

ASRQuant: From Scientific Literature to Auditable Quantitative Decisions and Algorithmic Trading

A Unified Python Laboratory for Hypothesis Discovery, Data, Modelling, Backtesting, Robustness, and Paper Execution

Alpha Kabinet TOURE
Alpha Stochastic Research
`research@asr-lab.online`

Version 0.5.0 — August 1, 2026

Abstract

Quantitative-finance research often begins in scientific papers but becomes fragmented when researchers manually translate claims into data requirements, features, signals, portfolios, backtests, robustness checks, and deployment decisions. This fragmentation weakens provenance: the final strategy may no longer retain the exact paper page, hypothesis statement, information lag, transformation, execution contract, or decision rule that produced it. We introduce **ASRQuant**, an open-source Python framework that unifies scientific-paper ingestion, corpus-relative hypothesis discovery, explicit data planning, leakage-aware features, signal and portfolio construction, robust econometric testing, auditable backtesting, multidimensional visualization, decision governance, and order-level paper trading through a one-import public interface. Version 0.5.0 parses text-based PDF papers with page-level citations, classifies extracted claims as established, replicated, underexplored, contradictory, or corpus-novel within the supplied corpus, and explicitly warns that corpus novelty is not proof of global novelty. A stateful **ResearchProject** preserves the complete transformation chain from source excerpts to a machine-readable manifest. The quantitative layer retains local and remote data adapters, stochastic simulation, Monte Carlo, martingale diagnostics, derivative pricing, econometrics, leakage-aware machine learning, portfolio and risk analytics, fixed income, more than 100 visualization entry points, implementation audits, and multidimensional parameter animations. The new paper-trading layer converts target weights into orders and fills while enforcing leverage, position, turnover, cash, short-selling, and drawdown controls; live routing remains disabled unless an explicit external broker adapter is supplied. The artifact is verified by 64 automated tests. ASRQuant remains research software: automated extraction does not establish causality or global novelty, backtests do not guarantee future profitability, and a research-governance decision is not personalized investment advice.

Keywords: quantitative finance; scientific literature; hypothesis discovery; economic hypotheses; stochastic processes; Monte Carlo; martingales; derivative pricing; econometrics; machine learning; backtesting; robustness; algorithmic trading; paper trading; reproducibility; Python.

Contents

1	Introduction	5
1.1	Research question	5
1.2	Contributions	6
2	Related Work	7
2.1	Backtesting systems	7
2.2	Performance and risk analytics	7
2.3	Backtest overfitting and multiple testing	7

2.4	Implementation risk and design uncertainty	7
2.5	Scientific Python and visualization	8
3	Design Principles	8
3.1	Explicit semantics over convenient ambiguity	8
3.2	A short public API over duplicated notebooks	8
3.3	Pandas-compatible inputs and outputs	8
3.4	Deterministic transformations	8
3.5	No hidden forward filling	8
3.6	Warnings should be structural	8
3.7	Visualization must preserve the underlying object	8
3.8	Research scope must remain honest	8
4	Formal Backtest Contract	9
4.1	Market data and returns	9
4.2	Target and effective weights	9
4.3	Constraints	9
4.4	Turnover	9
4.5	Transaction and financing costs	9
4.6	Cash and portfolio return	10
4.7	Rebalancing	10
4.8	Contract fingerprint	10
5	Implementation-Sensitivity Diagnostics	10
5.1	Implementation uncertainty interval	10
5.2	Engine sensitivity	10
5.3	Conclusion stability index	11
6	Software Architecture	11
7	Visualization Taxonomy	13
8	Performance and Risk Metrics	13
9	Regression and Statistical Diagnostics	14
9.1	OLS and robust covariance	14
9.2	Rolling regression	14
9.3	Stationarity tests	15
9.4	Dependent-data bootstrap	15
9.5	Multiple testing	15
9.6	Extended regression and time-series models	15
9.7	Leakage-aware machine learning	15
10	Time-Aware Validation and Leakage Controls	15
11	Data Ingestion and Near-Real-Time Feeds	16
12	Scientific Literature and Hypothesis Provenance	16
13	Hypothesis-to-Decision Research Workflow	17
14	Algorithmic Trading and Paper Execution	17

15 Stochastic Processes and Monte Carlo	18
15.1 Arithmetic and geometric Brownian motion	18
15.2 Mean reversion, stochastic volatility, and jumps	18
15.3 Monte Carlo estimators and uncertainty	18
16 Martingale Diagnostics	19
17 Derivative Pricing	19
17.1 Black–Scholes–Merton and Black–76	19
17.2 Bachelier normal pricing	19
17.3 Trees, implied volatility, and Monte Carlo	19
18 Portfolio, Volatility, Risk, and Fixed Income	20
19 Reproducibility and Provenance	20
20 Empirical Evaluation	21
20.1 Objectives	21
20.2 Synthetic market	21
20.3 Reference strategy	21
20.4 Reference result	21
20.5 Implementation audit: stable negative conclusion	22
20.6 Implementation audit: fragile conclusion	23
20.7 Parameter landscapes and multidimensional animation	24
20.8 Runtime scaling	26
20.9 Independent numerical model validation	27
21 Verification and Testing	27
22 Comparison with Existing Workflows	28
23 Limitations	28
23.1 Execution realism	28
23.2 Corporate actions and point-in-time data	29
23.3 Transaction-cost calibration	29
23.4 Statistical inference	29
23.5 Visualization risk	29
23.6 Optimization instability	29
23.7 Package maturity	29
24 Ethical and Practical Use	29
25 Roadmap	29
26 Conclusion	30
A Five-Line Workflows	30
A.1 CSV to audited backtest	30
A.2 Remote historical data	30
A.3 Monte Carlo and martingale diagnostics	31
A.4 Unified option pricing	31
A.5 In-memory custom strategy	31
A.6 Implementation audit	31
A.7 Regression diagnostics	31

A.8 Option surface	31
A.9 Scientific papers to hypothesis registry	32
A.10 Hypothesis to governed decision	32
A.11 Paper trading with risk controls	32
B Public API Catalog	33
C Reproducibility Checklist	33
D Artifact Structure	34

1 Introduction

Historical simulation is the central experimental instrument of systematic investment research. A strategy is transformed into positions, positions interact with realized returns, costs reduce gross performance, and a collection of statistics and visual diagnostics is used to judge whether the result is economically and statistically credible. The apparent simplicity of this workflow conceals many implementation choices. A signal observed at the close of day t may be applied to the return from $t - 1$ to t , to the return from t to $t + 1$, or to an execution price at the next open. Portfolio weights may be interpreted as pre-trade or post-trade weights. Missing observations may be dropped, forward-filled, or treated as non-tradable. Turnover may be computed before or after normalization. Costs may be charged once per rebalance, per side, or as a nonlinear function of traded notional. Each choice can alter the result even if the verbal strategy description is unchanged.

The statistical literature has extensively studied data snooping, multiple testing, inflated Sharpe ratios, and backtest overfitting [16, 5, 2, 3, 6]. More recent work has isolated implementation risk: the dispersion induced by differences between software engines or conventions rather than by differences in the logical strategy [17]. This paper takes a software-engineering position on that problem. Instead of treating implementation choices as incidental keyword arguments scattered throughout code, ASRQuant represents them as a first-class, immutable, serializable contract.

The objective is not merely to provide another plotting library or another backtesting engine. The objective is to make a broad quantitative-research workflow both *short to write* and *difficult to misinterpret*. A minimal analysis is intentionally compact:

```
import asrquant as asr
lab = asr.open_lab("prices.csv", date_column="Date")
result = lab.backtest("sma", fast=20, slow=100, costs_bps=5)
asr.save(result, "dashboard.png", kind="dashboard")
asr.report(result, "report.html")
```

However, the apparent simplicity is backed by an explicit default contract: one-bar execution delay, positive-price validation, deterministic weight alignment, leverage enforcement, transparent turnover, explicit basis-point costs, and a machine-readable experiment fingerprint. The user can override these choices, but the choices remain inspectable.

1.1 Research question

The central research question is:

Can a unified quantitative-finance library connect scientific literature to auditable quantitative decisions and paper execution in a few lines of code while preserving source provenance, explicit chronology, reproducibility, statistical discipline, and broad visual coverage?

The software and empirical evaluation address six subsidiary questions:

1. Can backtest semantics be represented in a compact contract that is both human-readable and machine-verifiable?
2. Can one high-level API cover market data, performance, risk, regression, portfolio, derivatives, machine learning, microstructure, and multidimensional parameter analysis through two-axis surfaces, slices, and animations without forcing the user to rewrite common formulas?
3. Can implementation sensitivity be surfaced as a standard output rather than an optional afterthought?
4. Can this functionality remain computationally practical on medium-scale research panels?
5. Can source-linked hypotheses and literature gaps be represented without making unsupported claims of global novelty?
6. Can research evidence be converted into a governed decision and order-level paper-trading record without silently authorizing live execution?

1.2 Contributions

The release makes the following contributions.

1. **A source-linked scientific-literature layer.** Text-based PDF papers are parsed page by page; extracted hypotheses retain source excerpts, page numbers, paper fingerprints, confidence scores, and corpus-relative novelty labels.
2. **A literature-to-decision research state machine.** A `ResearchProject` records the hypothesis, data plan, features, signal, portfolio, econometric test, backtest, robustness results, decision, paper-trading session, and chronological workflow history.
3. **An end-to-end quantitative research object.** `QuantLab` creates one path from files or remote providers to transformations, models, backtests, audits, figures, and reports.
4. **A provider-neutral data layer.** The same pandas contract accepts CSV, Parquet, Excel, JSON, Feather, SQL, Alpha Vantage, Binance, FRED, and optional Yahoo data, with explicit field selection and source metadata.
5. **A stochastic-process and Monte Carlo laboratory.** Seven process families, reproducible random seeds, antithetic sampling, pricing standard errors, and confidence intervals are exposed through a common result object.
6. **Integrated derivative analytics.** Black–Scholes–Merton, Bachelier, Black–76, CRR trees, implied volatility, analytic and numerical Greeks, European Monte Carlo, and arithmetic-average Asian Monte Carlo are available without external pricing code.
7. **Finite-sample martingale diagnostics.** Discounting, mean-increment tests, HAC predictive regressions, and Ljung–Box diagnostics make the martingale hypothesis inspectable while explicitly avoiding the false claim that finite data can prove it.
8. **A specification-first backtest contract.** Execution delay, rebalancing, leverage, missing-data policy, annualization, financing rate, and costs are encoded in an immutable `BacktestSpec`.
9. **Econometrics and leakage-aware machine learning.** Robust and rolling OLS, quantile and polynomial regression, regularization, logistic and factor models, cointegration, Granger-predictive tests, ARIMA, VAR, technical features, forward targets, and chronological walk-forward evaluation share aligned outputs.
10. **Broad risk, allocation, volatility, and fixed-income coverage.** The package includes historical and parametric tail risk, drawdown duration, CDaR, covariance shrinkage, maximum diversification, HRP, Black–Litterman, realized and range-based volatility, GARCH integration, bond pricing, yield, duration, convexity, and curve bootstrapping.
11. **Implementation-risk auditing and reproducibility.** Alternative delays, costs, and rebalance rules generate uncertainty intervals and conclusion-stability statistics; deterministic fingerprints and manifests record the experiment.
12. **A multidimensional parameter explorer.** Any scalar-valued experiment on a finite grid can be reduced to two selected surface axes; any number of remaining numeric or categorical parameters can be fixed or flattened into reproducible animation frames, with dotted-path metric extraction, parallel evaluation, failure metadata, best-point extraction, and HTML/GIF/MP4 export.
13. **A tested visualization system.** More than 100 public visualization functions and result-object plotting methods cover market data, stochastic paths, martingales, regressions, ML, risk, portfolios, derivatives, microstructure, heatmaps, contours, interactive 3D surfaces, and parameter animations.
14. **A broker-neutral paper-execution layer.** Target weights are translated into orders, fills, cash, positions, realized weights, and risk events under explicit leverage, position, turnover, order-notional, cash, short-selling, commission, slippage, partial-fill, and drawdown controls.

