a concrete artifact, *evaluated* on held-out data, and *validated* against look-ahead bias and regime dependence. Surviving artifacts are then *selected or combined* into that part's running output, a combined factor signal in Part I and a trading model in Part II. A final *persistent research-state update* writes the iteration's evidence to the store belonging to that part, which its next iteration consults before proposing a new hypothesis. This feedback separates each system from a one-shot pipeline: successive iterations accumulate and reuse evidence rather than restart from scratch.

For part $p \in \{\mathrm{I}, \mathrm{II}\}$, let $R_t^{(p)}$ denote its persistent research state before iteration t, $H_t^{(p)}$ its hypothesis, $C_t^{(p)}$ the constructed factor or trained model, and $E_t^{(p)}$ the validated evidence returned by the experiment. The within-part recursion is

$$R_{t+1}^{(p)} = \mathcal{U}_p\left(R_t^{(p)}, H_t^{(p)}, C_t^{(p)}, E_t^{(p)}\right), \qquad \left(H_{t+1}^{(p)}, C_{t+1}^{(p)}\right) \sim \mathcal{P}_p\left(\cdot \mid R_{t+1}^{(p)}\right). \tag{1}$$

The part-specific update $\mathcal{U}_p$ converts experimental outcomes into reusable research knowledge, and $\mathcal{P}_p$ uses that knowledge to guide the next proposal. The superscript emphasizes the separation: Part I does not update $R^{(\mathrm{II})}$, and Part II does not update $R^{(\mathrm{I})}$. We use *recursive self-improvement* in this bounded, research-process-level sense within each part; neither system updates the underlying language model or the evaluator.

The commonality is therefore a high-level research pattern, not a shared implementation. Construction writes a symbolic factor expression in Part I and trains a hybrid time-series model in Part II; evaluation reports an information coefficient in both, but under part-specific conventions, so the two coefficients are not directly comparable and we report them separately.

3.1. Sealed sandbox and constrained iteration

Because the language model proposes and, in places, writes its own experiments, the central failure mode is data leakage: an autonomously generated feature, label, or normalization that consults information unavailable at prediction time. Backtest overfitting of this kind is well documented and produces simulated performance that does not survive out of sample Bailey et al. (2017, 2014); Harvey et al. (2016). We contain it structurally by fixing a sandbox before any iteration begins. The data splits $\mathcal{D}$, the feature and label definitions $\mathcal{F}$ and $\mathcal{L}$, and the evaluator $\mathcal{V}$ are sealed and human-authored, and the model never edits them. Each action of the model is a specification θ drawn

from a constrained space Θ, which the harness compiles and scores through the sealed evaluator,

$$s_k = \mathcal{V}\big(C(\theta_k);\, \mathcal{S}\big), \qquad \theta_k \in \Theta, \quad \mathcal{S} = (\mathcal{D}, \mathcal{F}, \mathcal{L}, \mathcal{V}) \text{ sealed.} \tag{2}$$

The space Θ is defined so that no θ can alter $\mathcal{S}$. Leakage cannot enter through generation because generation cannot reach the sealed components.

Sealing the data path closes leakage through generation, but a second channel remains through selection: a search that can read the metric it will ultimately report will, given enough iterations, learn to select for it. We close this channel by separating the metric the loop optimizes from the metric it reports. During search the harness returns to the agent only a score on a validation slice fixed in advance; the score s_k that ranks candidates and the inner-validation signal that drives early stopping and checkpoint choice are computed on that slice alone. Where a part designates a final test window, that window is scored once, after the configuration is frozen, and is never returned to the agent or used to rank candidates. In Part II this window is the untouched 2021–2025 period, so the reported test coefficient is out of sample with respect to the entire search.

In Part I, the registry $\mathcal{O}$ is the standard vocabulary of formulaic-alpha operators Kakushadze (2016); Zhang et al. (2020). Its leaves are raw fields (open, high, low, close, volume, vwap, returns); its inner nodes are operators of three kinds: cross-sectional operators that act across the universe at a fixed timestamp (rank, z-score, sector neutralization), time-series operators that summarize a trailing window for each entity (lag, difference, moving correlation and covariance, rolling rank, rolling standard deviation, linear-decay weighting), and element-wise arithmetic and conditionals. A factor is a composition of these operators over the raw fields, represented as an expression tree Cui et al. (2021); Zhang et al. (2020); the running example

$$f = \mathrm{rank}\big(\,\mathrm{corr}(r^{1d}, v^{1d}, 20)\big) - \mathrm{rank}\big(\,\mathrm{std}(r^{1d}, 20)\big) \tag{3}$$

is one such tree. Every time-series operator reads only its trailing window and every cross-sectional operator reads only the current timestamp, so causality is closed under composition: any expression the model assembles from $\mathcal{O}$ is causal by construction, and it cannot introduce a primitive that consults future data. Operator-based search of this space has a long line of work, from genetic programming Cui et al. (2021); Zhang et al. (2020) and reinforcement learning Yu et al. (2023); Zhao et al. (2025a) to recent language-model agents Han et al. (2026); Shi et al. (2026b); Tang et al. (2025). Before assembling the expression, the agent states each factor as a falsifiable proposal, with a hypothesis, mechanism, predicted direction, and refutation conditions. Listing 1 shows one such proposal.

In Part II, the specification is a configuration in a domain-specific language whose registry covers the full experiment: the data split selected from the frozen set, the sampler, the architecture blocks, the loss, and the optimizer. A hypothesis is a single config diff. The registry is the model's only surface; it selects and parameterizes registered operators but cannot write the data loader, the split logic, or the evaluator. Listing 2 shows one such configuration. The architecture registry composes standard sequence-modeling primitives, including temporal convolutions Bai et al. (2018), state-space mixers Gu and Dao (2023), and attention Vaswani et al. (2017), into deep multi-resolution stacks.

Formally, $\Theta_{\mathrm{II}} = \{\, c = (\mathrm{split}, \mathrm{sampler}, \mathrm{arch}, \mathrm{loss}, \mathrm{optim}) \,\}$, where split is chosen from the frozen $\mathcal{D}$ and cannot be redefined, and the remaining components are drawn from their registries. The compiler builds the model and training run, and the sealed evaluator reports held-out IC, R^2, and Sharpe under a two-leg cost. Because the data path is sealed, the split is frozen, and every feature is causal, every $c \in \Theta_{\mathrm{II}}$ yields a leakage-free experiment. One config diff also corresponds to exactly one variant, which keeps variants directly comparable. Sections 4 and 5 instantiate these two registries in full.

Listing 1 | A proposal in Part I is a falsifiable factor hypothesis, with its mechanism, predicted direction, and refutation conditions, stated before any expression is built. The deployed expression is withheld; the full iteration record is in Appendix A.

```
proposal:
  hypothesis: >
    After a forced open-interest unwind, a price rebound that is not confirmed
    by aggressive taker flow is more likely to fail.
  mechanism: >
    Deleveraging removes forced pressure, but weak buy-flow and poor basis
    recovery indicate insufficient demand behind the rebound.
  expected_direction: higher_signal_predicts_lower_future_return
  expected_label: [ret_open_open_h10, ret_open_open_h30]
  falsification_criteria:
    - no IC concentration inside the deleveraging event window
    - sign flips or vanishes out of sample or across regimes
    - subsumed by a price, open-interest, or taker-flow baseline
  expected_failure_modes:
    - rebound strength already captured by short-horizon momentum
    - taker-flow gap is noise once the basis has normalized
  factor_blueprint:
    - deleveraging_intensity: ranked negative change in open interest
    - rebound_strength: short-horizon price recovery after the event
    - flow_gap: lack of taker-flow confirmation during the rebound
  expression: withheld
```

4. Part I: Autonomous Factor Discovery

4.1. System architecture

Part I of AQuA is a multi-agent system that turns a research goal into validated factors. An AI Manager mediates each run. It reads the research policy, the accumulated memory, and the record of previous runs, and from these it writes a plan that assigns every downstream agent a concrete task. The agents never call one another; each handoff passes through the Manager, which keeps a run auditable and reproducible. Figure 2 shows the pipeline.

