Adversarial Information-Time Testing for Financial Machine-Learning Pipelines

A Mutation-Based Benchmark for Temporal Leakage Detection

Edmen Wong
Alpha Tick Lab Research / AQP TECH ENTERPRISE
Alpha Quant Pro

Research Paper 01 - Public Preprint - Version 0.2 - 22 August 2026

Canonical page: https://alphaquantpro.com/research/adversarial-information-time-testing
Code and benchmark: https://github.com/AlphaQuantPro/research/tree/main/papers/adversarial-information-time-testing

Status: Public preprint. Not peer reviewed.
Research scope: Methodology and controlled fault-injection benchmark. No claim of tradable alpha or live-strategy profitability.

Abstract

Temporal leakage in financial machine-learning pipelines is not limited to explicit future-row access. A backtest can violate causal ordering through completed bars indexed by their opening timestamps, preprocessing parameters estimated on future observations, globally owned feature selection, supervised labels that have not matured at the training boundary, or execution conventions that attribute decisions to prices preceding the information required to generate them.

This paper develops **Adversarial Information-Time Testing (AITT)**, a behavioral testing framework for expressing such assumptions as executable temporal contracts. For a historical cutoff T, AITT preserves all information legitimately available by T while deliberately perturbing information outside that authorized set. An artifact attributed to T - including features, preprocessing state, model-selection state, training eligibility, signals, or execution state - should remain invariant under such a mutation if the artifact is temporally non-anticipative.

The framework formalizes storage time, information-availability time, feature time, label-maturity time, selection ownership, signal time, and execution authority. It then evaluates six paired causal/leaky fixtures using synthetic data and controlled fault injection. Each fixture is tested over 200 independently generated series and three temporal cutoffs, yielding 600 trials per fixture. Across 3,600 causal-control trials, no prefix violation is observed. Targeted mutations detect all 3,000 trials involving centered temporal features, full-sample normalization, global feature selection, immature supervised labels, and open-stamped completed bars. A one-shot random mutation detects 286 of 600 deliberately retroactive same-open execution violations, while a targeted sign-flip execution adversary detects all 600.

The execution result demonstrates a central limitation of finite mutation testing: detection is conditional on mutation adequacy. A passing mutation test is therefore not proof that every leakage path is absent. AITT is intended to complement walk-forward validation, purging, embargo methods, static leakage analysis, point-in-time data controls, and conventional backtest diagnostics.

Keywords: quantitative finance; financial machine learning; look-ahead bias; temporal leakage; backtesting; information time; metamorphic testing; property-based testing; walk-forward validation; reproducibility

1. Introduction

Historical simulation assumes an arrow of time. At every simulated decision point, a research system should have access only to information that would have been available at that point in the historical process.

That requirement is frequently represented through a single timestamp, yet a real quantitative pipeline can contain several different time semantics. A database row can carry a timestamp without every field in that row having been observable at that timestamp. A supervised sample can appear before a validation period while its target still depends on observations inside that period. A model can be fitted only on historical rows while its feature set was selected using the complete dataset. A signal can be derived from a completed market event and then evaluated as if the resulting order executed at an earlier price within that same event.

These failures need not contain an obvious operation such as a negative shift. Their common structure is more general: an output is attributed to time T although some information contributing to the output becomes available only after T.

Prior work addresses related problems from several directions. Backtest-overfitting research studies the consequences of repeated strategy selection. Purging and embargo methods address overlapping information intervals. Recent financial-machine-learning studies quantify performance changes created by particular timing and preprocessing conventions. Software-engineering research has developed static leakage detection, metamorphic testing, and property-based testing for machine-learning systems. Prefix-invariance checking has also appeared in open-source quantitative tooling.

Accordingly, this paper does **not** claim that metamorphic testing, future mutation, or prefix invariance is new in isolation.

The contribution is the integration of four elements:

1. an explicit information-time model for financial research artifacts;
2. authorization-preserving mutations defined against that model;
3. artifact-level invariance tests spanning features, selection, labels, bars, and execution; and
4. a paired mutation benchmark in which causal controls and deliberately contaminated implementations are evaluated under identical synthetic processes.

The research question is therefore narrow:

> *Given an explicit information-time contract, can temporal violations in a financial research pipeline be exposed by changing only information the simulated past was not authorized to use?*

AITT treats the answer as a falsification problem rather than a performance question.

Relationship to Technical Note 01

The information-time semantics used here extend the engineering contract introduced in Wong (2026), *Bar Time Is Not Information Time: A Causality Contract for Quantitative Trading Pipelines*. That technical note separates storage time from information availability and applies the distinction to label maturity, fold-local selection, and execution timing.