2 Related Work

2.1 Backtesting systems

Python backtesting systems occupy several design points. Event-driven frameworks model orders, fills, and broker interactions in detail. Vectorized frameworks represent signals and positions as arrays, enabling fast parameter sweeps. Performance-analysis packages produce tear sheets from precomputed returns. Examples include Backtrader, Zipline, backtesting.py, vectorbt, pyfolio, empyrical, and QuantStats. These projects have substantially lowered the barrier to quantitative experimentation. Nevertheless, they generally specialize in one layer of the workflow: execution simulation, analytics, or reporting. **ASRQuant** adopts a complementary position. It provides a lightweight reference backtester, but its primary abstraction is the experiment contract and the consistent propagation of that contract into audit outputs and reports.

Vectorized computation is particularly appropriate when the strategy can be represented as target weights over a common time index. Array-oriented systems can evaluate many assets or parameter configurations efficiently, while event-driven systems are more natural for intrabar order logic, partial fills, and exchange-specific mechanics. **ASRQuant** therefore does not claim to replace a production execution simulator. It deliberately targets research-grade daily or bar-based portfolio experiments in which signal timestamps and target weights can be made explicit.

2.2 Performance and risk analytics

The Sharpe ratio and related risk-adjusted statistics are standard in investment evaluation [14]. Their interpretation requires care under serial dependence and non-normality [9]. Drawdown, Sortino, Calmar, Omega, Value at Risk, and Expected Shortfall provide complementary views of the return distribution and path. Existing analytics packages automate many of these calculations. **ASRQuant** integrates these measures with turnover, costs, implementation warnings, and the exact specification fingerprint used to produce them.

2.3 Backtest overfitting and multiple testing

White’s Reality Check and Hansen’s Superior Predictive Ability test address data-snooping bias when many models are compared [16, 5]. The Deflated Sharpe Ratio corrects an observed Sharpe ratio for multiple selection and non-normal returns [2]. The Probability of Backtest Overfitting formalizes the likelihood that a selected in-sample configuration underperforms out of sample [3]. Harvey, Liu, and Zhu show that conventional significance thresholds are too weak when hundreds of factors are tested [6]. **ASRQuant** does not claim to fully automate all possible multiple-testing procedures in version 0.5.0, but it includes probabilistic and deflated Sharpe diagnostics, block bootstrap confidence intervals, permutation tests, and Benjamini–Hochberg false-discovery-rate control.

2.4 Implementation risk and design uncertainty

Yin et al. distinguish statistical overfitting from engine-level disagreement and propose metrics for implementation risk [17]. Their results show that transaction-cost details can create structured divergence across engines. Related 2026 work studies design uncertainty across implementation choices [4]. **ASRQuant** operationalizes this concern at the API level: users can call `lab.audit(...)` on the same target weights and inspect the distribution of performance across specified conventions. We use the neutral term *contract sensitivity* because the grid may contain both defensible alternatives and intentionally unrealistic diagnostics such as same-bar execution.

2.5 Scientific Python and visualization

The implementation builds on NumPy, pandas, SciPy, Matplotlib, Plotly, statsmodels, and scikit-learn [7, 11, 15, 8, 12, 13]. The contribution is not a replacement for these libraries; it is a finance-specific composition layer with standardized inputs, outputs, formulas, and audit metadata.

3 Design Principles

ASRQuant follows eight design principles.

3.1 Explicit semantics over convenient ambiguity

A backtest that cannot state when a signal becomes tradable is incomplete. Defaults are allowed, but defaults must be documented, inspectable, and included in the experiment fingerprint.

3.2 A short public API over duplicated notebooks

Common formulas should not be manually reimplemented in each project. The high-level API should cover the common path, while lower-level functions remain available for specialists.

3.3 Pandas-compatible inputs and outputs

A `Series` or `DataFrame` with a time index is the standard interchange format. This choice facilitates integration with data vendors, statistical packages, and notebooks.

3.4 Deterministic transformations

Given identical data, configuration, and package version, the engine should produce identical results. Random procedures require an explicit random seed.

3.5 No hidden forward filling

Price returns are computed with `fill_method=None`. Missing-data policies are selected before return calculation. This avoids silently creating zero returns across gaps.

3.6 Warnings should be structural

Same-bar execution is not forbidden because it can be useful as a diagnostic or for signals known before the bar begins. It is, however, marked in the result metadata and HTML report as a potential look-ahead risk.

3.7 Visualization must preserve the underlying object

Plot functions return figure objects rather than forcing display. Researchers can save, modify, test, or embed the figures.

3.8 Research scope must remain honest

The package is a research framework. Version 0.5.0 models basic order states and configurable partial fills in a deterministic paper broker, but it does not model exchange queue priority, venue latency, market halts, borrow recalls, corporate-action databases, or broker-specific live behavior.

4 Formal Backtest Contract

4.1 Market data and returns

Let $P_{t,i} > 0$ be the observed price of asset $i \in \{1, \dots, N\}$ at timestamp $t \in \{0, \dots, T\}$. Simple returns are

$$r_{t,i} = \frac{P_{t,i}}{P_{t-1,i}} - 1, \quad t = 1, \dots, T. \quad (1)$$

No return is defined across an unresolved missing observation. The data policy is one of:

1. **raise**: reject any missing price;
2. **drop**: retain only rows complete across the selected assets;
3. **ffill**: forward-fill first, then remove unresolved leading gaps.

The policy is part of the contract and therefore part of the specification fingerprint.

4.2 Target and effective weights

Let $z_{t,i}$ be the target weight produced by the strategy at timestamp t . Let $\delta \in \{0, 1, 2, \dots\}$ denote the execution delay in bars. Effective weights for the return ending at t are

$$w_{t,i} = z_{t-\delta,i}, \quad (2)$$

with unavailable pre-sample targets set to zero. Under the default $\delta = 1$, a signal formed at time t affects the next bar's return. Setting $\delta = 0$ applies the current target to the current return and must be justified by the information timestamp.

4.3 Constraints

For maximum absolute asset weight $u > 0$ and maximum gross leverage $L > 0$, raw targets are clipped and rescaled so that

$$|w_{t,i}| \leq u, \quad \sum_{i=1}^N |w_{t,i}| \leq L. \quad (3)$$

If `long_only=True`, negative targets are set to zero before leverage normalization. This is a deterministic projection, not a general constrained optimizer.

4.4 Turnover

One-way portfolio turnover is defined as

$$\tau_t = \sum_{i=1}^N |w_{t,i} - w_{t-1,i}|. \quad (4)$$

This convention counts a complete rotation from one fully invested asset to another as turnover 2. Because industry turnover definitions vary, the report labels the quantity explicitly as fraction of net asset value traded.

4.5 Transaction and financing costs

Let b be the sum of commission, spread, and slippage rates in basis points. Let $\kappa \geq 0$ and $\gamma \geq 1$ parameterize nonlinear market impact. Let q be the annualized short-borrow rate in basis points and A the annualization factor. The period cost is

$$C_t = \frac{b}{10^4} \tau_t + \kappa \tau_t^\gamma + \frac{q}{10^4 A} \sum_{i=1}^N \max(-w_{t,i}, 0). \quad (5)$$

This model is intentionally transparent. It is not a substitute for an empirically calibrated execution model based on volume, volatility, spread, and participation rate. Market-impact theory motivates nonlinear cost functions [1], but calibration remains asset- and venue-specific.

4.6 Cash and portfolio return

The residual cash weight is

$$c_t = 1 - \sum_{i=1}^N w_{t,i}. \quad (6)$$

For annualized risk-free rate r_f , gross and net portfolio returns are

$$R_t^{\text{gross}} = \sum_{i=1}^N w_{t,i} r_{t,i}, \quad (7)$$

$$R_t^{\text{net}} = R_t^{\text{gross}} + c_t \frac{r_f}{A} - C_t. \quad (8)$$

Equity evolves from initial capital V_0 according to

$$V_t = V_0 \prod_{s=1}^t (1 + R_s^{\text{net}}). \quad (9)$$

4.7 Rebalancing

A target panel can be applied every bar or sampled at a calendar rule such as weekly, monthly, or quarterly. After sampling, targets are carried forward until the next rebalance. The distinction between signal refresh and execution refresh is important: a daily signal evaluated under monthly rebalancing is not equivalent to a genuinely monthly signal.

4.8 Contract fingerprint

The specification is serialized as sorted JSON and hashed with SHA-256. The data frame is separately fingerprinted from values, timestamps, and column names. The experiment identifier combines both fingerprints. The hash is not a proof of data correctness; it is an identity check that detects changes to the analyzed object or contract.

5 Implementation-Sensitivity Diagnostics

Let $\mathcal{C} = \{c_1, \dots, c_K\}$ be a set of backtest contracts applied to the same prices and logical target weights. For a metric m , write $\theta_m(c_k)$ for the result under contract c_k .

5.1 Implementation uncertainty interval

$$\text{IUI}_m = \left[\min_{c \in \mathcal{C}} \theta_m(c), \max_{c \in \mathcal{C}} \theta_m(c) \right]. \quad (10)$$

This interval is descriptive. It does not have a sampling-probability interpretation unless the contract set itself is generated from a probabilistic design.

5.2 Engine sensitivity

$$\text{ES}_m = \max_{c \in \mathcal{C}} \theta_m(c) - \min_{c \in \mathcal{C}} \theta_m(c). \quad (11)$$

A large value indicates that the reported metric is highly conditional on implementation choices.

5.3 Conclusion stability index

For a directional conclusion based on total return, define

$$\text{CSI} = \left| \frac{1}{K} \sum_{k=1}^K \text{sign}(\theta_{\text{TR}}(c_k)) \right|. \quad (12)$$

The index equals one when all contracts agree on the sign and approaches zero when positive and negative conclusions are balanced. This simple release metric is related in spirit to conclusion-sensitivity measures in recent implementation-risk research [17]; future versions will support user-defined decision rules and equivalence margins.

6 Software Architecture

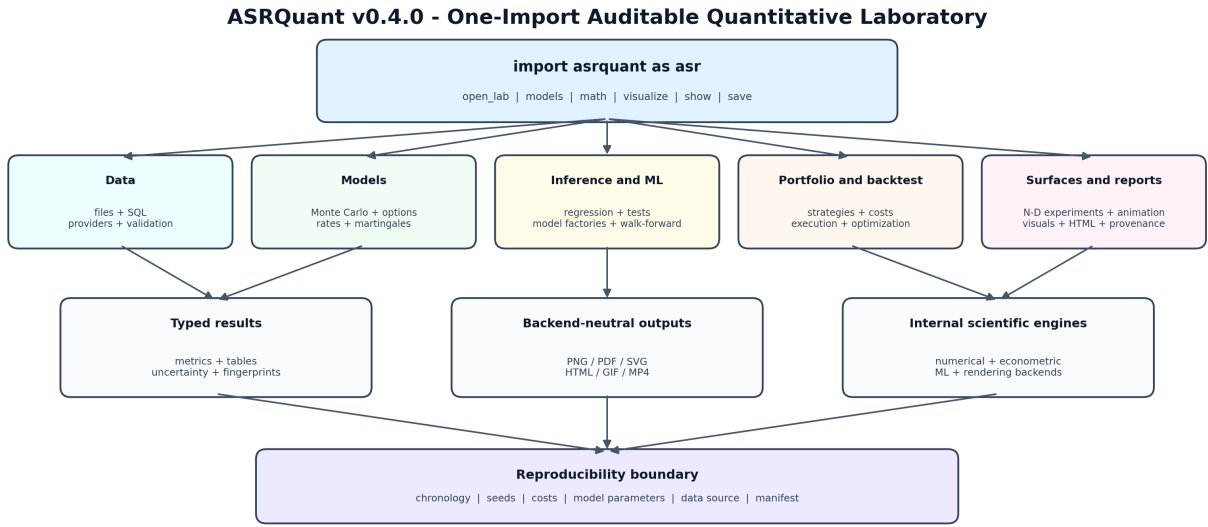


Figure 1: ASRQuant v0.5.0 architecture. The public API is intentionally short, but each output retains the data source, model parameters, research contract, and uncertainty information required for audit.