Six specialist agents run in sequence. The Data Steward loads and aligns the market data and returns a quality report together with a set of atomic features. The Visual Analyst searches the history for representative events of a requested type and summarizes them into event profiles. The Idea Miner proposes candidate factors from those profiles. The Factor Evaluator scores each candidate, the Backtest Engineer trades it in simulation, and the Research Librarian records the outcome. The proposal and validation steps are detailed below; the specific factors the system selects are not disclosed.

4.2. The research pipeline

A factor enters the pipeline as a proposal rather than as an expression. For each candidate the Idea Miner states a hypothesis, the economic mechanism behind it, the direction it is expected to predict, and the conditions under which it should be considered refuted. This framing requires every factor to carry its own rationale and its own test before it is built, a discipline that the recent agentic-mining literature has converged on Han et al. (2026); Shi et al. (2026a); Tang et al. (2025, 2026); Vu et al. (2026). The candidate is then assembled from the formulaic-alpha operator registry of Section 3.1, where Listing 1 shows the proposal form it emits.

Listing 2 | A hypothesis in Part II is one config diff over the sealed sandbox, the only artifact the model emits. Sealed components (splits, features, labels, evaluator) are referenced by id and never redefined; operator names and values are illustrative.

```
sandbox:
  splits:    frozen/sp_A          # train / val / test
  features:  frozen/feat_g12
  label:     frozen/lbl_fwd
  evaluator: frozen/eval          # held-out, two-leg cost, walk-forward

input:   {normalize: per_entity_z, stats_window: train_only, history: 64}
sampler: {kind: stratified_minute, batch: 8192}

arch:                             # composed from the architecture registry
  - conv_stem:       {scales: [3, 5, 15]}           # multi-scale 1-D conv
  - multi_resolution:
      fine:   {repeat: 4, block: [temporal_conv, sequence_mixer/state_space]}
      coarse: {repeat: 2, block: [sequence_mixer/attention, feedforward]}
      fuse:   cross_attention
  - block_repeat:
      times: 3                                      # panel interaction
      block: [cross_entity_mixer, temporal_conv/depthwise, feedforward]
  - readout:         {gate: [fine, coarse, panel], pool: [last, mean, attn]}

loss:  [spearman_ic, huber_csz, turnover_reg]
optim: {adamw, lr: 3.0e-4, schedule: cosine, precision: bf16, ddp: 8}
eval:  {cost_bps: 2, walk_forward: expanding}   # held-out
```

Evaluation follows the same contract for every proposal. The Factor Evaluator computes information coefficients across forward-return labels, monthly stability, held-out split behavior, market-regime behavior, turnover, expression complexity, and correlation with the existing factor pool. It also performs controlled comparisons against simple baselines built from price, volume, open interest, basis, and taker-flow changes. When a proposal is tied to an event, its effect inside the event window is compared with a control region outside the event. A factor is carried forward when its signal is not simply a restatement of a baseline and when its strongest performance appears in the market context predicted by its mechanism.

The Backtest Engineer then turns the evaluated signal into a simple quantile portfolio. It tests both the proposed direction and the reversed direction and keeps the cleaner trading interpretation. This direction-calibration step matters because a formula can have predictive content even when the initially stated sign is wrong. The final object stored by the system is therefore not only a factor score, but a factor together with its direction, target horizon, supporting mechanism, evaluation evidence, and trading behavior.

4.3. Cross-run learning

The system improves across runs because it maintains memory. After each run, the Research Librarian writes a structured record containing the goal, event type, visual observations, proposed mechanisms, evaluated factors, selected signals, backtest summaries, and updated beliefs. A belief attaches confidence to a mechanism in a market context, such as “open-interest crashes followed by weak flow-confirmed rebounds tend to reverse” or “quiet volume bursts continue only when price acceptance and taker flow agree.” The next run is planned against this memory.

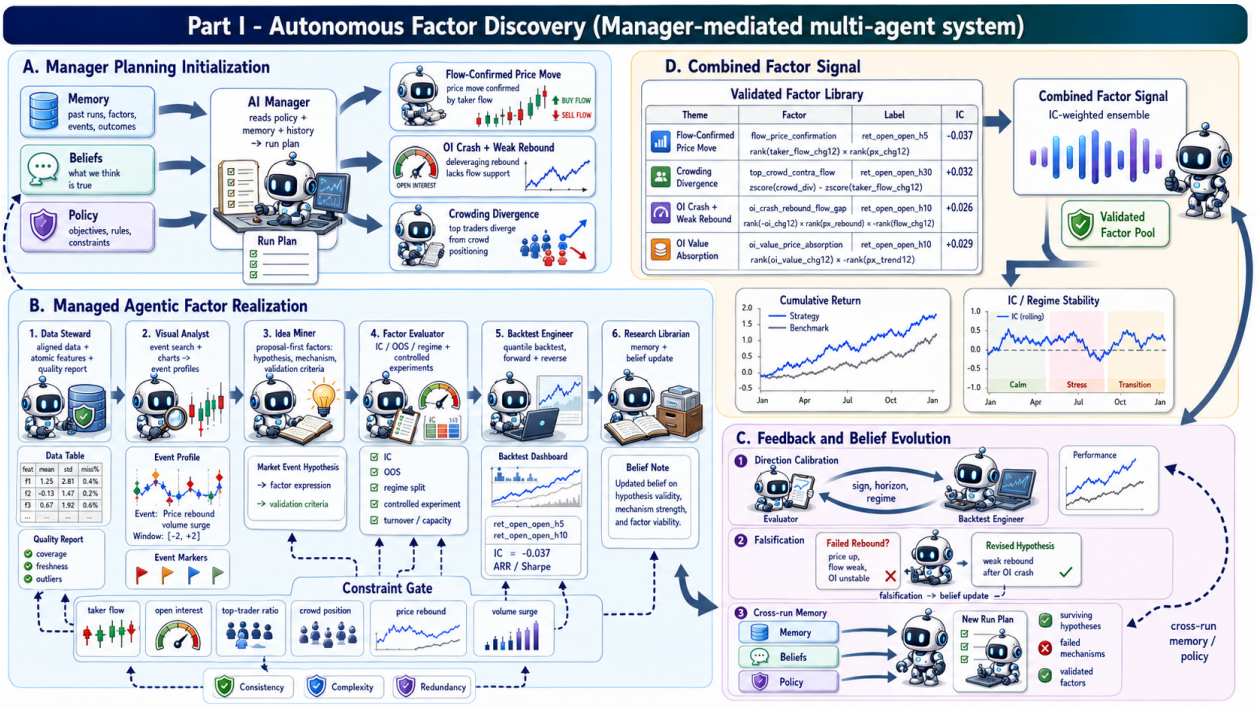

Figure 2 | Part I architecture. An AI Manager orchestrates a six-agent pipeline (Data Steward → Visual Analyst → Idea Miner → Factor Evaluator → Backtest Engineer → Research Librarian) that turns a research goal into a combined factor signal. Three nested feedback loops, for direction calibration, falsification-driven belief update, and cross-run memory and policy, drive iterative improvement, backed by persistent Memory, Beliefs, and Policy stores.

This creates three feedback loops. The first operates within a single backtest, where the system calibrates the direction of a signal. The second operates within a run, where failed proposals update the belief state and sharpen the interpretation of the event. The third operates across runs, where the AI Manager reads the accumulated memory and steers the next search toward mechanisms that have earned evidence. In practice, later runs do not start from a blank prompt. They inherit prior observations, avoid mechanisms that repeatedly failed, and refine promising mechanisms with new event definitions, horizons, or conditioning variables.