The present paper does not re-claim those semantics as a new publication contribution. Its contribution is the formal mutation relation, paired causal/leaky fixtures, empirical fault-injection results, mutation-adequacy analysis, and reproducibility package built around those semantics.

2. Related Work

2.1 Backtest overfitting

Backtest overfitting concerns strategies or model configurations whose historical performance reflects repeated search rather than persistent predictive structure. Bailey et al. formalize this problem through the Probability of Backtest Overfitting and related validation procedures.

Temporal admissibility is different. A pipeline can be temporally correct and statistically overfit. Conversely, a simple model with little parameter search can still be invalid if its features or training state contain information unavailable at the simulated decision time.

AITT therefore tests one layer of research correctness rather than statistical generalization.

2.2 Label overlap, purging, and embargo

Financial supervised-learning labels often span intervals. A target such as a forward return or barrier outcome cannot be considered known when its feature vector is created. Purging and embargo methods reduce contamination caused by overlapping training and evaluation intervals.

AITT retains the same underlying information-overlap concern but makes target maturity an explicit timestamp rather than representing it only through a fixed number of rows.

2.3 Decision-time leakage

Recent work has demonstrated that evaluation timing conventions can materially alter financial-machine-learning results. Zhang et al. construct a paired benchmark around decision-time leakage, while Mir and Shrestha study introduced leakage under multiple validation regimes.

Those studies primarily measure the consequences of specified leakage mechanisms. AITT asks a complementary question: can an unauthorized dependency be exposed directly by altering information that should not influence the historical artifact?

2.4 Static, metamorphic, and property-based testing

Static analysis can detect recognizable leakage patterns from source structure. Behavioral mutation testing evaluates another surface: whether the integrated pipeline actually changes under a transformation that should be irrelevant to the authorized historical state.

Metamorphic testing is useful where a complete output oracle is unavailable. Instead of asking for the exact correct prediction, it asks whether a defined relationship between original and transformed runs holds. Property-based testing similarly evaluates general behavioral specifications across generated inputs.

AITT specializes this idea for temporal financial systems: change unauthorized future information, then verify that the authorized historical state does not change.

3. Formal Information-Time Model

3.1 Storage time and availability time

Represent a data record as:

```
d_i = (x_i, tau_i^s, tau_i^a)
```

where x_i is the recorded value, tau_i^s is the storage or indexing timestamp, and tau_i^a is the earliest time at which the information represented by x_i is legitimately available.

These times need not be equal.

For a completed bar covering an interval from open to close, final high, low, close, and volume require the completed event interval. A bar stored under its opening timestamp therefore does not become informationally available merely because the row label equals that opening timestamp.

3.2 Authorized information set

Let F_T denote the complete information set legitimately available by historical cutoff T.

An artifact A_T is temporally non-anticipative if there exists a mapping g_T such that:

```
A_T = g_T(F_T)
```

Once F_T is fixed, changing information that arrives after T cannot change A_T.

3.3 Feature time

For feature vector z_i, define feature time as the latest availability time of any dependency used to construct it:

```
tau_i^f = max tau_j^a over d_j in Dep(z_i)
```

This definition covers lags, rolling statistics, resampling, cross-sectional transforms, external datasets, and learned preprocessing state.

3.4 Label maturity

For supervised target y_i, define tau_i^l as the latest information time required to determine the label. For forward-looking targets, tau_i^l is generally later than feature time.

A strict fold-eligibility rule is therefore:

```
training_label_end_time < validation_start
```

This rule is expressed in event time rather than row distance.

3.5 Selection ownership

Let S_k denote research state associated with training fold k. This may include feature ranking, correlation filtering, dimensionality reduction, normalization parameters, hyperparameters, thresholds, ensemble weights, or model-family selection.

Fold-local ownership requires that S_k depend only on the training information set for fold k. Chronological estimator fitting is insufficient if S_k was learned globally.

3.6 Signal and execution authority

Let signal time be the earliest time at which all information required for a signal is available. Execution is valid only when the decision is operationally eligible.

```
execution_time >= eligible_time >= signal_time
```

Eligible time may incorporate quote availability, latency, spread, session rules, and event ordering. AITT does not prescribe one universal microstructure model; it requires the authority convention to be explicit.

4. Adversarial Information-Time Testing

Definition 1 - Authorization-preserving mutation

For cutoff T, a mutation M_T is authorization-preserving when it leaves the authorized information set unchanged:

```
M_T(D)|F_T = D|F_T
```

The mutation may alter any information outside F_T.