Table 1: Principal modules in version 0.5.0.

Module	Responsibility
<code>easy.py</code> , <code>models.py</code> , <code>math.py</code>	One-import facade, backend-neutral figure handles, friendly estimator factories, model-name resolution, table constructors, and numerical helpers.
<code>api.py</code>	High-level QuantLab : files, providers, models, regression, ML, pricing, backtests, sweeps, figures, and reports.
<code>data.py</code> , <code>providers.py</code>	Local and SQL loading, validation, OHLCV aggregation, quality reports, fingerprints, provider adapters, and polling feeds.
<code>simulation.py</code> , <code>martingales.py</code>	ABM, GBM, correlated GBM, OU, CIR, Vasicek, Heston, Merton, bootstrap, Monte Carlo pricing, and martingale diagnostics.
<code>derivatives.py</code>	BSM, Bachelier, Black–76, CRR, implied volatility, payoffs, analytic and finite-difference Greeks.
<code>statistics.py</code> , <code>machine_learning.py</code>	Econometrics, inference, time-series models, feature engineering, forward targets, and walk-forward evaluation.
<code>strategies.py</code> , <code>backtest.py</code> , <code>audit.py</code> , <code>research.py</code>	Nine strategy constructors, vectorized simulation, costs, trades, comparisons, implementation audits, and parameter sweeps.
<code>metrics.py</code> , <code>volatility.py</code>	Performance, benchmark, drawdown, tail-risk, Sharpe-quality, realized, range-based, EWMA, and optional GARCH diagnostics.
<code>optimization.py</code> , <code>fixed_income.py</code>	Covariance estimators, classical allocation, ERC, maximum diversification, HRP, Black–Litterman, bonds, duration, convexity, and curves.
<code>surfaces.py</code>	N-dimensional finite-grid evaluation, metric extraction, frame tables, static slices, Plotly animations, GIF/MP4 export, and experiment safeguards.
<code>report.py</code> , <code>provenance.py</code> , <code>viz/*</code>	Portable reports, manifests, and more than 100 static or interactive visual functions and plotting methods.

The architecture separates the public research contract from implementation engines. The recommended user surface is a single import, `import asrquant as asr`. The `models` namespace creates validated estimators, `math` provides numerical and distribution helpers, and `visualize`, `show`, and `save` wrap static and interactive rendering behind a backend-neutral handle. Calculations return scalars, compatible tables, or typed result containers. Mature numerical, econometric,

machine-learning, and visualization libraries remain internal because reimplementing them would increase numerical and maintenance risk; ordinary workflows do not import or orchestrate them directly. Network access is isolated in provider adapters; no import or model operation silently performs a remote request. Remote acquisition and near-real-time polling are data functions only and cannot place orders.

7 Visualization Taxonomy

Version 0.5.0 contains more than 100 public visualization functions and plotting methods, excluding private helpers. Table 2 summarizes the coverage.

Table 2: Visualization coverage.

Category	Included diagnostics
Market data	Prices, normalized prices, log scale, returns, cumulative returns, dependency-light candlesticks, rolling mean and volatility, distribution with KDE and normal overlay, ECDF, Q–Q, box and violin plots, ACF, PACF, static and rolling correlation, monthly and daily-calendar heatmaps, scatter matrix.
Performance	Equity, equity plus drawdown, dashboard, rolling Sharpe and Sortino, annual and monthly returns, turnover, costs, gross/net exposure, weights, return contributions, benchmark comparison, rebalance-bar returns.
Risk	Historical, Gaussian, and Cornish–Fisher VaR; Expected Shortfall; rolling tail risk; empirical loss tails; CDaR; drawdowns; stress bars; sensitivity heatmaps; Monte Carlo fans; contract-audit plots.
Stochastic processes	Path ensembles, uncertainty fans, terminal distributions, Monte Carlo convergence, and martingale diagnostic panels.
Regression	Regression scatter, actual versus fitted, residual four-panel diagnostics, coefficient intervals, rolling coefficients, factor-exposure heatmaps.
Portfolio	Efficient-frontier cloud, allocation pie, risk contributions, correlation network, weights heatmap.
Derivatives	Payoff diagrams, BSM/Bachelier comparisons, option value curves, Greek curves, implied-volatility surfaces, Greek surfaces, Monte Carlo convergence, and yield curves.
Machine learning	Feature importance, prediction paths, residuals, ROC, confusion matrix, calibration, regime probabilities.
Microstructure	Bid–ask spread, volume profile, order-book depth, slippage curve, trade timeline.
General 2D/3D	Generic response surfaces, parameter heatmaps, contour maps, interactive Plotly surfaces, multi-parameter frame sliders, stable-scale animations, and GIF/MP4 export.

The goal is not to assert that every plot should be used in every paper. The goal is to make standard diagnostics easy enough that researchers are less tempted to omit them due to implementation cost.

8 Performance and Risk Metrics

For return series $R_1, \dots, R_T$ and annualization factor A , the annualized arithmetic volatility is

$$\hat{\sigma}_A = \sqrt{A} \hat{\sigma}(R). \quad (13)$$

The compound annual growth rate is

$$\text{CAGR} = \left(\prod_{t=1}^T (1 + R_t) \right)^{A/T} - 1. \quad (14)$$

The annualized Sharpe ratio is

$$\widehat{SR} = \sqrt{A} \frac{\bar{R} - r_f/A}{\widehat{\sigma}(R - r_f/A)}. \quad (15)$$

The package reports `NaN` when volatility is numerically zero rather than returning an infinite Sharpe ratio. The Sortino ratio replaces total volatility with root-mean-square downside deviation. Maximum drawdown is computed from the running wealth maximum:

$$W_t = \prod_{s=1}^t (1 + R_s), \quad (16)$$

$$D_t = \frac{W_t}{\max_{u \leq t} W_u} - 1, \quad (17)$$

$$\text{MDD} = \min_t D_t. \quad (18)$$

Historical Value at Risk and Expected Shortfall at confidence level α are

$$\text{VaR}_\alpha = -Q_{1-\alpha}(R), \quad (19)$$

$$\text{ES}_\alpha = -\mathbb{E}[R \mid R \leq Q_{1-\alpha}(R)]. \quad (20)$$

The metrics table also includes Calmar, Omega, ulcer index, tail ratio, hit rate, profit factor, skewness, excess kurtosis, probabilistic Sharpe ratio, average turnover, annual turnover, and optional benchmark alpha, beta, tracking error, and information ratio.

9 Regression and Statistical Diagnostics

9.1 OLS and robust covariance

For dependent variable y and regressor matrix X , `ASRQuant` estimates

$$y = X\beta + \varepsilon \quad (21)$$

by ordinary least squares. The default covariance estimator is heteroskedasticity-and-autocorrelation consistent (HAC/Newey–West style), with an automatically selected lag proportional to $T^{1/4}$ unless the user specifies it. HC0–HC3 and classical covariance are also available. Returned diagnostics include R^2 , adjusted R^2 , Durbin–Watson, Jarque–Bera, Ljung–Box, Breusch–Pagan, and White-test p-values.

9.2 Rolling regression

Rolling OLS exposes time-varying factor loadings. For window h , coefficients at time t solve

$$\widehat{\beta}_t = \arg \min_{\beta} \sum_{s=t-h+1}^t (y_s - X_s \beta)^2. \quad (22)$$

The visualization layer displays coefficient paths and factor-exposure heatmaps.

9.3 Stationarity tests

The Augmented Dickey–Fuller and KPSS tests are reported together because their null hypotheses differ. The software does not convert the pair into an automatic stationarity verdict; interpretation remains the user’s responsibility.

9.4 Dependent-data bootstrap

A moving-block bootstrap resamples contiguous blocks, preserving local dependence better than an i.i.d. bootstrap. The default block length is proportional to $T^{1/3}$. The function returns the point estimate, percentile confidence bounds, and bootstrap standard error.

9.5 Multiple testing

Benjamini–Hochberg adjusted p-values are provided for false-discovery-rate control. The approximate Deflated Sharpe Ratio uses the expected maximum Sharpe under a specified number of trials and then evaluates a probabilistic Sharpe statistic. This is a research diagnostic; a complete trial ledger remains necessary for credible interpretation.

9.6 Extended regression and time-series models

Quantile regression estimates a conditional quantile $Q_\tau(y | X) = X\beta_\tau$ and is useful when predictor relationships differ across the return distribution. Polynomial transformations allow controlled nonlinear terms. Ridge, Lasso, and Elastic Net are implemented through standardized scikit-learn pipelines. Binary logistic regression returns probabilities, robust coefficient intervals, pseudo- R^2 , AIC, and BIC. Factor regression applies the same HAC infrastructure to excess returns. Engle–Granger cointegration, Granger-predictive tests, ARIMA, and vector autoregression are exposed as explicit models; the term “Granger causality” is interpreted only as incremental predictability and not as structural causation.

9.7 Leakage-aware machine learning

For price P_t and horizon h , the default regression target is

$$y_t = \frac{P_{t+h}}{P_t} - 1, \quad (23)$$

with the last h observations left undefined. Feature constructors generate only explicit lags and rolling statistics; they never backward-fill future information. Each walk-forward fold fits a fresh estimator clone on chronological training data, optionally inserts a gap, and evaluates regression with RMSE, MAE, and R^2 or classification with accuracy, ROC AUC, and log loss. This design cannot detect every semantic leak, but it prevents random temporal shuffling and accidental estimator reuse across folds.

10 Time-Aware Validation and Leakage Controls

Random train/test splitting is generally inappropriate for autocorrelated financial time series. ASRQuant therefore provides ordered walk-forward splits with fixed or expanding training windows. If I_k^{train} and I_k^{test} denote the k th split, the invariant is

$$\max I_k^{\text{train}} < \min I_k^{\text{test}}. \quad (24)$$

Purged K-fold splitting removes observations near the test interval and applies an optional embargo after the interval. This reduces overlap when labels or features span multiple periods. A heuristic leakage report also compares signal correlations with same-bar and next-bar returns

and checks index ordering. These checks cannot prove absence of leakage. Point-in-time source validation and feature-specific availability timestamps remain essential.

11 Data Ingestion and Near-Real-Time Feeds

ASRQuant accepts a pandas Series or DataFrame directly and provides constructors for CSV, Parquet, Excel, JSON, Feather, and SQL. The local-file contract is:

```
import asrquant as asr
lab = asr.QuantLab.from_csv(
    "prices.csv", date_column="Date",
    columns=["SPY", "TLT", "GLD"]
)
print(lab.quality)
```

Provider adapters expose one method, `history(symbol, **kwargs)`, and normalize provider-specific responses into time-indexed pandas tables. Version 0.5.0 includes Alpha Vantage equities/intraday data, public Binance Spot candles, FRED macroeconomic series, and optional Yahoo downloads. Multiple symbols are aligned by timestamp and field. A `PollingFeed` yields the latest observation at an explicit interval for notebooks and dashboards. It is intentionally not a WebSocket order-management or brokerage component. Provider availability, entitlements, redistribution rights, rate limits, timestamp semantics, and real-time status remain properties of the upstream service and must be recorded by the researcher.