4.4. A worked iteration

To make the loop concrete, Appendix A records one full iteration, and we summarize it here. The run asks whether a weak price rebound after an open-interest crash predicts reversal. The Manager turns this question into a plan: find deleveraging episodes, inspect price and flow after the unwind, and test whether rebound quality separates continuation from failure. The Visual Analyst returns event profiles that separate a clean forced-deleveraging pattern, where open interest falls, volume expands, and price rebounds only briefly, from a healthier reset where taker flow and basis recover with price. From this the Idea Miner proposes a mechanism in which a rebound left unconfirmed by aggressive taker flow is more likely to fail. The evaluator clears it against the price, open-interest, and flow baselines, with its skill concentrated in the event window its mechanism targets. The strongest factors in this family reach single-factor information coefficients on the order of 0.026 to 0.037. A later run changes the goal to quiet-market volume expansion and reuses the same machinery without touching the evaluator, showing that the loop explores a new mechanism by re-planning rather than re-coding. The deployed expressions are withheld throughout.

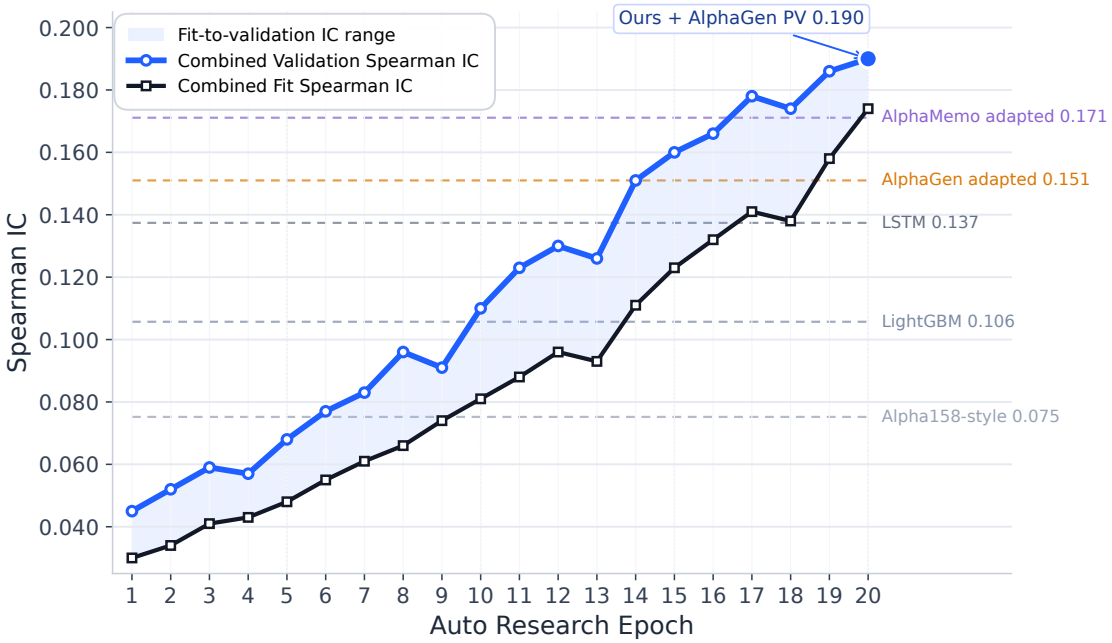

Figure 3 | Part I combined-factor signal across autonomous-research iterations. The combined validation IC improves as the loop accumulates and reuses validated evidence, reaching a combined signal IC of approximately 0.190. *Part I IC is the combined-factor Spearman IC on the crypto five-minute universe, a different convention from Part II; the two numbers should not be compared directly.*

4.5. Results

The output of Part I is a combined factor signal formed from the factors that survive the evaluation and direction-calibration stages. Figure 3 reports the quality of this combined signal across autonomous-research iterations. The combined validation information coefficient rises as the loop accumulates and reuses evidence, reaching approximately 0.190. The improvement coincides with the addition of richer event profiles, open-interest and flow conditioning, crowding-divergence mechanisms, and regime-aware factor selection. As noted in Section 3, the Part I information coefficient follows the combined-factor convention on the crypto five-minute universe, and is not comparable to the per-stock coefficient reported in Part II.

Each mechanism discovered by the loop is an economically grounded signal, typically with an information coefficient around 0.03 in absolute value. This is the expected regime for intraday formulaic signals: the strength comes not from any single rule but from combining a library of mechanism-grounded factors into a stronger aggregate signal. The Part I result is therefore not a claim that one discovered expression is sufficient, but that an autonomous, memory-bearing research harness can repeatedly turn market hypotheses into tested factor evidence and improve the combined signal over iterations.

5. Part II: Autonomous Model Development

5.1. The research loop

Part II of AQuA develops trading models through the config-driven loop of Figure 4. Each iteration begins with a hypothesis written as a single config diff over the sealed sandbox of Section 3.1: a change to the architecture, the loss, the sampler, or the optimizer. The harness compiles the diff into

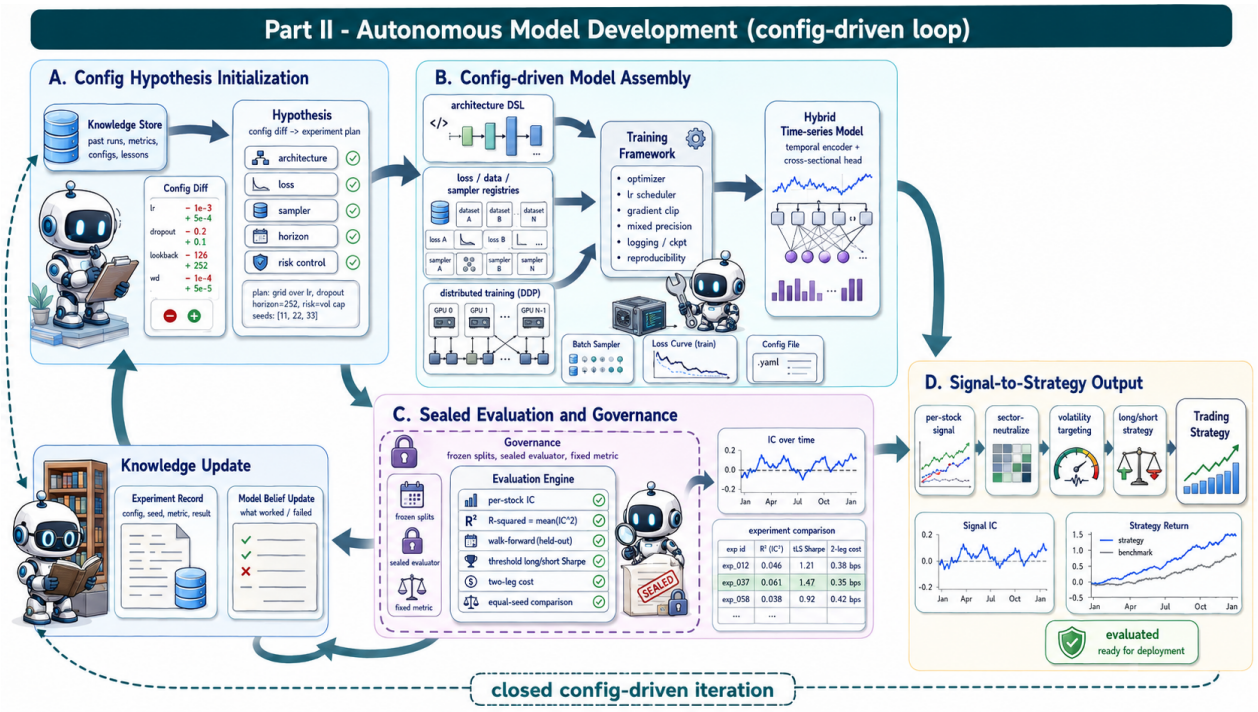

Figure 4 | Part II architecture. A config-driven loop, from a hypothesis (config diff) through the training framework and the evaluation engine to a knowledge update, iterates over model variants. The training framework (architecture DSL, loss, data, and sampler registries, distributed training) produces a hybrid time-series model; the evaluation engine reports held-out per-stock IC, R^2 = mean(IC^2), walk-forward, and a two-leg-cost threshold long/short Sharpe, within a governance contract of frozen splits, a sealed evaluator, and a fixed selection metric. The model signal is constructed into a trading strategy (sector-neutralize, then volatility targeting, then long/short).

a training run, scores it through the sealed evaluator, and writes the result to a knowledge store that the next hypothesis reads. Because one diff produces exactly one variant and the data path is fixed, two variants differ only in the knobs that changed. This keeps them directly comparable and keeps the search leakage-free.