Definition 2 - Prefix invariance

Artifact A_T satisfies prefix invariance under M_T when:

```
A_T(M_T(D)) = A_T(D)
```

For floating-point artifacts, exact equality may be replaced by a justified tolerance epsilon.

Proposition 1 - Non-anticipativity implies mutation invariance

If A_T is a function of F_T alone and M_T preserves F_T, then A_T must be unchanged under M_T.

This is immediate because the input on which A_T is allowed to depend has not changed.

Practical violation criterion

For a deterministic calculation define:

```
Delta_T = ||A_T(M_T(D)) - A_T(D)||
```

A violation is recorded when Delta_T exceeds the declared numerical tolerance.

A failed invariance test has a strong interpretation. If the authorized information set has not changed but a supposedly historical artifact changes, then one of the following is true:

- the artifact depends on unauthorized information;
- nondeterministic state was not adequately controlled; or
- the declared information-time contract is incomplete.

All three outcomes require investigation.

Passing a finite mutation suite has a weaker interpretation. A dependency can remain undetected if the chosen mutation does not activate it. This limitation is measured explicitly in the execution experiment below.

5. Stage-Specific Test Relations

AITT can be applied at multiple pipeline stages.

Stage	Unauthorized information mutated	Artifact expected to remain invariant
Bar construction	raw events arriving after T	completed bars authorized by T
Feature engineering	observations after T	features with feature_time <= T

Preprocessing	validation/future observations	historical scaler/PCA state
Feature selection	validation-only relationships	training-owned selected features
Label formation	outcomes after T	eligibility of matured training labels
Model fitting	unauthorized label information	historical model artifact
Signal generation	post-decision information	signals attributed to T
Execution	post-execution information	positions/orders already attributed to T

The earliest meaningful artifact should be compared whenever possible. If the feature matrix already changes, there is little value in waiting for final portfolio PnL to reveal the problem.

6. Controlled Mutation Benchmark

6.1 Synthetic process

The benchmark uses synthetic data so causal ground truth is known by construction and no result depends on a particular market dataset.

Five feature processes are generated over N = 4096 observations. Each process follows an AR-style recursion with coefficients:

```
phi = (0.65, 0.40, 0.20, -0.25, 0.0)
```

Synthetic returns contain lagged dependencies on the first three features plus independent noise. The predictive structure is included only to keep selection and supervised-learning fixtures non-degenerate.

6.2 Repetitions

The experiment uses 200 independent pseudo-random seeds. Each realization is evaluated at three cutoffs: 0.4N, 0.6N, and 0.8N. Each fixture therefore receives 600 trials.

6.3 Paired fixtures

C1 / L1 - Rolling features
C1 uses a trailing 21-observation rolling mean from lagged observations. L1 uses a centered 21-observation window that can consume values after the time to which a feature is attributed.

C2 / L2 - Normalization
C2 estimates normalization state from the historical prefix. L2 estimates mean and scale from the full sample.

C3 / L3 - Feature-selection ownership
C3 performs feature selection only on the training prefix. L3 performs feature selection globally before historical evaluation.

C4 / L4 - Label maturity
Forward labels use a 20-observation horizon. C4 admits only labels that have fully matured by the cutoff. L4 admits rows based only on feature-row position even when their targets depend on post-cutoff outcomes.

C5 / L5 - Completed-bar availability
Raw observations are grouped into five-event bars. C5 authorizes a completed bar only after the last required event

becomes available. L5 treats the completed bar as available at its opening storage timestamp.

C6 / L6 - Execution authority
C6 permits a close-derived signal to affect a position from the following open onward. L6 uses the current day's close to determine a position retrospectively recorded at that same day's open.

6.4 Mutation operators

Future feature values receive large Gaussian perturbations for rolling and normalization tests. Feature-selection tests construct a strong predictive relationship in a fifth feature only after the cutoff. Label-maturity tests perturb future returns. Completed-bar tests perturb future raw events used by an incorrectly open-stamped bar.

The execution experiment uses two adversaries. The first randomly perturbs an unavailable close. The second deliberately moves the unavailable current close to the opposite side of the current open, guaranteeing a decision-sign change in the planted retroactive-execution mutant.

The targeted adversary is not intended to model a market distribution. It is a fault-injection operator designed to test whether the mutant is killable.

