Architecting Autonomous Trading Agents: A Probabilistic and Filter-Driven Approach

Executive Summary

The evolution of algorithmic trading is entering a new phase, characterized by the synthesis of probabilistic state estimation, dynamic market regime analysis, and modular AI agent architectures. This paradigm moves beyond the limitations of rigid, single-purpose algorithms, paving the way for adaptive, goal-oriented intelligent systems capable of navigating the complex, non-stationary, and noisy environment of financial markets. The central thesis of this report is that a robust autonomous trading system is most effectively realized as a multi-agent collective. In this model, specialized agents, orchestrated by a central reasoning engine, collaborate to perceive, reason, and act under inherent market uncertainty.

The mathematical foundation for these agents is provided by the fusion of probabilistic frameworks—such as Bayesian inference, Hidden Markov Models (HMMs), and state-space models—with advanced signal processing techniques, notably the Kalman filter. These tools allow agents not merely to observe market data but to form probabilistic beliefs about latent market states, including a quantifiable measure of their own uncertainty. This capability is the cornerstone of building genuinely risk-aware and adaptive systems.

The primary architectural recommendation derived from this analysis is a modular, event-driven system structured around a "meta-strategy" paradigm. This architecture features a ReasoningAgent, potentially powered by a Large Language Model (LLM) framework like LangChain, which dynamically selects, parameterizes, and deploys trading strategies. These strategies are encapsulated as "tools" that the agent can invoke based on real-time market regime classifications provided by dedicated Signal and Analysis agents. This design prioritizes realism, adaptability, and explicit risk management, representing a significant advancement over the raw speed and inherent brittleness of simplistic vectorized approaches. This report provides a comprehensive blueprint for designing and implementing such systems, covering the full lifecycle from signal generation and validation to agent orchestration and

deployment.

Core Frameworks: From Signal to Strategy

The foundation of any autonomous trading system is its ability to transform raw, noisy market data into actionable, probabilistic insights. This process involves a sophisticated pipeline of signal filtering, state inference, and strategy formulation. This section deconstructs the core algorithmic and probabilistic frameworks that underpin these capabilities.

Taxonomy of Signal Filtering and Denoising Techniques

Filtering is not merely a pre-processing step for noise reduction; it is a fundamental modeling decision that defines the "state" of the market for an AI agent. While classical filters provide a smoothed observation, probabilistic filters yield a posterior distribution over a latent state, including a quantifiable measure of uncertainty—a critical input for any risk-aware system.

State-Space Models and Statistical Filters

These models operate on the assumption that an unobserved (latent) state variable generates the noisy observations we see in market data.[1] The goal of the filter is to infer this hidden state.

Kalman Filter: The Kalman filter is an optimal recursive algorithm for estimating the state of a linear dynamic system from a series of noisy measurements.[3] It operates in a two-step loop: prediction and update.[3] The system is defined by a set of matrices: the state transition matrix (F), the observation matrix (H), the process noise covariance (Q), and the measurement noise covariance (R).[3]

The state update equations are:

1. Prediction:

   $$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k$$
   $$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$$

2. Update:

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - H_k \hat{x}_{k|k-1})$$
$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

Here, $\hat{x}$ is the state estimate and P is the error covariance matrix. The state vector $\hat{x}$ can be defined to model not just price, but also its derivatives like velocity and acceleration, allowing the filter to function as a sophisticated trend and momentum estimator.[3]

Crucially, the error covariance matrix P represents the filter's uncertainty about its own state estimate. A risk-aware AI agent can directly ingest this uncertainty. For example, a spike in the trace of the covariance matrix indicates high uncertainty in the state estimate, perhaps due to a volatile news event. The agent's risk management module can then use this information to reduce position size, widen stops, or halt trading, without needing a separate volatility model. This elevates the filter from a simple "tool" to a core component of the agent's perception and reasoning system.[3] Applications in finance include dynamic hedge ratio estimation in pairs trading and volatility estimation.[5]

Bayesian Filtering: The Kalman filter is a specific instance of the broader Bayesian filtering paradigm.6 This framework formalizes the process of belief updating, where a prior probability distribution over a market state, $P(H)$, is combined with new evidence (market data), characterized by the likelihood $P(E|H)$, to compute an updated posterior probability distribution, $P(H|E)$, via Bayes' theorem 6:

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

This iterative process of updating beliefs is fundamental to how an adaptive AI agent should reason and learn from incoming market information.6 Simpler models like Naive Bayes classifiers can also be applied for market prediction tasks.9

Frequency-Domain and Polynomial Filters

These filters operate on a sliding window of data and generally make fewer assumptions about the underlying data generating process than state-space models.