5.2. The hybrid model

Our predictor is a hybrid model that combines convolutional feature extraction with sequence modeling Kabir et al. (2025); Zhang et al. (2019), shown in Figure 5. The front-end is a deep convolutional stack: several blocks of one-dimensional convolutions run over the input history at multiple kernel sizes and dilations, building a rich local representation of each stock's recent price-volume dynamics at every timestep. We leave the precise convolutional configuration unspecified. These representations feed a temporal-modeling stage instantiated from sequence models, spanning recurrent networks (LSTM Hochreiter and Schmidhuber (1997)), state-space models (Mamba Gu and Dao (2023)), and attention (Transformer Bai et al. (2018); Vaswani et al. (2017)); the configuration used in our experiments uses attention. A cross-sectional stage then mixes information across the panel of stocks at each timestep, the branches are fused with a gating mechanism, and a pooled readout emits the per-stock score.

The whole model is one point in the configuration space of Section 3.1: the convolutional front-end, the sequence family, the cross-sectional mixing, and the fusion are all operators in the registry, so a

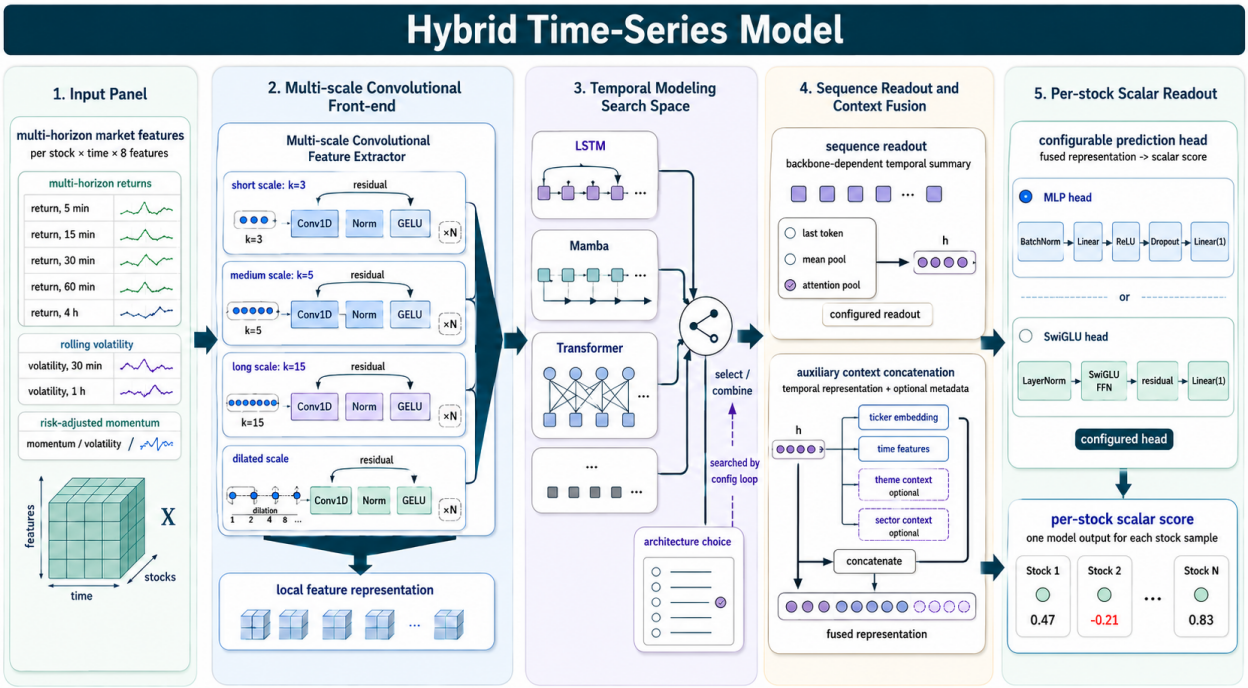

Figure 5 | The hybrid time-series model. A multi-scale convolutional front-end extracts local representations from multi-horizon return, volatility, and risk-adjusted momentum features at each timestep. A configurable temporal backbone, instantiated as a recurrent, state-space, or attention-based sequence model, captures longer-range temporal dynamics. A backbone-dependent sequence readout produces a fixed-dimensional representation, which is concatenated with configured auxiliary information such as ticker embeddings, time features, and optional theme or sector context. A configurable MLP or SwiGLU prediction head then emits one scalar score for each stock sample.

new model is a new config rather than new code. The framework trains it under the sealed sandbox, and the exact feature set, normalization, and label construction are part of that sandbox and are not disclosed.

5.3. Task, features, and baselines

We instantiate Part II on intraday US equity prediction. The task is to predict each stock's forward return over the next thirty minutes. We split the data chronologically: we train on 2010–2019, leave 2020 as an embargo gap that no part of training or selection touches, and report on the untouched 2021–2025 test window. Model selection (early stopping and checkpoint choice) is driven only by an inner-validation slice taken from the end of the training window, and the test window is used solely for final evaluation, never informing training or selection. The model input is a short history of pure price-volume features, standardized per stock.

No single feature carries the signal on its own. Table 1 reports the single-feature information coefficient of a representative set of price-volume features over the held-out window. None of these features, nor a ridge linear combination of them, exceeds about 0.03 in magnitude; the ridge reaches +0.025. The predictable signal lives in their joint, nonlinear, and temporal structure, which is what the model is built to capture.

The autonomous loop searched a range of model families on this task, all evaluated identically on the held-out window. Table 2 reports their per-stock raw IC, from a linear model through gradient

Table 1 | Single-feature information coefficient of representative price-volume features, held out over 2021–2025 (per-stock raw IC). No single feature, or ridge combination of them, exceeds about 0.03 in magnitude.

Price-volume feature	single-feature IC
return, 5 min	−0.031
return, 15 min	−0.021
return, 30 min	−0.013
return, 60 min	−0.001
return, 4 h	+0.002
volatility, 30 min	+0.006
volatility, 1 h	+0.004
momentum / volatility	+0.007

Table 2 | Model comparison on the held-out window (2021–2025), per-stock raw IC. Models are trained on identical data and scored by the same evaluator. Our hybrid model is the strongest.

Model	Family	raw IC
Linear (ridge)	linear	+0.0251
LGB	gradient boosting	+0.0397
xLSTM	recurrent	+0.0434
LSTM	recurrent	+0.0535
GRU	recurrent	+0.0613
Ours (hybrid)	**hybrid**	**+0.0843**

boosting and recurrent networks Beck et al. (2024); Cho et al. (2014); Hochreiter and Schmidhuber (1997) to our hybrid model. The progression is clear: linear and tree models capture part of the signal, sequence models more, and our hybrid model is the strongest.