12 Scientific Literature and Hypothesis Provenance

Let a supplied corpus be $\mathcal{C} = \{P_1, \dots, P_J\}$, where every paper P_j is represented as a sequence of extracted pages $P_j = (p_{j1}, \dots, p_{jM_j})$. A source excerpt is the tuple

$$e = (j, m, s),$$

where j identifies the paper, m identifies the page, and s is the extracted sentence. A candidate hypothesis H_k stores a statement, an expected direction when recoverable, a confidence score, and a nonempty evidence set E_k of such tuples. The page coordinate is preserved throughout reporting so that a researcher can inspect the original context rather than accepting a detached generated summary.

The default extractor is deterministic and conservative. It identifies hypothesis-like language, research questions, directional claims, and future-work or limitation statements. Similar candidates are grouped with transparent token overlap. A custom extractor may connect a language model, but it must return the same page-linked evidence contract. The framework does not treat generated output as evidence unless the source passage is stored.

A separate evidence-status field distinguishes passages that appear to report an empirical test, passages that explicitly state an untested gap, and passages that merely propose a relationship. These labels are triage metadata rather than final scientific judgments; the cited page remains the authoritative object for human review.

Novelty labels are defined only relative to $\mathcal{C}$. If n_k is the number of distinct papers supporting candidate H_k , the current heuristic distinguishes established claims, replicated claims, under-explored claims, contradictory directional language, and corpus gaps. The label “corpus-novel” means

$$\nexists H_\ell \in \mathcal{C} \setminus \{P_j\} \text{ directly matched by the supplied retrieval procedure,}$$

not that no related paper exists in the global literature. This distinction is essential because a finite local corpus cannot prove a universal negative about prior research.

Scanned PDFs with no extractable text are flagged rather than silently processed. Optical character recognition may change symbols, signs, equations, and variable names; therefore OCR must be performed and reviewed outside the default ingestion path before a scientific claim is extracted.

13 Hypothesis-to-Decision Research Workflow

An operational economic hypothesis is represented as

$$H = (q, x, y, d, h, \mathcal{U}, H_0, \mathcal{M}, \mathcal{F}),$$

where q is the verbal claim, x is the predictor, y is the target, d is the expected sign, h is the horizon, $\mathcal{U}$ is the universe, H_0 is the null, $\mathcal{M}$ is the economic mechanism, and $\mathcal{F}$ is a set of falsification criteria. The object can be constructed from a paper-linked candidate or entered directly by the researcher.

A data plan maps the constructs in H to proposed variables, sources, symbols, frequencies, fields, availability lags, and point-in-time requirements. Provider mappings are suggestions rather than automatic construct validation. For a variable observed or published with lag ℓ_j , a feature transformation g_j is constrained to

$$X_{j,t} = g_j(D_{j,1:t-\ell_j}),$$

so that the feature cannot use information unavailable at the decision timestamp. Feature specifications include raw values, differences, returns, momentum, rolling moments, z-scores, exponential averages, ranks, volatility, drawdowns, ratios, spreads, and interactions.

A signal map ϕ transforms the feature vector into target weights,

$$w_t^* = \phi(X_t; \theta),$$

where θ contains thresholds, direction, long and short assets, gross exposure, and an additional signal lag. A portfolio layer then enforces position, leverage, long-only, and optional volatility-targeting constraints before the existing backtest execution delay is applied.

Before portfolio evaluation, the framework can test a predictive implication of H . For a two-asset relative-value hypothesis, the future target is

$$y_{t,h} = R_{t,t+h}^L - R_{t,t+h}^S,$$

and a HAC regression estimates

$$y_{t,h} = \alpha + \beta X_t + \varepsilon_{t,h}.$$

The result records whether $\text{sign}(\hat{\beta})$ agrees with d , but this predictive regression is not interpreted as proof of structural causality.

Robustness combines chronological subperiods, moving-block bootstrap inference, alignment diagnostics, and an implementation grid over delays, costs, and rebalance rules. A governed decision score aggregates positive net performance, Sharpe quality, probabilistic Sharpe, subperiod stability, implementation resilience, drawdown control, chronology, and hypothesis-sign consistency. The output status is one of reject, research-only, collect-more-data, revise-hypothesis, paper-trading candidate, or limited-capital candidate. An unrestricted live-deployment status is deliberately absent because a backtest alone cannot authorize production trading.

14 Algorithmic Trading and Paper Execution

The execution layer separates a strategy target from an order. At time t , with marked portfolio equity V_t , current position $q_{i,t}$, price $S_{i,t}$, and constrained target weight $w_{i,t}^*$, the desired notional change is

$$\Delta N_{i,t} = w_{i,t}^* V_t - q_{i,t} S_{i,t}.$$

The paper trader converts $\Delta N_{i,t}$ into buy or sell orders after applying maximum position weight, gross leverage, order-notional, turnover, cash, and short-selling constraints. Sells are processed before buys so that cash released by reductions is available to finance subsequent purchases. Commission and slippage are explicit basis-point functions of filled notional.

The reference paper broker supports market, limit, stop, and stop-limit order primitives; created, accepted, partially-filled, filled, rejected, and cancelled states; deterministic position and cash accounting; and a configurable participation rate. A drawdown kill switch sets desired exposure to zero when

$$1 - \frac{V_t}{\max_{u \leq t} V_u} \geq d_{\max}.$$

Every order, fill, realized weight, cash balance, equity value, and risk event is returned as a tabular object. The package exposes a minimal broker protocol for external adapters, but it does not include credentials, venue authentication, asynchronous broker callbacks, or live routing by default.

15 Stochastic Processes and Monte Carlo

15.1 Arithmetic and geometric Brownian motion

Arithmetic Brownian motion is

$$X_t = X_0 + \mu t + \sigma W_t, \quad (25)$$

while geometric Brownian motion is

$$S_t = S_0 \exp \left[\left(\mu - \frac{1}{2} \sigma^2 \right) t + \sigma W_t \right]. \quad (26)$$

The GBM implementation uses the exact transition, supports antithetic Gaussian shocks, and returns the full path matrix with parameters and terminal summaries.

15.2 Mean reversion, stochastic volatility, and jumps

The Ornstein–Uhlenbeck model is simulated as

$$dX_t = \kappa(\theta - X_t)dt + \sigma dW_t. \quad (27)$$

The Vasicek short-rate process uses its exact Gaussian transition. CIR employs full-truncation Euler,

$$dr_t = \kappa(\theta - r_t)dt + \sigma \sqrt{r_t} dW_t, \quad (28)$$

to avoid negative numerical variance states. The Heston model jointly evolves price and variance,

$$dS_t = \mu S_t dt + \sqrt{v_t} S_t dW_t^S, \quad (29)$$

$$dv_t = \kappa(\theta - v_t)dt + \xi \sqrt{v_t} dW_t^v, \quad (30)$$

$$d\langle W^S, W^v \rangle_t = \rho dt, \quad (31)$$

using full truncation for v_t . Merton jump diffusion augments log returns with a compound Poisson normal jump term. These discretizations are research baselines and do not eliminate bias for difficult parameter regimes.

15.3 Monte Carlo estimators and uncertainty

For discounted payoff $Y = e^{-rT} g(S_T)$ and N simulated paths,

$$\hat{V}_N = \frac{1}{N} \sum_{i=1}^N Y_i, \quad (32)$$

$$\widehat{\text{SE}}(\hat{V}_N) = \frac{s_Y}{\sqrt{N}}, \quad (33)$$

$$\text{CI}_{1-\alpha} = \hat{V}_N \pm z_{1-\alpha/2} \widehat{\text{SE}}(\hat{V}_N). \quad (34)$$

The framework returns the estimate, standard error, confidence interval, discounted path payoffs, and underlying simulation. Terminal-payoff claims use a generic payoff callback. Convenience functions price European and arithmetic-average Asian calls or puts under risk-neutral GBM.

16 Martingale Diagnostics

A process (M_t) is a martingale with respect to filtration $(\mathcal{F}_t)$ when it is integrable and

$$\mathbb{E}[M_t \mid \mathcal{F}_s] = M_s, \quad s \leq t. \quad (35)$$

Under a risk-neutral measure, a correctly discounted tradable is expected to satisfy this property under appropriate assumptions. The software constructs $\tilde{S}_t = e^{-rt}S_t$ and examines increments $\Delta\tilde{S}_t$. The diagnostic suite reports: a one-sample test of zero mean increments; a HAC regression

$$\Delta\tilde{S}_t = \alpha + \beta\tilde{S}_{t-1} + \varepsilon_t; \quad (36)$$

and a Ljung–Box test for increment autocorrelation. Failure to reject these finite-dimensional restrictions is not proof of the conditional-expectation property. Conversely, rejection can indicate drift, dependence, misspecified discounting, structural breaks, or discretization effects.

17 Derivative Pricing

17.1 Black–Scholes–Merton and Black–76

For spot S , strike K , maturity T , rate r , dividend yield q , and lognormal volatility σ , the BSM European call is

$$C = Se^{-qT}\Phi(d_1) - Ke^{-rT}\Phi(d_2), \quad (37)$$

$$d_1 = \frac{\log(S/K) + (r - q + \sigma^2/2)T}{\sigma\sqrt{T}}, \quad d_2 = d_1 - \sigma\sqrt{T}. \quad (38)$$

Black–76 replaces spot carry with a discounted forward price. Analytic BSM delta, gamma, vega, theta, and rho are returned in vectorized form.

17.2 Bachelier normal pricing

For forward F , normal volatility σ_N , discount factor D , and

$$d = \frac{F - K}{\sigma_N\sqrt{T}}, \quad (39)$$

the Bachelier call is

$$C_N = D \left[(F - K)\Phi(d) + \sigma_N\sqrt{T}\phi(d) \right]. \quad (40)$$

The implementation includes calls, puts, forward delta, gamma, vega, and calendar theta, and permits negative forwards even though the current validation requires positive maturities, discount factors, and normal volatility.

17.3 Trees, implied volatility, and Monte Carlo

The Cox–Ross–Rubinstein tree uses $u = e^{\sigma\sqrt{\Delta t}}$, $d = u^{-1}$, and

$$p = \frac{e^{(r-q)\Delta t} - d}{u - d} \quad (41)$$

to price European or American calls and puts by backward induction. Implied volatility is inverted by Brent bracketing for BSM, Black–76, or Bachelier. A unified `price_option` dispatcher standardizes results from closed form, tree, and Monte Carlo models.

18 Portfolio, Volatility, Risk, and Fixed Income

Given expected returns μ and covariance Σ , the allocation layer supports minimum variance, maximum Sharpe, equal risk contribution, maximum diversification, Monte Carlo and constrained efficient frontiers, hierarchical risk parity, and Black–Litterman posterior moments. Covariance estimators include sample, exponentially weighted, Ledoit–Wolf, and Oracle Approximating Shrinkage. These models remain sensitive to estimation error; the library does not represent optimized weights as uniquely correct decisions.

Volatility utilities include rolling close-to-close, Parkinson, Garman–Klass, EWMA, and optional GARCH(p, q) forecasts. Risk metrics include historical and parametric VaR, Cornish–Fisher VaR, Expected Shortfall, CDaR, drawdown duration, Omega, ulcer index, probabilistic and deflated Sharpe ratios, capture ratios, and stress scenarios.

The fixed-income layer prices zero-coupon and fixed-coupon bonds, solves yield to maturity, calculates Macaulay and modified duration and convexity, and bootstraps simple zero curves from par instruments. It is a deterministic curve baseline; calendars, day-count conventions, settlement, accrued interest, credit, callable structures, and multi-curve collateral frameworks are outside the current implementation.