Savitzky-Golay (SG) Filter: The SG filter performs a local least-squares polynomial fit to a window of data points. This makes it a type of finite impulse response (FIR) filter that can be more effective at preserving higher-order moments of the signal, such as the shape of peaks and valleys, compared to a simple moving average.[10] The key parameters are the window

length (must be odd) and the polynomial degree.[12] A higher degree can capture more complex curves but risks overfitting to noise, while a larger window provides more smoothing but increases lag.[12] However, its primary limitation is its poor frequency response, characterized by mediocre stopband attenuation, which makes it less effective at suppressing high-frequency noise compared to filters designed in the frequency domain.[13]

Butterworth Filter: The Butterworth filter is a type of signal processing filter designed to have a frequency response that is "maximally flat" in the passband, meaning it introduces no ripples.[15] In trading, it is typically used as a low-pass filter to smooth price data and identify underlying trends with less lag than simple moving averages.[17] The filter's order, n, determines the steepness of the roll-off into the stopband, with the response attenuating at $-20n$ dB/decade.[15] A higher order provides a sharper cutoff but can introduce more phase distortion (lag).[15] A two-pole Butterworth filter is often cited as a good compromise for reducing lag while maintaining a smooth output.[17]

Hodrick-Prescott (HP) Filter: Originating in macroeconomics, the HP filter decomposes a time series y_t into a trend component τ_t and a cyclical component c_t by solving a minimization problem 18:
$$ \min_{\tau} \left(\sum_{t=1}^{T}{(y_{t}-\tau_{t})^{2}} + \lambda \sum_{t=2}^{T-1}{[(\tau_{t+1}-\tau_{t})-(\tau_{t}-\tau_{t-1})]^{2}} \right) $$
The smoothing parameter λ penalizes variations in the trend's growth rate.19 While useful for historical analysis, the standard two-sided HP filter is non-causal; it uses future data points to determine the trend at the current time point.19 This results in a "repainting" issue, where the entire historical trend line is revised as each new data point arrives, making it fundamentally unsuitable for generating signals in a live trading or realistic backtesting environment.20

Machine Learning-Based Filters & Denoising

Modern techniques leverage neural networks to learn optimal, data-driven filtering functions, often outperforming classical methods on noisy and non-linear financial data.

Autoencoders (AE): Denoising autoencoders are unsupervised neural networks trained to reconstruct a clean input signal from a version that has been intentionally corrupted with noise.[21] To perform this task, the network must learn a robust, compressed representation (in the "bottleneck" layer) of the underlying data structure, effectively learning to separate signal from noise.[21]

Recurrent Neural Networks (LSTM/GRU): Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks are designed to model sequential data and can be applied to

both forecasting and denoising tasks.[23] Hybrid architectures, such as an Autoencoder-LSTM (AE-LSTM) or AE-GRU, combine the powerful non-linear feature extraction of an autoencoder with the temporal modeling capabilities of an RNN.[24] These models first encode the input time series into a latent representation and then use this representation as input to an LSTM/GRU decoder for prediction, demonstrating superior performance in forecasting volatile assets.[24]

Spectral Filtering with Transformers: A state-of-the-art approach involves combining learnable frequency filters with Transformer-based models.[26] Transformers can suffer from a bias toward low-frequency components in the data.[26] By adding a spectral filtering layer at the beginning of the architecture, the model can be explicitly guided to capture high-frequency patterns, leading to significant improvements in forecasting performance with minimal additional computational cost.[26]

The following table provides a comparative summary of these filtering methods.