5.4. Evaluation engine

Every run is scored by the same sealed evaluator. It reports three quantities on held-out data: a per-stock time-series information coefficient, a per-stock R^2 defined as the cross-sectional mean of squared IC, and a threshold long/short Sharpe ratio under a two-leg turnover cost.

Figure 6 summarizes the trading result. The cumulative return of the volatility-targeted, dollar-neutral long/short book climbs steadily across 2021–2025, making new highs into the end of the window and ending well above a Nasdaq-100 buy-and-hold benchmark over the same period without taking its 2022 drawdown.

5.5. From signal to strategy

We trace one model from prediction to deployable strategy. A trained model produces a per-stock score at each timestamp. The score is turned into a dollar-neutral threshold long/short book: stocks above an upper threshold are held long and stocks below a lower threshold short, the book is rebalanced on a fixed cadence, and each leg pays a two-leg turnover cost. Two construction steps lift the held-out Sharpe. Sector-neutralizing the score removes common sector exposure and raises the held-out Sharpe to +2.15, with the training and held-out values nearly equal, which indicates that the construction is

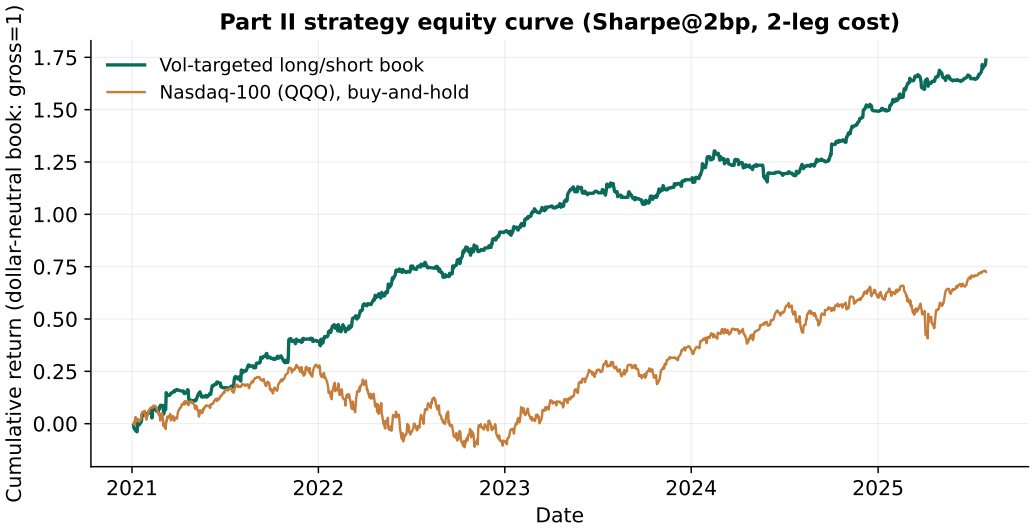

Figure 6 | Part II strategy equity curve. Cumulative return of the volatility-targeted, dollar-neutral threshold long/short book (gross exposure 1, two-leg turnover cost 2 bps) over 2021–2025, with the Nasdaq-100 (QQQ) buy-and-hold return over the same window shown for reference. The market-neutral book compounds more smoothly and sidesteps the 2022 index drawdown; QQQ is long-only and not risk-matched to the neutral book.

not overfit. A causal volatility-targeting overlay, which scales daily exposure by an online estimate of trailing volatility toward an expanding-median target, raises it further to +2.50.

5.6. Results

Table 3 reports the held-out metrics for 2021–2025. The model reaches a per-stock raw information coefficient of +0.0843 and a per-stock R^2 of 1.20%. The threshold long/short book reaches a Sharpe of +2.15 before the volatility overlay and +2.50 after it, and a fully causal walk-forward that chooses every parameter from past data alone still reaches +2.0. Table 4 breaks the Sharpe down by year. It is positive in every year from 2021 to 2025, including the held-out years and the 2022 drawdown, so the strategy is not carried by a single regime. As stated in Section 3, the Part II information coefficient is a per-stock time-series quantity, reported on its own and not against Part I.

6. Discussion

Part I and Part II are separate systems rather than a coupled or shared-state framework. They use different agents, memories, candidate spaces, and outputs. Each nevertheless exhibits the same high-level property: an autonomous loop proposes a candidate, validates it on held-out data, accumulates the resulting evidence in its own persistent research state, and uses that state to improve later research decisions. Part I applies this within-system recursion to economic hypotheses and symbolic factor expressions; Part II applies it independently to model architectures and training configurations. The commonality is therefore descriptive rather than architectural: in both cases, one experiment changes the design of the next within the same system.

What makes both systems reliable is where each places trust. Its sandbox seals the data path and scores the search on a validation metric it cannot confuse with the reported one, so a surviving result is credible from how the environment is built rather than from an audit of the agent's reasoning. This

Table 3 | Part II main results on the held-out evaluation window (2021–2025). All figures are out of sample, with model selection on validation only. The information coefficient is the per-stock time-series Pearson IC (raw); R^2 is the cross-sectional mean of squared IC; the Sharpe ratio is for a dollar-neutral threshold long/short book at a two-leg turnover cost of 2 bps, with parameters tuned on a training fraction and scored on the held-out remainder. *(IC convention differs from Part I and the two should not be compared.)*

Metric (held out, 2021–2025)	Value
① Per-stock IC (raw)	+0.0843
② Per-stock R^2 = mean(IC^2) (raw)	1.20%
③ Sharpe@2bp, sector-neutral book	+2.15
+ causal volatility targeting	+2.50
fully-causal walk-forward (no hindsight)	+2.00

Table 4 | Part II strategy Sharpe@2bp by calendar year, held out, showing performance is not concentrated in any single regime.

Year	2021	2022	2023	2024	2025
Sharpe@2bp	+1.7	+3.5	+1.9	+1.8	+2.7

lets each autonomous, and at times opaque, search improve its research process without inheriting its capacity to overfit the number it optimizes.

We found it useful to separate leakage into two channels, and that split is the part of this work most likely to transfer beyond finance. Generation leakage enters when the agent can define a feature, label, or transform that consults information unavailable at prediction time; we close it by construction, since no admissible specification can reach the sealed data path. Selection leakage enters when the agent can read the metric it will be judged on and, over enough iterations, learn to select for it; we close it by reporting a metric the loop never optimizes against. Any autonomous research agent scored by an evaluator faces both channels, whatever the domain, so the sealed-sandbox and split-metric construction is a general recipe rather than a finance-specific trick.

Coupling the two systems is the natural next step: the factors discovered in Part I are direct inputs to the models trained in Part II. That coupling introduces a leakage channel neither part has on its own. If factor discovery and model training draw on the same data, a factor selected for its in-sample signal can hand the model a subtly overfit input, so the two searches come to share information the sealed metric was meant to keep apart. Keeping the coupled system honest means sealing the discovered factor set before the model loop begins, and treating the factor library as another frozen component of the Part II sandbox rather than a live search the model can steer.

7. Limitations

Two scope limitations bound these results. Each system is demonstrated on a single market and horizon, crypto at five minutes for Part I and US equities at thirty minutes for Part II, and we do not claim the numbers transfer to other markets or frequencies without re-tuning. The loops also run with a human operator who sets the research goal, owns the sandbox, and supervises promotion, so the systems are autonomous within those bounds rather than unattended. The reported metrics are simulated under a turnover-cost model and have not been validated in live trading.

The guarantees are also uneven across the two channels of leakage. Sealing the data and feature path is structural: no admissible specification can reach past it, so causal correctness holds by construction. Keeping the final test window out of selection is weaker. The harness returns only validation scores during search, but the isolation of the test window rests on the sealed protocol and operator discipline rather than a hard technical barrier, and an operator with direct access to the store could in principle consult it. We therefore treat test isolation as a governance property to be audited over a run, not as a cryptographic guarantee.