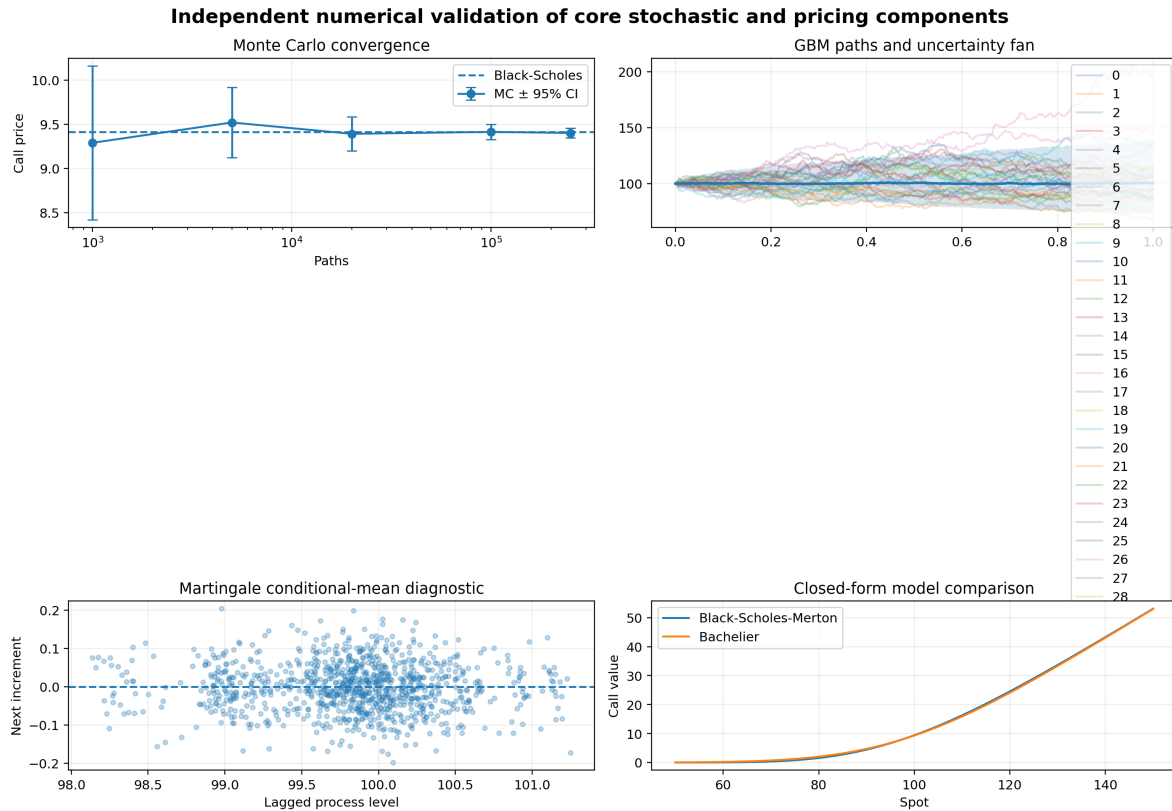


Figure 2: Numerical validation panels: Monte Carlo convergence to Black–Scholes–Merton, GBM path uncertainty, a finite-sample martingale diagnostic, and a Black–Scholes/Bachelier model comparison.

19 Reproducibility and Provenance

A result object stores the validated prices, asset returns, target weights, effective weights, gross and net returns, equity, turnover, total cost, cost decomposition, specification, and metadata. A manifest records:

- UTC creation timestamp;
- Python, operating-system, NumPy, and pandas versions;
- data fingerprint;

- specification fingerprint;
- combined experiment fingerprint;
- user-supplied metadata such as dataset version, source, or research issue.

The self-contained HTML report embeds figures as base64 data. This avoids broken relative image links when a report is moved or archived. The source repository also includes `CITATION.cff`, an MIT license, a contributor guide, a code of conduct, continuous-integration workflow, and benchmark scripts.

20 Empirical Evaluation

20.1 Objectives

The evaluation does not attempt to demonstrate a profitable trading strategy. It assesses four software properties:

1. deterministic end-to-end execution;
2. visibility of cost and timing sensitivity;
3. ability to produce research diagnostics and figures;
4. practical runtime scaling for vectorized experiments.

20.2 Synthetic market

We use a two-state regime-switching generator with Markov transition matrix

$$\Pi = \begin{pmatrix} 0.97 & 0.03 \\ 0.08 & 0.92 \end{pmatrix}. \quad (42)$$

State 0 has annualized drift 8% and volatility 12%; state 1 has annualized drift -12% and volatility 32%. Asset shocks have pairwise correlation 0.35. Prices are generated by exponentiating cumulative log returns. Random seeds are fixed in the benchmark script.

20.3 Reference strategy

The logical strategy is a moving-average crossover. For fast window f and slow window $s > f$,

$$z_{t,i}^{\text{raw}} = \mathbf{1} \{ \text{SMA}_f(P_{:,i})_t > \text{SMA}_s(P_{:,i})_t \}. \quad (43)$$

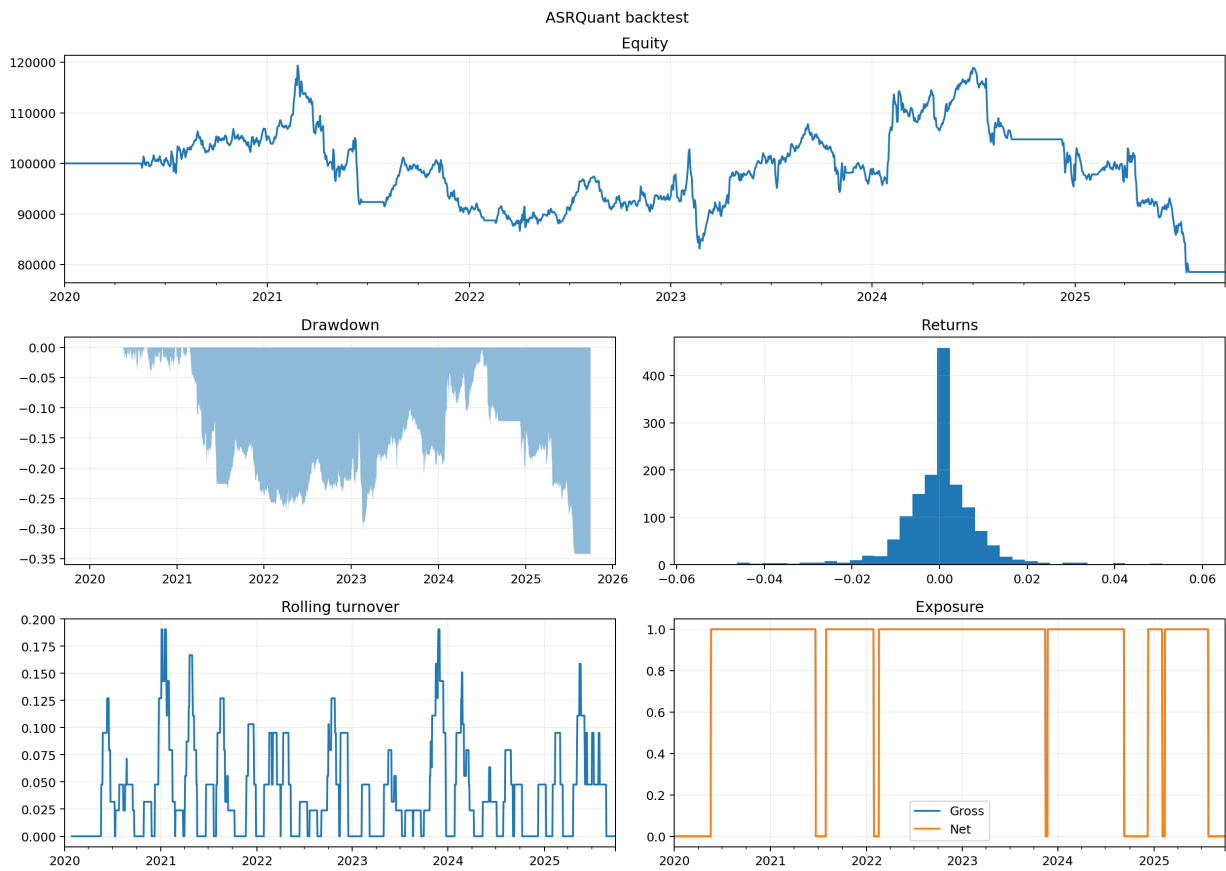
Active long signals are normalized cross-sectionally to unit gross exposure. Unless otherwise stated, $f = 20$, $s = 100$, execution delay is one bar, and linear transaction cost is 5 basis points per unit of turnover.

20.4 Reference result

The reference experiment contains 1,500 business-day observations and four assets. It intentionally produces a negative result, avoiding the implication that the software example has been selected for profitability.

Table 3: Selected reference metrics.

Metric	Value
Total return	-21.4705%
CAGR	-3.9792%
Annual volatility	14.9722%
Sharpe ratio	-0.1963
Sortino ratio	-0.2710
Maximum drawdown	-34.2904%
Expected Shortfall 95%	2.4404%
Average turnover per bar	3.9778%
Annualized turnover	10.0240

**Figure 3:** Reference dashboard combining equity, drawdown, return distribution, turnover, and exposure.

20.5 Implementation audit: stable negative conclusion

The first audit combines three delays $\{0, 1, 2\}$, four linear cost levels $\{0, 5, 10, 25\}$ basis points, and two rebalancing conventions (every bar and month-end), for 24 contracts. Total return ranges from -30.34% to -9.16%. The return sensitivity is 21.18 percentage points, but every contract remains negative, so the sign-based conclusion stability index equals one. This example distinguishes metric instability from conclusion instability.

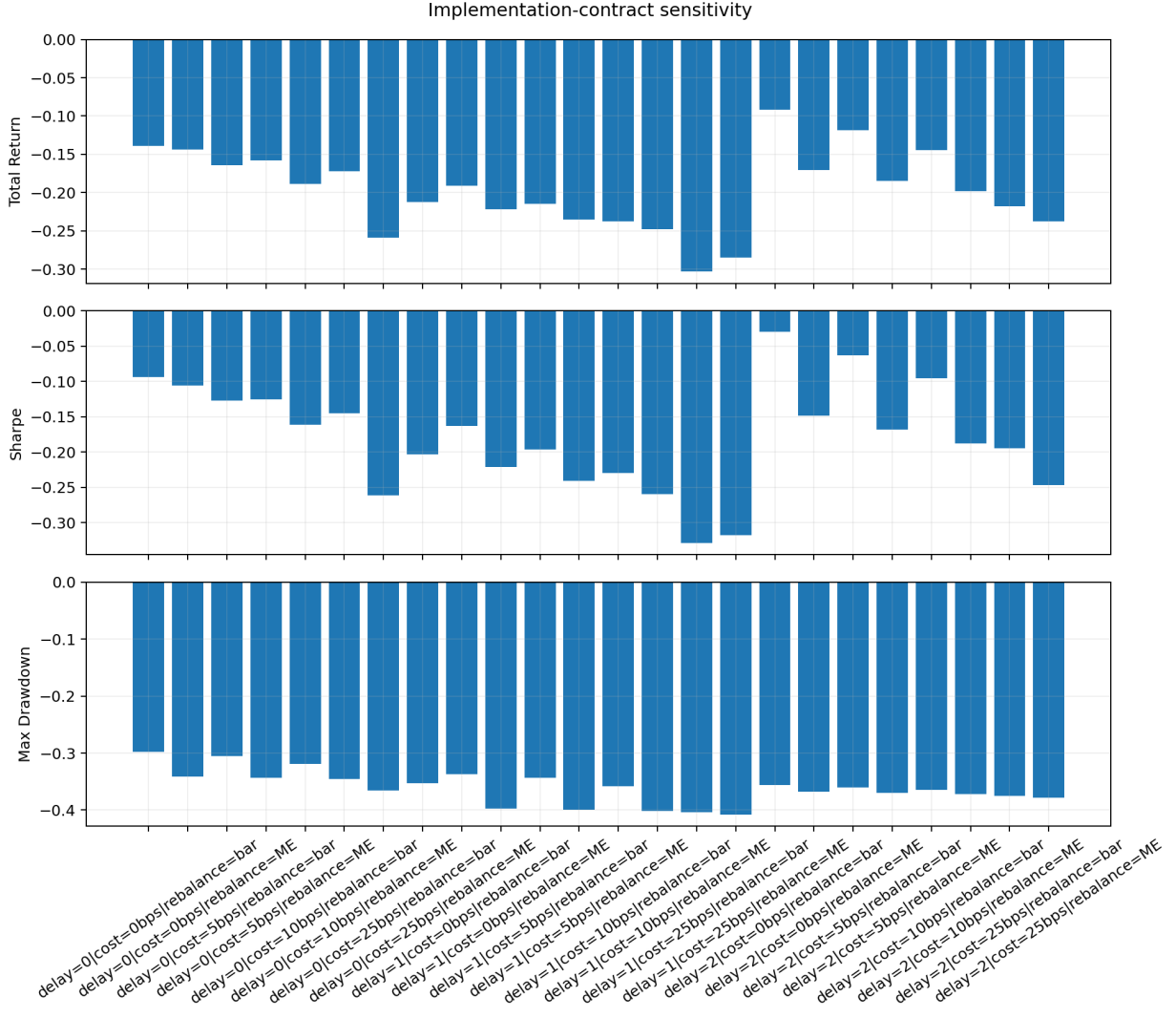


Figure 4: Selected metrics under 24 implementation contracts. The sign is stable, but magnitude is not.