7. Results

7.1 Primary benchmark

Fixture	Status	Trials	Violations	Rate
Trailing rolling mean	Causal	600	0	0.000
Centered rolling mean	Leaky	600	600	1.000
Expanding normalization	Causal	600	0	0.000
Full-sample normalization	Leaky	600	600	1.000
Fold-local feature selection	Causal	600	0	0.000
Global feature selection	Leaky	600	600	1.000
Maturity-aware training	Causal	600	0	0.000
Row-index-only label eligibility	Leaky	600	600	1.000
Availability-aware completed bar	Causal	600	0	0.000
Open-stamped completed bar	Leaky	600	600	1.000
Next-open execution	Causal	600	0	0.000
Same-open retro-execution - random mutation	Leaky	600	286	0.477

Across all six causal controls, 0 of 3,600 trials violate their expected historical invariance relation.

For L1 through L5, targeted mutations detect 3,000 of 3,000 planted leakage trials.

These rates characterize only the specified synthetic benchmark and must not be interpreted as universal false-positive or detection rates for arbitrary production pipelines.

7.2 Mutation adequacy: execution result

The execution fixture provides the most useful negative result.

A random perturbation of an unavailable close does not necessarily reverse the sign of the decision derived from that close. Consequently, only 286 of 600 planted retroactive-execution faults trigger a historical-state change under that mutation.

When the mutation is replaced by a targeted sign flip, detection rises to 600 of 600.

The interpretation is important:

> *A non-anticipativity test can reveal only a dependency that the chosen mutation actually excites.*

A passing result under one weak mutation is therefore not strong evidence of causal correctness. Mutation families require coverage and adequacy analysis just as ordinary software test suites do.

8. Why Walk-Forward Validation Is Not Sufficient

Suppose an estimator is refitted chronologically at each fold. That does not prove the entire research process is chronological if feature selection, preprocessing, dimensionality reduction, threshold tuning, or model-family selection were performed globally beforehand.

AITT tests ownership behaviorally. If validation-only information is mutated, training-owned selection state should remain unchanged. The same relation can be applied to scaler parameters, PCA loadings, hyperparameters, thresholds, and ensemble weights.

Walk-forward validation, purging, and AITT therefore answer different questions and should be used together rather than treated as substitutes.

9. Label Maturity as an Information Interval

Consider a feature row at time t with a forward horizon h. The feature row can be historical while the label still depends on outcomes after the validation boundary.

Using row position alone can therefore admit an observation whose target is not yet known at the declared training cutoff. An explicit label maturity time is stronger evidence than a fixed purge distance when horizons are irregular, event driven, or affected by missing sessions.

AITT adds a behavioral check: if supposedly historical training state changes when only immature future outcomes are perturbed, the training eligibility rule is not respecting target maturity.

10. Deployment as a Research Validation Gate

AITT is intended to run before performance claims are evaluated.

A practical sequence is:

```
Raw data
 -> availability-time validation
 -> feature construction
 -> AITT feature tests
 -> label-maturity checks
 -> fold-local selection
 -> AITT selection tests
 -> model fitting
 -> signal generation
 -> execution validation
 -> performance evaluation
```

A minimal implementation can execute the baseline pipeline, apply a declared unauthorized-information mutation, re-run with identical deterministic state, and compare the historical artifacts that should be invariant.

For deterministic artifacts, hashes may be retained to turn temporal-causality regressions into standard CI pass/fail checks.

11. Limitations and Threats to Validity

AITT has several important limitations.

First, the benchmark is synthetic. Synthetic experiments provide known causal ground truth but do not estimate how frequently the tested failure modes occur in real production pipelines.

Second, the experiment is a controlled fault-injection benchmark. The leaky fixtures are intentionally constructed and the mutations are designed around those mechanisms. High mutant-kill rates are not universal detection accuracy.

Third, finite mutation suites are incomplete. The execution experiment demonstrates that a real planted dependency can survive a mutation that does not activate the relevant decision path.

Fourth, the framework assumes the historical source snapshot itself is point-in-time valid. If a vendor retrospectively revises a historical field while preserving an earlier timestamp, contamination may already exist inside the apparent prefix and cannot necessarily be detected by mutating only post-cutoff rows.

Fifth, stochastic systems require careful control. GPU nondeterminism, randomized models, parallel reductions, asynchronous execution, or unpinned environments can create artifact differences unrelated to temporal leakage.

Sixth, temporal admissibility does not imply economic validity. AITT does not establish predictive skill, profitability, transaction-cost realism, market capacity, robustness to regime change, independence from multiple testing, or live-trading performance.