8. Conclusion

We presented AQuA, which comprises two separate autonomous research systems: one for factor discovery and one for model development. The two parts operate over different research objects and do not share agents, memories, candidate spaces, or research state. Both nevertheless implement recursive self-improvement within their own research process: validated evidence is incorporated into a part-specific state that guides subsequent hypotheses and candidate designs. A separate sealed sandbox in each part keeps the data path and evaluator outside this recursive update. On a crypto universe the factor system reaches a combined signal IC of about 0.190, and on US equities the model system reaches a per-stock IC of +0.0843 and a regime-robust threshold long/short Sharpe of up to +2.50 out of sample. Coupling the two systems, so that discovered factors feed the model loop, is the natural next direction.

References

B. Asher Marconi. Time Series Foundation Models for Multivariate Financial Time Series Forecasting, 2026. URL https://www.ssrn.com/abstract=6085266.

M. Ashrafzadeh, M. Sadrani, and S. H. Zolfani. Deep learning and machine learning models for portfolio optimization: Enhancing return prediction with stock clustering. *Results in Engineering*, 27:106263, Sept. 2025. ISSN 25901230. doi: 10.1016/j.rineng.2025.106263. URL https://linkinghub.elsevier.com/retrieve/pii/S2590123025023357.

Y. Atinafu and R. Cohen. RewardHackingAgents: Benchmarking evaluation integrity for LLM ML-engineering agents, 2026. URL https://arxiv.org/abs/2603.11337. arXiv:2603.11337.

S. Bai, J. Z. Kolter, and V. Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, Apr. 2018. URL http://arxiv.org/abs/1803.01271. arXiv:1803.01271.

D. Bailey, J. Borwein, M. López De Prado, and Q. J. Zhu. The probability of backtest overfitting. *The Journal of Computational Finance*, 2017. ISSN 14601559. doi: 10.21314/JCF.2016.322. URL http://www.risk.net/journal-of-computational-finance/technical-paper/2471206/the-probability-of-backtest-overfitting.

D. H. Bailey, J. M. Borwein, M. López De Prado, and Q. J. Zhu. Pseudo-Mathematics and Financial Charlatanism: The Effects of Backtest Overfitting on Out-of-Sample Performance. *Notices of the American Mathematical Society*, 61(5):458, May 2014. ISSN 0002-9920, 1088-9477. doi: 10.1090/noti1105. URL https://www.ams.org/jourcgi/jour-getitem?pii=noti1105.

B. Baker, J. Huizinga, L. Gao, Z. Dou, M. Y. Guan, A. Madry, W. Zaremba, J. Pachocki, and D. Farhi. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation, 2025. URL https://arxiv.org/abs/2503.11926. arXiv:2503.11926.

M. Beck, K. Pöppel, M. Spanring, A. Auer, O. Prudnikova, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter. xLSTM: Extended Long Short-Term Memory, Dec. 2024. URL http://arxiv.org/abs/2405.04517. arXiv:2405.04517.

A. Blum and M. Hardt. The ladder: A reliable leaderboard for machine learning competitions. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1006–1014. PMLR, 2015. URL https://proceedings.mlr.press/v37/blum15.html.

L. Cao. Chain-of-Alpha: Unleashing the Power of Large Language Models for Alpha Mining in Quantitative Trading, Aug. 2025. URL http://arxiv.org/abs/2508.06312. arXiv:2508.06312.

Q. Chen and H. Kawashima. Multi-Agent LLM Framework for Formulaic Alpha Generation and Selection in Quantitative Trading. In *2025 IEEE International Conference on Big Data (BigData)*, pages 7143–7152. IEEE, Dec. 2025. doi: 10.1109/BigData66926.2025.11400963. URL https://ieeexplore.ieee.org/document/11400963/.

K. Cho, B. v. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, Sept. 2014. URL http://arxiv.org/abs/1406.1078. arXiv:1406.1078.

C. Cui, W. Wang, M. Zhang, G. Chen, Z. Luo, and B. C. Ooi. AlphaEvolve: A Learning Framework to Discover Novel Alphas in Quantitative Investment. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2208–2216, Virtual Event China, June 2021. ACM. ISBN 9781450383431. doi: 10.1145/3448016.3457324. URL https://dl.acm.org/doi/10.1145/3448016.3457324.

C. Denison, M. MacDiarmid, F. Barez, D. Duvenaud, S. Kravec, S. Marks, N. Schiefer, R. Soklaski, A. Tamkin, J. Kaplan, B. Shlegeris, S. R. Bowman, E. Perez, and E. Hubinger. Sycophancy to subterfuge: Investigating reward-tampering in large language models, 2024. URL https://arxiv.org/abs/2406.10162. arXiv:2406.10162.

A. Gu and T. Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces, 2023. URL http://arxiv.org/abs/2312.00752. arXiv:2312.00752.

S. Gu, B. Kelly, and D. Xiu. Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies*, 33(5):2223–2273, May 2020. ISSN 0893-9454, 1465-7368. doi: 10.1093/rfs/hhaa009. URL https://academic.oup.com/rfs/article/33/5/2223/5758276.

T. Guo and E. Hauptmann. Exploring the Synergy of Quantitative Factors and Newsflow Representations from Large Language Models for Stock Return Prediction, Nov. 2025. URL http://arxiv.org/abs/2510.15691. arXiv:2510.15691.

T. Guo, H. Shen, J. Luo, B. Chen, H. Ding, J. Huang, L. Liu, Y. Ma, and M. Zhang. AlphaPROBE: Alpha Mining via Principled Retrieval and On-graph biased evolution, Feb. 2026. URL http://arxiv.org/abs/2602.11917. arXiv:2602.11917.

J. Han, S. Zhang, W. Li, Y. Dong, T. Hu, Y. Zhu, X. Yu, X. Guo, Z. Liu, K. Wang, J. Liu, T. Jiang, R. An, S. Hu, Z. Yang, R. Che, and H. Wang. QuantaAlpha: An Evolutionary Framework for LLM-Driven Alpha Mining, May 2026. URL http://arxiv.org/abs/2602.07085. arXiv:2602.07085.

C. R. Harvey, Y. Liu, and H. Zhu. ... and the Cross-Section of Expected Returns. *Review of Financial Studies*, 29(1):5–68, Jan. 2016. ISSN 0893-9454, 1465-7368. doi: 10.1093/rfs/hhv059. URL https://academic.oup.com/rfs/article-lookup/doi/10.1093/rfs/hhv059.

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, Nov. 1997. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco.1997.9.8.1735. URL https://direct.mit.edu/neco/article/9/8/1735-1780/6109.

Y. Huang, Z. Fan, K. Hu, and Y. Ye. From hypotheses to factors: Constrained llm agents in cryptocurrency markets, Apr. 2026. URL http://arxiv.org/abs/2604.26747. arXiv:2604.26747.

M. R. Kabir, D. Bhadra, M. Ridoy, and M. Milanova. LSTM–Transformer-Based Robust Hybrid Deep Learning Model for Financial Time Series Forecasting. *Sci*, 7(1):7, Jan. 2025. ISSN 2413-4155. doi: 10.3390/sci7010007. URL https://www.mdpi.com/2413-4155/7/1/7.

Z. Kakushadze. 101 Formulaic Alphas, Mar. 2016. URL http://arxiv.org/abs/1601.00991. arXiv:1601.00991.

K. Kirtac and G. Germano. Large language models in finance: estimating financial sentiment for stock prediction, 2025. URL https://www.ssrn.com/abstract=5166656.