20.6 Implementation audit: fragile conclusion

A shorter 500-bar, three-asset experiment with fast window 10 and slow window 40 illustrates a fragile edge. Table 4 reports all six contracts.

Table 4: Fragile strategy under execution and cost alternatives.

Delay	Cost (bps)	Total return	Sharpe	Max drawdown
0	0	12.5975%	0.4777	-10.6600%
0	5	10.1847%	0.4040	-10.8532%
0	10	7.8224%	0.3301	-11.2408%
1	0	1.9718%	0.1405	-13.3180%
1	5	-0.2094%	0.0684	-13.6062%
1	10	-2.3450%	-0.0039	-13.8936%

The implementation uncertainty interval for total return is $[-2.3450\%, 12.5975\%]$, the return sensitivity is 14.94 percentage points, and the sign-based conclusion stability index is $1/3$. A researcher reporting only the same-bar, zero-cost result would communicate a materially different conclusion from a researcher using the package default plus modest costs.

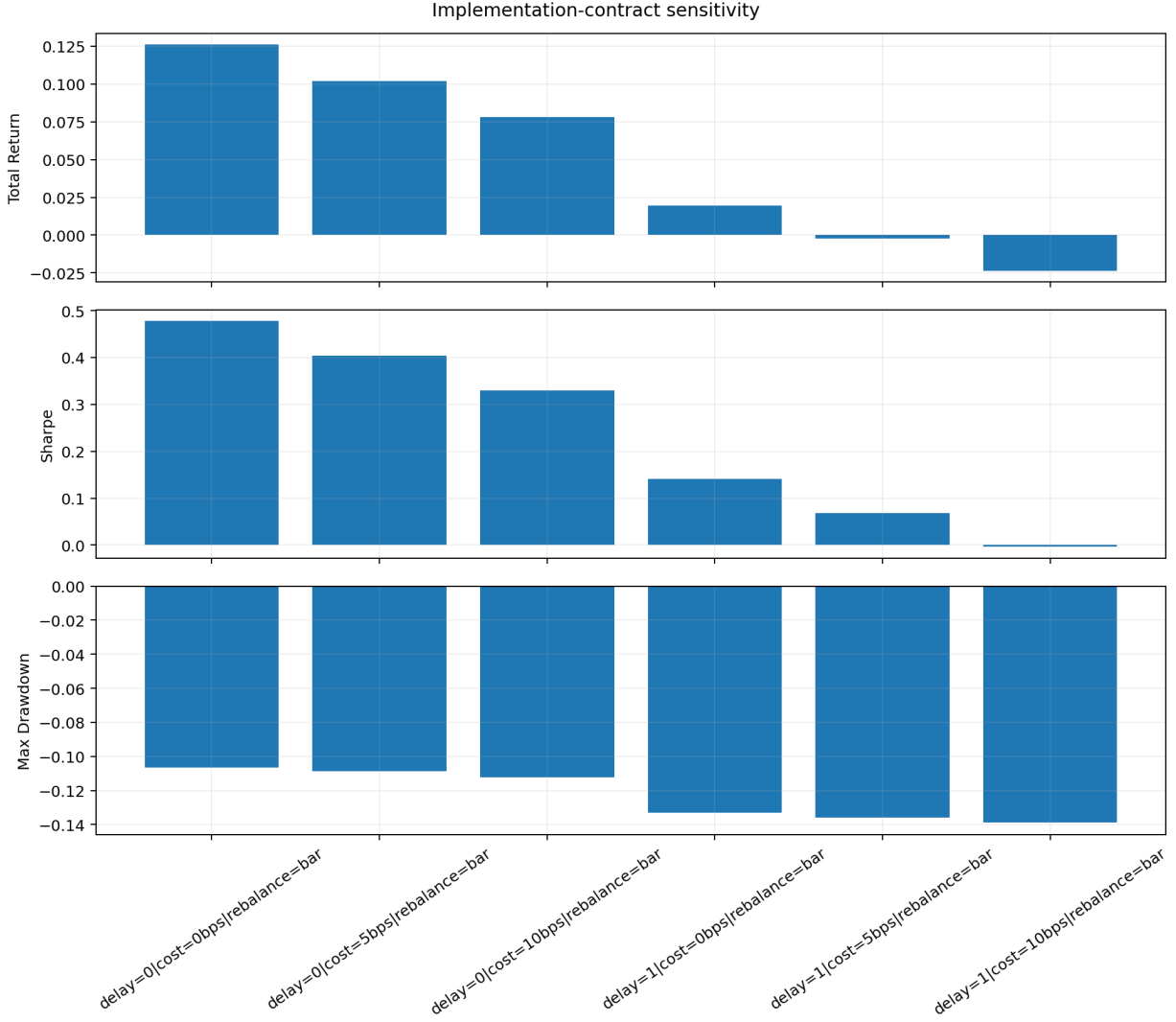


Figure 5: Fragile strategy audit. The sign changes when a one-bar delay and realistic costs are introduced.

20.7 Parameter landscapes and multidimensional animation

A geometric response surface displays two parameters. Let $g(\theta)$ denote any experiment returning an object and let m be a scalar metric selector. For displayed axes x_i and y_j , fixed parameters c , and an animation-frame configuration ϕ_k , ASRQuant evaluates

$$Z_{kij} = m(g(x_i, y_j, \phi_k, c)). \quad (44)$$

The frame configuration may itself contain several parameters. If $\Phi_1, \dots, \Phi_q$ are the selected animation dimensions, the ordered frame table is the Cartesian product

$$\mathcal{F} = \Phi_1 \times \dots \times \Phi_q. \quad (45)$$

This construction does not pretend that a literal surface can have more than two geometric axes. It represents higher-dimensional experiments as a reproducible sequence of two-dimensional slices. Frame values may be numeric, Boolean, or categorical, allowing examples such as risk aversion and cost on the axes while hedge frequency, volatility regime, model family, and random seed determine the animation frames. The metric selector may be a mapping key, dotted object path such as `metrics.Sharpe`, or an arbitrary reduction callback.

Parameter sweeps can be visualized as heatmaps, contours, three-dimensional surfaces, or interactive HTML animations with a parameter slider. GIF and MP4 exports are also supported,

with stable global color and vertical scales to avoid visual changes caused only by automatic rescaling. Figure 6 shows Sharpe ratios across fast and slow moving-average windows under a fixed execution contract. Figure 7 illustrates four slices from a surface family in which two additional hedging parameters define the frames.

SMA Sharpe parameter surface

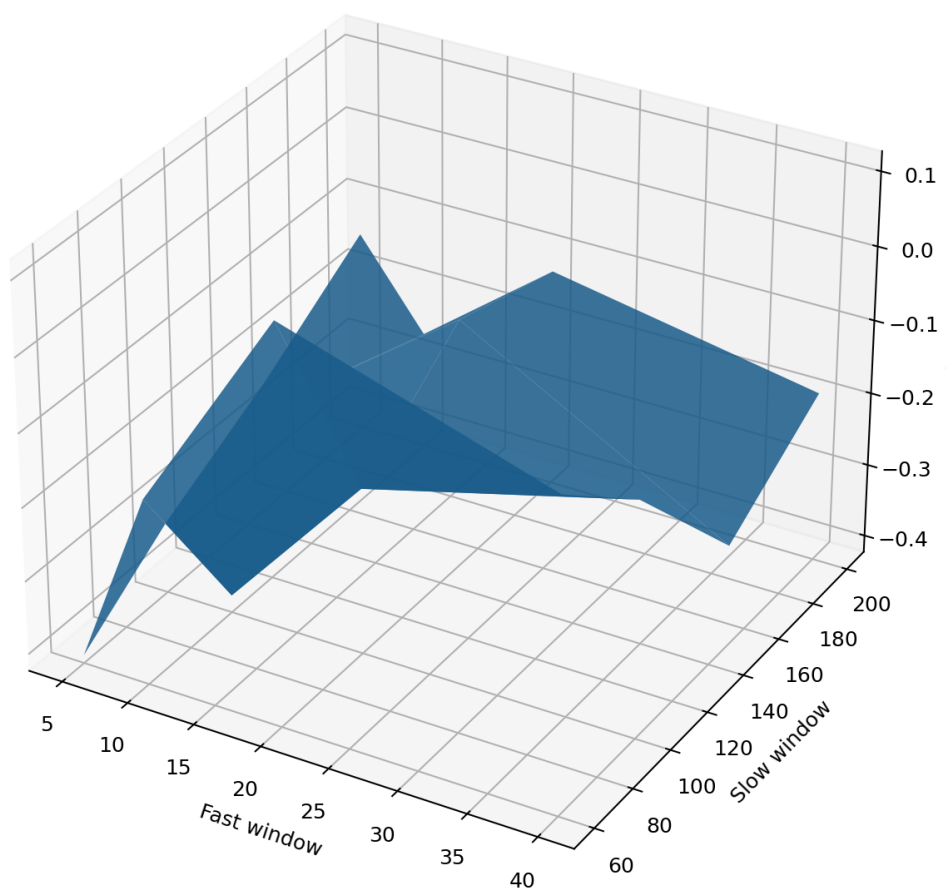


Figure 6: Sharpe-ratio parameter surface for moving-average windows.