Table 1: Comparative Analysis of Filtering Methods

Method	Underlying Assumption	Key Parameters	Computational Complexity	Latency Profile	Output Type	Primary Use Case	Key Weakness
Kalman Filter	Linear dynamic system with Gaussian noise	F, H, Q, R	Moderate, $O(k^3)$ where k is state dimension [30]	Low (recursive)	Distribution (State & Covariance)	Dynamic state estimation, trend tracking, pairs trading	Assumes linearity and Gaussian noise; requires good process model [31]
Savitzky-Golay	Data is locally polynomial	Window size, polynomial degree	Low (FIR filter)	Low (causal if one-sided)	Point Estimate	Smoothing while preserving peaks;	Poor high-frequency noise suppression;

						derivative estimation	can overfit noise [14, 31]
Butterworth	Signal and noise occupy different frequency bands	Filter order, cutoff frequency	Low (IIR filter)	Low (causal)	Point Estimate	Trend identification with minimal passband distortion and low lag [17]	Slower roll-off than Chebyshev/elliptic filters; can introduce phase lag [15]
Hodrick-Prescott	Series is composed of a slowly varying trend and a cycle	Smoothing parameter λ	Moderate	High (non-causal)	Point Estimate	Macroeconomic trend-cycle decomposition (offline analysis)	"Repaints" history; unsuitable for real-time signal generation [20]
Denoising Autoencoder	Signal is lower-dimensional than noise	Network architecture, latent dim, corruption level	High (training), Low (inference)	Low (inference)	Point Estimate	Non-parametric noise reduction; feature extraction	Requires large datasets for training; can be a "black box" [21]
Spectral Filter +	Signal has importa	Filter parameters,	High (training),	Moderate (inferen	Point Estimate	High-performance	High computational

Transformer	nt frequency domain features	attention heads, layers	Moderate (inference)	ce)		forecasting on complex, multi-frequency series	requirements for training; complex architecture [26]

Probabilistic Models for Market Regime and State Inference

Financial markets are inherently non-stationary; their statistical properties change over time. Probabilistic models designed to identify and adapt to these changes, or "regimes," are essential for building robust trading strategies.

Regime Detection with Hidden Markov Models (HMMs)

HMMs are a powerful statistical tool for modeling time series that are believed to be generated by an unobserved, underlying process that switches between a finite number of states.[32] For financial markets, these hidden states naturally correspond to market regimes, such as 'bull market' vs. 'bear market' or 'low volatility' vs. 'high volatility'.[33]

An HMM is defined by three key components [32]:

1. **Hidden States (S):** A finite set of unobservable states. For example, $S = \{\text{Low-Vol}, \text{High-Vol}\}$.
2. **Transition Probabilities (A):** A matrix $A_{ij} = P(S_{t+1}=j | S_t=i)$ that defines the probability of moving from one state to another. High diagonal values imply regime persistence.[33]
3. **Emission Probabilities (B):** A set of probability distributions $B_i(O_t) = P(O_t | S_t=i)$ that define the likelihood of observing the data O_t (e.g., daily returns) given the system is in hidden state i. For financial returns, these are often modeled as Gaussian distributions with a unique mean and variance for each state.[34]

By fitting an HMM to a time series of returns (e.g., using the Baum-Welch algorithm), one can solve for the most likely sequence of hidden states (using the Viterbi algorithm) and, crucially

for trading, infer the probability of being in each state at the current time.[36] This probabilistic classification of the market regime is a direct and powerful input for an AI agent tasked with dynamic strategy selection.[37] Python libraries such as hmmlearn provide accessible implementations for fitting models like GaussianHMM to financial data.[34]

Volatility Clustering (ARCH/GARCH)

A well-documented stylized fact of financial returns is volatility clustering: large price changes tend to be followed by large changes, and small changes by small changes.[39] Autoregressive Conditional Heteroskedasticity (ARCH) and Generalized ARCH (GARCH) models explicitly capture this time-varying volatility.[40]

The ARCH(q) model defines the conditional variance σ^2_t as a function of past squared error terms (ϵ^2_{t-i}) 40:

$$\sigma^2_t = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon^2_{t-i}$$
The GARCH(p,q) model extends this by also including past conditional variances (σ^2_{t-j}), allowing for a more parsimonious representation of volatility persistence 39:
$$ \sigma^2_t = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon^2_{t-i} + \sum_{j=1}^{p} \beta_j \sigma^2_{t-j} $$
The output of a GARCH model provides a continuous, forward-looking estimate of volatility. This can be interpreted as a proxy for the market's "risk regime" and can be used by a risk management agent to scale position sizes or adjust stop-loss levels dynamically.