Q. Lin, R. Feng, Y. Feng, Z. Huang, Y. Chen, Z. Yang, L. Zhou, B. Fei, J. Liu, and Y. Li. FactorEngine: A Program-level Knowledge-Infused Factor Mining Framework for Quantitative Investment, Apr. 2026. URL http://arxiv.org/abs/2603.16365. arXiv:2603.16365.

F. Liu, Y. Huang, S. Luo, Y. Wang, Y. Yang, X. Li, Z. Hu, J. Feng, and Q. Liu. Cognitive alpha mining via llm-driven code-based evolution, Nov. 2025. URL http://arxiv.org/abs/2511.18850. arXiv:2511.18850.

T. Liu. A Comparative Study of Transformer-Based and Classical Models for Financial Time-Series Forecasting. *Journal of Risk and Financial Management*, 19(3):203, Mar. 2026. ISSN 1911-8074. doi: 10.3390/jrfm19030203. URL https://www.mdpi.com/1911-8074/19/3/203.

C. Lu, C. Lu, R. T. Lange, J. Foerster, J. Clune, and D. Ha. The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery, Sept. 2024. URL http://arxiv.org/abs/2408.06292. arXiv:2408.06292.

H. Luo, H. T. Ko, J. Chen, D. Sun, Y. Zhang, and C. Liu. AlphaBench: Benchmarking large language models in formulaic alpha factor mining. In *International Conference on Learning Representations (ICLR)*, 2026.

K. Miyazaki, T. Kawahara, S. Roberts, and S. Zohren. Toward Expert Investment Teams: A Multi-Agent LLM System with Fine-Grained Trading Tasks. *The Journal of Financial Data Science*, page jfds.2026.008, June 2026. ISSN 2640-3943. doi: 10.3905/jfds.2026.008. URL http://pm-research.com/lookup/doi/10.3905/jfds.2026.008.

B. Romera-Paredes, M. Barekatain, A. Novikov, M. Balog, M. P. Kumar, E. Dupont, F. J. R. Ruiz, J. S. Ellenberg, P. Wang, O. Fawzi, P. Kohli, and A. Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, Jan. 2024. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-023-06924-6. URL https://www.nature.com/articles/s41586-023-06924-6.

R. Shi, S. Yan, Y. Cai, and C. Lv. Hubble: An LLM-Driven Agentic Framework for Safe, Diverse, and Reproducible Alpha Factor Discovery, Apr. 2026a. URL http://arxiv.org/abs/2604.09601. arXiv:2604.09601.

Y. Shi, Y. Duan, and J. Li. Navigating the Alpha Jungle: An LLM-Powered MCTS Framework for Formulaic Alpha Factor Mining. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40

(2):997–1005, Mar. 2026b. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v40i2.37069. URL https://ojs.aaai.org/index.php/AAAI/article/view/37069.

A. Singhi. An Adaptive Multi-Agent Bitcoin Trading System, 2025. URL https://www.ssrn.com/abstract=5580590.

Z. Song, H. S.-H. Tsang, R. T.-C. Hsung, Y. Zhu, and W.-L. Lo. From Market Volatility to Predictive Insight: An Adaptive Transformer–RL Framework for Sentiment-Driven Financial Time-Series Forecasting. *Forecasting*, 7(4):55, Oct. 2025. ISSN 2571-9394. doi: 10.3390/forecast7040055. URL https://www.mdpi.com/2571-9394/7/4/55.

Z. Tang, Z. Chen, J. Yang, J. Mai, Y. Zheng, K. Wang, J. Chen, and L. 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 V.2*, pages 2813–2822. ACM, Aug. 2025. doi: 10.1145/3711896.3736838. URL https://dl.acm.org/doi/10.1145/3711896.3736838.

Z. Tang, X. Yin, W. Chen, Z. Chen, Y. Zheng, W. Ye, K. Wang, and L. Lin. AlphaAgentEvo: Evolution-oriented alpha mining via self-evolving agentic reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026. OpenReview lNmZrawUMu.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need, 2017. URL http://arxiv.org/abs/1706.03762. arXiv:1706.03762.

S. M. Vu, T. T. Pham, and V. H. Tran. Self-Improving Alpha Mining for Quantitative Trading via Multi-Agent Large Language Models with Knowledge Base Accumulation, 2026. URL https://www.ssrn.com/abstract=6906675.

T. Wang. Enhancing Stock Market Prediction with Temporal Graph Neural Networks and Large Language Model-Based Explainability. *Procedia Computer Science*, 274:147–160, 2025. ISSN 18770509. doi: 10.1016/j.procs.2025.12.015. URL https://linkinghub.elsevier.com/retrieve/pii/S1877050925037366.

Y. Wang, J. Xu, H. Zhang, S.-L. Huang, D. D. Sun, and X.-P. Zhang. FactorMiner: A Self-Evolving Agent with Skills and Experience Memory for Financial Alpha Discovery, Feb. 2026. URL http://arxiv.org/abs/2602.14670. arXiv:2602.14670.

Z. Weng, S. Zhang, T. Wang, and Y. Xia. AlphaLogics: A Market Logic-Driven Multi-Agent System for Scalable and Interpretable Alpha Factor Generation, Mar. 2026. URL http://arxiv.org/abs/2603.20247. arXiv:2603.20247.

Y. Wu, C. Lou, J. Zhang, S. Chen, and Y. Yang. EvoAlpha: An LLM-Enhanced Evolutionary Framework for Formulaic Alpha Mining. In *ICASSP 2026 - 2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 18572–18576. IEEE, May 2026. doi: 10.1109/ICASSP55912.2026.11463591. URL https://ieeexplore.ieee.org/document/11463591/.

A. Xin, J. Siow, J. Wang, Z. Yao, F. Zhang, J. Song, L. Hou, and J. Li. EurekAgent: Agent Environment Engineering is All You Need For Autonomous Scientific Discovery, June 2026. URL http://arxiv.org/abs/2606.13662. arXiv:2606.13662.

J. Yi, J. Yang, Y. Jin, Y. Li, and J. Li. AlphaSchema: Exploring the space of trading semantics for llm-based alpha mining, July 2026. URL http://arxiv.org/abs/2607.26642. arXiv:2607.26642.

H. Yu, Z. Zheng, J. Z. Pan, T. Liu, Z. Wang, and F. He. AlphaMemo: Structured Search-Process Memory for Self-Evolving Alpha Mining Agents, May 2026a. URL http://arxiv.org/abs/2606.20625. arXiv:2606.20625.

S. Yu, H. Xue, X. Ao, F. Pan, J. He, D. Tu, and Q. He. Generating Synergistic Formulaic Alpha Collections via Reinforcement Learning, May 2023. URL http://arxiv.org/abs/2306.12964. arXiv:2306.12964.

X. Yu, Y. Fu, M. Fan, E. Li, Y. Gao, and S. Xu. Towards autonomous formulaic alpha discovery: An evolutionary computation perspective, Aug. 2026b. URL http://arxiv.org/abs/2608.01789. arXiv:2608.01789.

L. Zhang, T. Jia, Y. Zhai, Z. Xie, C. Duan, M. He, P. S. Yu, and Y. Li. From Feedback Loops to Policy Updates: Reinforcement Fine-Tuning for LLM-Based Alpha Factor Discovery, May 2026. URL http://arxiv.org/abs/2605.15412. arXiv:2605.15412.

T. Zhang, Y. Li, Y. Jin, and J. Li. AutoAlpha: an Efficient Hierarchical Evolutionary Algorithm for Mining Alpha Factors in Quantitative Investment, Apr. 2020. URL http://arxiv.org/abs/2002.08245. arXiv:2002.08245.

Z. Zhang, S. Zohren, and S. Roberts. DeepLOB: Deep Convolutional Neural Networks for Limit Order Books, 2019. URL http://arxiv.org/abs/1808.03668. arXiv:1808.03668.