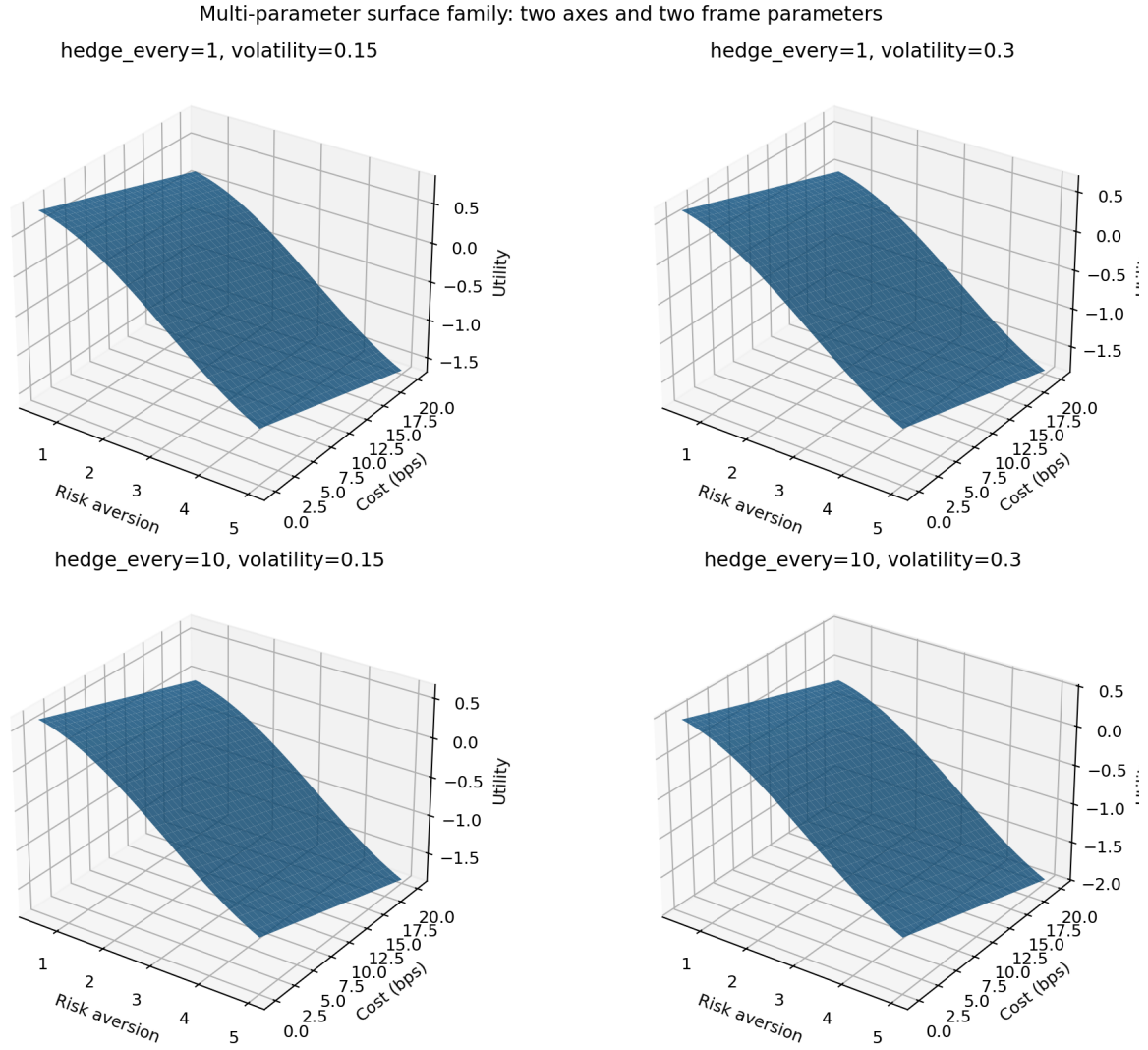


Figure 7: A multidimensional parameter experiment represented by two surface axes and a Cartesian frame table over hedge frequency and volatility.

A surface does not validate a strategy. Evaluating many configurations increases selection risk, and visually smooth regions may reflect interpolation, common random numbers, or model structure rather than genuine economic robustness. The package therefore records evaluation counts, fixed parameters, frame parameters, failures, and best-point extraction, while leaving out-of-sample design and multiple-testing control to the research protocol.

20.8 Runtime scaling

Runtime was measured as the median of five runs after data construction on the benchmark environment. The table reports end-to-end strategy construction and backtesting. Results are environment-specific and should not be interpreted as universal performance guarantees.

Table 5: Vectorized runtime scaling.

Bars	Assets	Data cells	Median seconds
1,000	1	1,000	0.0065
1,000	10	10,000	0.0093
5,000	10	50,000	0.0136
5,000	50	250,000	0.0361
20,000	50	1,000,000	0.1145

The scaling is consistent with array-oriented operations over the price and weight panels. Memory, rather than arithmetic, will become the main limitation for very large grids or high-frequency datasets.

20.9 Independent numerical model validation

For $S_0 = K = 100$, $T = 1$, $r = 3\%$, and $\sigma = 20\%$, the analytic BSM call value is 9.413403. Table 6 compares independent numerical engines.

Table 6: Independent pricing checks against the analytic BSM value.

Method	Resolution	Estimate	Absolute error
Monte Carlo, antithetic	1,000 paths	9.289937	0.123466
Monte Carlo, antithetic	20,000 paths	9.391629	0.021774
Monte Carlo, antithetic	100,000 paths	9.413542	0.000139
CRR binomial	100 steps	9.393596	0.019808
CRR binomial	1,000 steps	9.411420	0.001983

The 100,000-path Monte Carlo 95% interval is [9.326114, 9.500970], containing the analytic value. Error need not decline monotonically for one random sequence, but the standard error scales approximately as $N^{-1/2}$. The tree error decreases as the time grid is refined.

A zero-drift arithmetic Brownian sample with 1,008 increments produced p-values 0.4493 for zero mean, 0.3314 for lagged-level predictability, and 0.7927 for Ljung–Box autocorrelation. These are non-rejections in one controlled sample, not evidence that a market price series is a martingale. In a separate 1,500-observation factor simulation with true coefficients $(\alpha, \beta_M, \beta_V) = (0.0002, 1.25, -0.40)$, HAC estimates were (0.000276, 1.25523, -0.39490) and all true values lay within the reported 95% intervals. These checks test formula wiring, alignment, and numerical convergence rather than economic usefulness.

21 Verification and Testing

The release contains 64 automated tests spanning:

- specification fingerprints and validation;
- missing-data policies and data fingerprints;
- OHLC consistency;
- buy-and-hold return identity;
- cost monotonicity and execution-delay sensitivity;
- leverage and weight constraints;
- five-line API, local-file ingestion, provider-response normalization, CLI commands, and HTML report generation;
- performance, benchmark, drawdown, volatility, and tail-risk metrics;
- implementation-audit grids, trade ledgers, strategy comparisons, and parameter sweeps;

- static, single-frame, and multi-parameter surface evaluation, categorical frame tables, dotted metric extraction, parallel execution, error recovery, Plotly HTML export, and multidimensional backtest animations;
- walk-forward, gap-aware, and purged temporal splits;
- ABM, GBM, Heston, and jump-process simulation invariants;
- Monte Carlo pricing against Black–Scholes–Merton and CRR convergence;
- Bachelier, Black–76, put–call parity, implied-volatility recovery, and option dispatch;
- martingale diagnostics under controlled driftless processes;
- portfolio optimizer constraints, covariance estimators, HRP, Black–Litterman, and risk parity;
- OLS/HAC coefficient recovery, robust and regularized regressions, bootstrap, stationarity, FDR, and walk-forward ML;
- bond, yield, duration, convexity, and volatility-estimator checks;
- smoke tests for market, performance, risk, stochastic, portfolio, regression, derivative, ML, and microstructure figures.

All tests pass in the artifact environment used to produce this paper. Visual tests check successful figure construction rather than pixel-perfect rendering, because minor renderer differences should not fail scientific calculations.

22 Comparison with Existing Workflows

Table 7: Qualitative design comparison. This table describes primary emphasis, not every optional feature of each project.

Project	Backtest	Tear sheet	Broad viz	Contract audit	Primary emphasis
backtesting.py	Yes	Partial	Partial	No	Concise event-oriented strategy testing.
vectorbt	Yes	Yes	Strong	Partial/custom	High-performance vectorized research.
pyfolio	No	Yes	Performance	No	Portfolio performance and risk analytics.
QuantStats	No	Yes	Performance	No	Return-stream profiling and reports.
ASRQuant	Yes	Yes	Cross-domain	Yes	Explicit semantics plus integrated research diagnostics.

The comparison is not intended to rank mature projects against a version 0.5.0 alpha release. ASRQuant can interoperate with them: external engines can supply returns or weights, and future adapters can compare their results under a common contract.

23 Limitations

23.1 Execution realism

The reference engine is weight-based and bar-based. It does not model limit-order queues, partial fills, order priority, exchange fees by venue, latency, intrabar path, market halts, auction mechanics, or stochastic fills. Researchers working at high frequency should use an event-driven simulator and can still use ASRQuant for analytics and visualization.

23.2 Corporate actions and point-in-time data

The package validates numerical panels but does not supply a corporate-action database or point-in-time fundamental data. Split adjustment, dividends, delistings, survivorship, restatements, and publication lags must be handled upstream and documented in the manifest.

23.3 Transaction-cost calibration

Basis-point costs and power-law impact are simplified models. Calibrated execution costs should depend on spread, volatility, volume, participation, order type, and market state. A visually attractive cost plot cannot compensate for weak calibration.

23.4 Statistical inference

The included DSR is an approximation and depends on the declared number of trials. The framework cannot infer unrecorded experiments. Block-bootstrap validity depends on stationarity and block-length choices. Regression p-values are not causal evidence.

23.5 Visualization risk

Plots can encourage narrative overfitting. Parameter surfaces, animations, regime colors, and three-dimensional figures are descriptive tools. A large Cartesian grid increases selection opportunities and computational cost. Surfaces should be accompanied by pre-specified hypotheses, genuine out-of-sample evaluation, multiple-testing control where appropriate, and uncertainty analysis. Stable frame scales prevent one visual artifact but do not establish robustness.

23.6 Optimization instability

Mean–variance solutions can be highly sensitive to estimated means and covariances. The package includes shrinkage estimators, HRP, risk parity, maximum diversification, and Black–Litterman, but these methods do not remove estimation risk. Distributionally robust optimization, explicit posterior-uncertainty propagation, taxes, integer lots, and transaction-aware multi-period optimization remain outside the current release.

23.7 Package maturity

Version 0.5.0 is an alpha research release. The public API may evolve before version 1.0. PyPI name availability is not guaranteed until the distribution is actually registered and uploaded.

24 Ethical and Practical Use

The package is not financial advice and does not predict future returns. A backtest may be technically correct and economically irrelevant. Researchers should disclose data sources, availability timestamps, costs, rejected specifications, and all material implementation decisions. Users should avoid presenting synthetic or in-sample results as evidence of deployable profitability. Sensitive or proprietary data should not be embedded in public HTML reports or manifests.

25 Roadmap

The implemented scope is intentionally broad, but not equivalent to a production trading platform or a complete replacement for specialist libraries. Priority extensions are:

1. an event-driven engine with orders, partial fills, limit logic, corporate actions, borrow recalls, and venue calendars;

2. authenticated WebSocket feeds, persistent storage, and broker adapters separated from the research package by strict safety boundaries;
3. point-in-time fundamentals, vintage macro data, security-master identifiers, delistings, and survivorship controls;
4. local-volatility, SABR, Hull–White, affine term-structure, credit, exotic, and adjoint-differentiation engines;
5. calibrated volume, spread, square-root impact, capacity, and queue-position models;
6. nested and combinatorial purged validation, Probability of Backtest Overfitting, White Reality Check, SPA, and complete experiment ledgers;
7. robust and distributionally robust optimization, turnover constraints, taxes, integer lots, and multi-period allocation;
8. distributed, JIT, GPU, adaptive sampling, surrogate modelling, and out-of-core execution for very large parameter grids;
9. typed plugin protocols, stable serialized schemas, visual regression tests, and a versioned documentation site.

26 Conclusion

ASRQuant demonstrates that ease of use and scientific rigor need not be opposites. A small public API can unify local and remote data, stochastic simulation, martingale diagnostics, closed-form and numerical pricing, econometrics, machine learning, portfolio construction, multidimensional parameter exploration, visualization, and backtesting, provided that the underlying assumptions remain explicit. The central design choice is therefore not maximal automation; it is auditable automation. Simulations retain seeds and parameters, Monte Carlo prices retain uncertainty, regressions retain robust diagnostics, ML retains chronological folds, and backtests retain the execution contract, costs, trade ledger, and implementation sensitivity.