12. Reproducibility

The reference implementation contains no proprietary market data or client information. The public package contains the synthetic generator, paired fixtures, mutation operators, reference environment, trial-level outputs, summary results, and targeted execution-adversary results.

Reference implementation:
https://github.com/AlphaQuantPro/research/tree/main/papers/adversarial-information-time-testing

The publication version should be linked to a persistent public archive when available. A versioned archive and cryptographic hash should bind the manuscript to the exact benchmark revision used for reported results.

13. Discussion

The main value of AITT is not another warning that researchers should avoid look-ahead bias. That principle is already well established.

The methodological change is that temporal claims can be translated into falsifiable software properties.

A claim that a feature uses only historical information can be challenged by mutating unauthorized future observations. A claim that feature selection is fold-local can be challenged by altering validation-only relationships. A claim that a label is eligible for training can be challenged by mutating outcomes after the feature timestamp but before label maturity. A claim that an execution price was attainable can be challenged by modifying information required to generate the corresponding signal.

These tests do not replace code review, statistical validation, realistic simulation, or point-in-time data controls. They provide an additional behavioral surface on which causal assumptions can fail visibly.

14. Conclusion

Financial backtests are meaningful only when their historical state respects the information that was actually available at each simulated time.

Storage time is not necessarily information time. Chronological model fitting does not guarantee chronological model selection. A historical feature row does not imply that its label has matured. A signal timestamp does not imply

that an earlier execution price was attainable.

AITT expresses these assumptions as behavioral invariants.

```
If only unauthorized future information changes,
authorized historical artifacts must not change.
```

In the controlled benchmark presented here, the framework distinguishes paired causal and contaminated implementations across rolling features, normalization, feature selection, label maturity, completed-bar timing, and execution authority. The execution experiment also establishes a necessary caution: a weak mutation can fail to activate a genuine leakage path.

AITT should therefore be interpreted as adversarial falsification, not proof.

The broader proposal is simple: before historical performance is treated as evidence of strategy quality, the research system should make its information-time assumptions explicit and attack those assumptions with executable tests.

Data Availability

All reported experiments use synthetic data generated by the public reference implementation. No proprietary market, broker, or client data are required.

Code Availability

Reference benchmark code and generated result tables are published at:

https://github.com/AlphaQuantPro/research/tree/main/papers/adversarial-information-time-testing

Competing Interests

The author is the founder of AQP TECH ENTERPRISE, which develops Alpha Tick Lab and Alpha Quant Pro. This paper concerns research-validation methodology and does not report, audit, or establish the investment performance of any commercial trading strategy.

Research Status

Public preprint. Not peer reviewed. The experiments are methodological fault-injection tests and are not evidence of tradable alpha.

References

1. Bailey, D. H., Borwein, J. M., Lopez de Prado, M., & Zhu, Q. J. (2017). *The Probability of Backtest Overfitting*. Journal of Computational Finance. https://doi.org/10.21314/JCF.2016.322
2. Lopez de Prado, M. (2018). *Advances in Financial Machine Learning*. Wiley.
3. Zhang, F., Li, Z., Peng, S., & Chen, Y. (2026). *When Alpha Disappears: A One-Switch Benchmark for Decision-Time Leakage in Financial Backtests*. arXiv:2605.23959.
4. Mir, Z. A., & Shrestha, D. (2026). *Quantifying Backtest Overfitting from Information Leakage: A Walk-Forward and Embargo-Based Diagnostic Framework Applied to Deep Learning Return Forecasts*. SSRN. https://doi.org/10.2139/ssrn.7029819
5. Yang, C., Brower-Sinning, R. A., Lewis, G. A., & Kastner, C. (2022). *Data Leakage in Notebooks: Static Detection and Better Processes*. ACM/IEEE ASE. https://doi.org/10.1145/3551349.3556918

6. Xie, X., Ho, J. W. K., Murphy, C., Kaiser, G., Xu, B., & Chen, T. Y. (2011). *Testing and validating machine learning classifiers by metamorphic testing*. Journal of Systems and Software, 84(4), 544-558. https://doi.org/10.1016/j.jss.2010.11.920
7. Wauters, C., & De Roover, C. (2024). *Property-based Testing within ML Projects: An Empirical Study*. IEEE ICSME. https://doi.org/10.1109/ICSME58944.2024.00067
8. Wong, E. (2026). *Bar Time Is Not Information Time: A Causality Contract for Quantitative Trading Pipelines*. Technical Note 01, v1.0. Alpha Tick Lab Research / AQP TECH ENTERPRISE. DOI: 10.5281/zenodo.21965788