J. Zhao, C. Zhang, M. Qin, and P. Yang. QuantFactor REINFORCE: Mining Steady Formulaic Alpha Factors With Variance-Bounded REINFORCE. *IEEE Transactions on Signal Processing*, 73:2448–2463, 2025a. ISSN 1053-587X, 1941-0476. doi: 10.1109/TSP.2025.3576781. URL https://ieeexplore.ieee.org/document/11024173/.

J. Zhao, C. Zhang, C. Wang, and P. Yang. Learning from Expert Factors: Trajectory-level Reward Shaping for Formulaic Alpha Mining, July 2025b. URL http://arxiv.org/abs/2507.20263. arXiv:2507.20263.

L. Zhou, H. Ling, C. Fu, Y. Huang, M. Sun, W. Yu, X. Wang, X. Li, X. Su, J. Zhang, X. Chen, C. Liang, X. Qian, H. Ji, W. Wang, M. Zitnik, and S. Ji. Autonomous Agents for Scientific Discovery: Orchestrating Scientists, Language, Code, and Physics, 2025. URL http://arxiv.org/abs/2510.09901. arXiv:2510.09901.

A. Part I iteration record and walkthrough

This appendix gives the full version of the worked iteration summarized in Section 4. Listing 3 shows the record the system stores for a single iteration, and the text below traces two iterations in detail.

Listing 3 | One Part I iteration record. The Idea Miner emits a falsifiable proposal rather than a bare formula. The evaluator then attaches controlled tests, direction calibration, and a memory update. Variable names are representative; deployed expressions are withheld.

```
run:
  goal: "After an open-interest crash, does a weak rebound predict reversal?"
  event_type: open_interest_crash
  universe: BTCUSDT_5m

manager_plan:
  visual_search:
```

```
    trigger:
      - sharp_drop(open_interest)
      - sharp_drop(open_interest_value)
      - elevated(volume)
    context_fields:
      - price
      - volume
      - basis_rate
      - taker_buy_sell_ratio
      - long_short_account_ratio
      - top_trader_position_ratio
  target_labels: [ret_open_open_h10, ret_open_open_h30]
  instruction: "Test whether rebound quality after deleveraging separates continuation
    from failure."

visual_observation:
  event_profile:
    - open interest falls abruptly during a high-volume unwind
    - price often rebounds after forced pressure fades
    - rebounds with weak taker-flow confirmation frequently stall
    - basis recovery and positioning reset separate clean rebounds from failed ones

proposal:
  proposal_id: oi_crash_weak_rebound_flow_gap
  hypothesis: >
    After a forced open-interest unwind, a price rebound that is not confirmed
    by aggressive taker flow is more likely to fail.
  mechanism: >
    Deleveraging removes forced pressure, but weak buy-flow and poor basis
    recovery indicate insufficient demand after the rebound.
  expected_direction: higher_signal_predicts_lower_future_return
  factor_blueprint:
    - deleveraging_intensity: ranked negative change in open interest
    - rebound_strength: short-horizon price recovery after the event
    - flow_gap: lack of taker-flow confirmation during the rebound
    - basis_filter: weak or compressed basis-rate recovery
  expression: withheld

evaluation_contract:
  primary_labels:
    - ret_open_open_h10
    - ret_open_open_h30
  baselines:
    - short_horizon_price_momentum
    - open_interest_change
    - taker_flow_change
    - basis_rate_change
  controlled_tests:
    - event_window_ic_vs_control_window_ic
    - full_factor_vs_best_component
    - monthly_ic_stability
    - correlation_with_existing_factor_pool
    - proposed_direction_vs_reversed_direction

structured_observation:
  verdict: selected_for_factor_pool
  single_factor_ic_range: approximately_0.026_to_0.037
  finding: >
    The mechanism is strongest when the open-interest shock is followed by
    weak rebound acceptance and poor taker-flow confirmation.
  memory_update:
```

```
belief: "OI crash rebounds without flow confirmation are more likely to fail."
action: "Increase priority of deleveraging-plus-flow-gap mechanisms in later runs."
```

Listing 3 traces one iteration from a research question to a stored belief. The run asks whether a weak rebound after an open-interest crash predicts reversal. The AI Manager converts this question into a concrete plan: search for deleveraging episodes, inspect price and flow behavior after the unwind, and test whether rebound quality separates continuation from failure.

The Visual Analyst returns event profiles rather than a single chart. Some episodes show a clean forced-deleveraging pattern: open interest falls quickly, volume expands, basis compresses, and price rebounds only briefly. Other episodes show a healthier reset, where taker flow and basis recover with price. The Idea Miner uses this distinction to propose mechanisms such as weak rebound after deleveraging, flow-confirmed continuation, and crowded-position unwind. The exact expressions are withheld, but the generated factors combine open-interest shocks, short-horizon price response, taker-flow imbalance, basis behavior, and positioning divergence.

The Factor Evaluator then scores each proposal against multiple horizons. In this family of runs, the strongest open-interest-crash example selected by the system was an "OI crash rebound flow gap" mechanism, which tests whether a rebound after an open-interest shock is unsupported by aggressive flow. Related proposals tested whether open-interest value rises without price acceptance, whether top-trader positioning diverges from broader account ratios, and whether taker-flow confirmation changes the sign of short-horizon price continuation. The best single-factor examples in this family reached information coefficients on the order of 0.026–0.037 depending on the label and event context, and the selected signals were then passed to the combination layer.

A second iteration illustrates how the same loop changes research direction. The goal is changed to quiet-market volume expansion: the system searches for low-volatility, low-activity periods followed by a sudden burst in volume, quote volume, and trade count. The Visual Analyst rejects generic already-volatile high-volume cascades and focuses on true quiet-to-active transitions. The Idea Miner then proposes continuation factors when the burst is accepted by price, taker flow, and open interest, and reversal factors when the burst has a weak candle body, noisy flow, or no basis confirmation. This shows how the same architecture can explore a new market mechanism without changing the underlying evaluator.

B. A Part II failure case: leakage that survived agent review

The sealed sandbox of Section 3.1 is not the design AQuA started from. It is the response to concrete failures of an earlier, more permissive loop in which the agent could write feature and factor code directly and a second agent reviewed each candidate for look-ahead bias. We record the most instructive failure here, because it is the reason AQuA constrains the agent to a fixed operator registry rather than trusting review.

In the earlier loop, the agent authored each feature as code and a separate reviewer agent checked it for causality before training. One proposed feature was an intraday volume-participation ratio: the volume traded from the open up to the current minute, divided by a daily volume normalizer. The intent is causal and the description reads as backward-looking, so the reviewer agent approved it. The implementation, however, normalized by the current day's total volume, a sum that runs from the open through the close. The denominator therefore depended on bars after the current minute, and the feature quietly encoded end-of-day information into every intraday timestamp. The same failure appeared in a multi-resolution variant whose daily branch aggregated all of the current day's bars and was then read at mid-day timestamps.

The symptom was a held-out information coefficient far above what comparable price-volume features produced, and it did not survive a clean re-split of the evaluation window. A manual audit traced it to the full-day denominator. The reviewer agent had reasoned about the feature's economic intent, a ratio of past volume, rather than the exact set of bars its implementation touched, a blind spot it shared with the author agent.

The lesson is that an LLM reviewing LLM-written code is advisory, not structural: the author and the reviewer share the same failure modes, so a subtle temporal-footprint bug can pass both. AQuA's response is operatorization. The agent no longer writes feature or factor code. It composes a fixed registry of causal operators in which every time-series operator reads only a trailing window ending at the current timestamp and every cross-sectional operator reads only the current timestamp. Causality is then closed under composition, as described in Section 3.1, and a full-day normalizer is not expressible in the specification space at all. The guarantee moves from "the reviewer should catch leakage" to "leakage cannot be written."