The numerical checks confirm internal consistency across independent engines, but they do not validate any trading hypothesis. Likewise, access to intraday or polled data does not make the package a production execution system. The appropriate use of ASRQuant is to shorten reproducible quantitative research while making it harder to forget the choices that determine the result.

A Five-Line Workflows

A.1 CSV to audited backtest

```
import asrquant as asr
lab = asr.QuantLab.from_csv("prices.csv", date_column="Date")
result = lab.backtest("momentum", lookback=126, costs_bps=8)
print(result.metrics)
result.report("momentum_report.html")
```

A.2 Remote historical data

```
import asrquant as asr
lab = asr.QuantLab.from_provider(
    "binance", ["BTCUSDT", "ETHUSDT"],
    interval="1d", limit=1000
)
result = lab.backtest("sma", fast=20, slow=100)
result.plot()
```

A.3 Monte Carlo and martingale diagnostics

```
sim = lab.monte_carlo(  
    "heston", drift=0.03, paths=10000,  
    initial_variance=0.04, long_variance=0.04  
)  
sim.plot("fan")  
martingale = lab.martingale_test(rate=0.03)  
print(martingale.statistics)
```

A.4 Unified option pricing

```
price = lab.option(  
    "black_scholes", strike=100, maturity=1,  
    rate=0.03, volatility=0.20  
)  
mc = lab.option(  
    "monte_carlo", strike=100, maturity=1,  
    rate=0.03, volatility=0.20, paths=100000  
)  
print(price.summary, mc.summary)
```

A.5 In-memory custom strategy

```
import asrquant as asr  
lab = asr.QuantLab(prices)  
result = lab.backtest("momentum", lookback=126, costs_bps=8)  
print(result.metrics)  
result.report("momentum_report.html")
```

A.6 Implementation audit

```
audit = lab.audit(  
    "sma", fast=20, slow=100,  
    execution_delays=(0, 1, 2),  
    linear_costs_bps=(0, 5, 10, 25)  
)  
print(audit.summary)  
audit.plot()
```

A.7 Regression diagnostics

```
from asrquant.statistics import ols  
from asrquant.viz.regression import residual_diagnostics  
fit = ols(strategy_returns, factors, covariance="HAC")  
print(fit.coefficients)  
residual_diagnostics(fit)
```

A.8 Option surface

```
from asrquant.viz.derivatives import greek_surface
fig = greek_surface(
    strikes, maturities,
    spot=100, rate=0.03, volatility=0.20,
    greek="gamma", interactive=True
)
fig.show()
```

A.9 Scientific papers to hypothesis registry

```
import asrquant as asr
project = asr.research.from_pdfs(
    "papers/",
    topic="interest rates and equity style returns",
)
registry = project.discover_hypotheses()
print(registry.to_frame())
```

A.10 Hypothesis to governed decision

```
project = asr.research.from_hypothesis(
    "Rising yields predict value outperformance over growth.",
    predictor="US10Y",
    expected_sign="positive",
    horizon=20,
)
project.attach_data(data, tradable_assets=["VALUE", "GROWTH"])
project.build_features("recommended")
project.build_signal()
project.test_hypothesis()
project.construct_portfolio()
project.backtest(costs_bps=5, execution_delay=1)
project.robustness()
print(project.decide().summary)
```

A.11 Paper trading with risk controls

```
paper = project.paper_trade(
    commission_bps=1,
    slippage_bps=2,
    policy=asr.RiskPolicy(
        max_gross_leverage=1.0,
        max_position_weight=0.25,
        max_daily_turnover=0.5,
        max_drawdown=0.15,
    ),
)
print(paper.orders)
print(paper.risk_events)
```

B Public API Catalog

The primary namespaces are `asr.research` for literature and workflow objects and `asr.trading` for orders, paper brokers, risk policies, and execution records. The top-level facade also exports `LiteratureCorpus`, `EconomicHypothesis`, `FeatureSpec`, `SignalSpec`, `PortfolioSpec`, `ResearchProject`, `RiskPolicy`, and `paper_trade`.

Namespace	Main public objects
Top level	<code>QuantLab</code> , typed backtest, simulation, pricing, martingale, ML, provider, and volatility result objects.
Literature/research	<code>LiteratureCorpus</code> , <code>PaperDocument</code> , <code>HypothesisRegistry</code> , <code>EconomicHypothesis</code> , data and feature plans, <code>SignalSpec</code> , <code>PortfolioSpec</code> , <code>ResearchProject</code> , robustness, decisions, manifests, and reports.
Trading	<code>Order</code> , <code>Fill</code> , order enums, <code>RiskPolicy</code> , <code>PaperBroker</code> , <code>PaperTrader</code> , <code>PaperTradingResult</code> , and the broker-adapter protocol.
Data/providers	Local file and SQL loaders, OHLCV validation/resampling, quality reports, fingerprints, Alpha Vantage, Binance, FRED, Yahoo, downloads, polling feeds.
Stochastic/martingale	ABM, GBM, correlated GBM, OU, CIR, Vasicek, Heston, Merton, stationary bootstrap, Monte Carlo pricing, discounting, martingale diagnostics.
Derivatives	BSM, Bachelier, Black–76, CRR European/American, implied volatility, payoffs, analytic and finite-difference Greeks, European and Asian Monte Carlo.
Strategies/backtests	Buy-and-hold, SMA, momentum, mean reversion, volatility target, breakout, Bollinger, RSI, pairs; contracts, costs, trades, audits, sweeps, comparisons, HTML reports.
Statistics/ML	OLS/HAC/HC, rolling, quantile, polynomial, Ridge/Lasso/Elastic Net, logistic, factor, stationarity, bootstrap, permutation, FDR, cointegration, Granger-predictive tests, ARIMA, VAR, lag features, technical features, forward targets, walk-forward fit.
Risk/volatility	Performance and benchmark ratios, historical/parametric/Cornish–Fisher VaR, ES, CDaR, drawdowns, PSR/DSR, realized, Parkinson, Garman–Klass, EWMA, optional GARCH.
Portfolio/fixed income	Sample/EWMA/shrinkage covariance, minimum variance, maximum Sharpe, ERC, maximum diversification, efficient frontier, HRP, Black–Litterman, bonds, yield, duration, convexity, curve bootstrap.
Visualization	Market, performance, risk, stochastic, martingale, regression, portfolio, derivatives, ML, microstructure, generic heatmap, contour, static and interactive 3D modules.

C Reproducibility Checklist

A defensible experiment should record at least:

1. exact data source, extraction time, and dataset version;
2. universe construction and survivorship treatment;
3. price adjustment and corporate-action policy;
4. information-availability timestamp for every feature;
5. signal definition and all tested alternatives;

6. execution delay and execution-price convention;
7. rebalance schedule and weight normalization;
8. leverage, shorting, concentration, and cash constraints;
9. commission, spread, slippage, impact, and borrow assumptions;
10. in-sample, validation, and out-of-sample partitions;
11. number of trials and multiple-testing correction;
12. benchmark and risk-free-rate source;
13. package and dependency versions;
14. data, specification, and experiment fingerprints;
15. all random seeds;
16. rejected, failed, or superseded experiments.

D Artifact Structure

```
ASRQuant/
pyproject.toml, README.md, LICENSE, CITATION.cff
src/asrquant/
  api.py, data.py, providers.py, config.py
  simulation.py, martingales.py, derivatives.py
  statistics.py, machine_learning.py, validation.py
  strategies.py, backtest.py, audit.py, research.py
  metrics.py, volatility.py, optimization.py
  fixed_income.py, provenance.py, report.py, viz/
tests/
examples/
benchmarks/
docs/
paper/main.tex, paper/ASRQuant_paper.pdf, paper/figures/
```

References

- [1] R. Almgren and N. Chriss. Optimal execution of portfolio transactions. *Journal of Risk*, 3:5–39, 2001.
- [2] D. H. Bailey and M. López de Prado. The Deflated Sharpe Ratio: Correcting for selection bias, backtest overfitting, and non-normality. *Journal of Portfolio Management*, 40(5):94–107, 2014.
- [3] D. H. Bailey, J. M. Borwein, M. López de Prado, and Q. J. Zhu. The probability of backtest overfitting. *Journal of Computational Finance*, 20(4):39–69, 2017.
- [4] A. Battistella, J.-C. Bertrand, G. Coqueret, and N. McLoughlin. Design uncertainty in asset allocation backtesting. SSRN working paper, 2026.
- [5] P. R. Hansen. A test for superior predictive ability. *Journal of Business & Economic Statistics*, 23(4):365–380, 2005.
- [6] C. R. Harvey, Y. Liu, and H. Zhu. ... and the cross-section of expected returns. *Review of Financial Studies*, 29(1):5–68, 2016.
- [7] C. R. Harris et al. Array programming with NumPy. *Nature*, 585:357–362, 2020.
- [8] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

- [9] A. W. Lo. The statistics of Sharpe ratios. *Financial Analysts Journal*, 58(4):36–52, 2002.
- [10] H. Markowitz. Portfolio selection. *Journal of Finance*, 7(1):77–91, 1952.
- [11] W. McKinney. Data structures for statistical computing in Python. In *Proceedings of the 9th Python in Science Conference*, pages 56–61, 2010.
- [12] F. Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [13] S. Seabold and J. Perktold. Statsmodels: Econometric and statistical modeling with Python. In *Proceedings of the 9th Python in Science Conference*, 2010.
- [14] W. F. Sharpe. Mutual fund performance. *Journal of Business*, 39(1):119–138, 1966.
- [15] P. Virtanen et al. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17:261–272, 2020.
- [16] H. White. A Reality Check for data snooping. *Econometrica*, 68(5):1097–1126, 2000.
- [17] D. Yin, T. Miki, V. Lesnichenko, and V. Gural. Implementation Risk in Portfolio Backtesting: A Previously Unquantified Source of Error. SSRN working paper, March 19, 2026.
- [18] L. Bachelier. Théorie de la spéculation. *Annales scientifiques de l’École Normale Supérieure*, 17:21–86, 1900.
- [19] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.
- [20] R. C. Merton. Theory of rational option pricing. *Bell Journal of Economics and Management Science*, 4(1):141–183, 1973.
- [21] J. C. Cox, S. A. Ross, and M. Rubinstein. Option pricing: A simplified approach. *Journal of Financial Economics*, 7(3):229–263, 1979.
- [22] R. C. Merton. Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3(1–2):125–144, 1976.
- [23] O. Vasicek. An equilibrium characterization of the term structure. *Journal of Financial Economics*, 5(2):177–188, 1977.
- [24] J. C. Cox, J. E. Ingersoll, and S. A. Ross. A theory of the term structure of interest rates. *Econometrica*, 53(2):385–407, 1985.
- [25] S. L. Heston. A closed-form solution for options with stochastic volatility. *Review of Financial Studies*, 6(2):327–343, 1993.
- [26] W. K. Newey and K. D. West. A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica*, 55(3):703–708, 1987.
- [27] R. F. Engle and C. W. J. Granger. Co-integration and error correction: Representation, estimation, and testing. *Econometrica*, 55(2):251–276, 1987.
- [28] R. F. Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, 50(4):987–1007, 1982.
- [29] T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, 1986.

- [30] F. Black and R. Litterman. Global portfolio optimization. *Financial Analysts Journal*, 48(5):28–43, 1992.
- [31] O. Ledoit and M. Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365–411, 2004.
- [32] Alpha Vantage. API Documentation, accessed July 31, 2026. <https://www.alphavantage.co/documentation/>.
- [33] Binance. Official Spot API and WebSocket Market Streams Documentation, accessed July 31, 2026. <https://developers.binance.com/>.
- [34] Federal Reserve Bank of St. Louis. FRED API Documentation, accessed July 31, 2026. <https://fred.stlouisfed.org/docs/api/fred/>.