Predictability and Complexity via Entropy Measures

Entropy, a concept from information theory, measures the amount of uncertainty or randomness in a system.[41] In finance, it can be used to quantify the predictability of a time series.

- **Shannon and Approximate Entropy:** A time series with low entropy is more regular and predictable, while one with high entropy is more random.[41] According to the Efficient Market Hypothesis, asset prices should follow a random walk, implying high entropy. Periods of significantly low entropy may therefore signal market inefficiency and the presence of statistical arbitrage opportunities.[42] By calculating entropy over a rolling window, an agent can detect transitions between efficient (random) and inefficient

(predictable) market regimes.[42]

- **Transfer Entropy:** This is a non-parametric, model-free measure used to quantify the directed flow of information from one time series to another.[46] It generalizes the concept of Granger causality to non-linear relationships.[47] For a multi-asset trading agent, transfer entropy is an invaluable tool for mapping the network of influence and lead-lag relationships within its investment universe, helping to identify sources of systemic risk or information contagion.[46]

Modeling Dependencies with Probabilistic Graphical Models (PGMs)

PGMs use graph-based representations to encode conditional dependence structures among a set of random variables, making them ideal for modeling complex financial systems.[48]

- **Bayesian Networks (BNs):** BNs are Directed Acyclic Graphs (DAGs) where nodes represent variables (e.g., interest rates, VIX, sector ETFs, individual stocks) and directed edges represent conditional dependencies.[49] BNs excel at data fusion, providing a framework to combine expert domain knowledge (which can inform the graph structure) with statistical relationships learned from data (the conditional probability tables).[49] Once constructed, a BN can perform probabilistic inference, such as calculating the probability of a market downturn given a spike in the VIX and rising interest rates.[49]
- **Dynamic Bayesian Networks (DBNs):** DBNs extend BNs to model temporal processes by representing variables at different points in time as nodes in an "unrolled" graph.[51] This allows them to explicitly model how system states and their interdependencies evolve over time, making them a powerful tool for dynamic market modeling and forecasting.[51]

Trading Strategy Archetypes and Their Probabilistic Formulation

This section connects the abstract probabilistic models to concrete, implementable trading strategies, framing them in a way that is directly consumable by an intelligent agent.

Mean-Reversion and Statistical Arbitrage

Statistical arbitrage strategies are built on the identification of a portfolio or "spread" of

assets whose value exhibits mean-reverting behavior.[53] The canonical example is pairs trading, but the concept extends to complex baskets of assets identified through statistical methods like cointegration or principal component analysis (PCA).[54]

The probabilistic model most commonly used for a mean-reverting spread is the Ornstein-Uhlenbeck (OU) process, a stochastic differential equation given by 56:

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

where X_t is the spread value, θ is the long-term mean to which it reverts, κ is the speed of mean reversion, σ is the volatility, and dW_t is a Wiener process. An agent trades this by selling the spread when X_t is significantly above θ and buying it when it is significantly below.
A more advanced approach uses state-space models to represent the spread, allowing the parameters (κ, θ, σ) to be time-varying and estimated dynamically, for instance with a Kalman filter. This provides a more adaptive model that can react to structural changes in the relationship between the assets.[55]

Momentum and Trend Detection

Momentum strategies are based on the empirical observation that assets with strong past performance tend to continue performing well in the short to medium term.[57] While often implemented with simple technical indicators, a more rigorous probabilistic formulation provides deeper insights.

The work of Liu provides an explicit solution for an optimal dynamic trading strategy for an asset with momentum.[59] The key finding is that the expected return of a momentum asset is non-Markovian; it depends not just on the current momentum value but on the entire historical price path within the look-back period. For example, a "rebound" path and a "hump-shaped" path with identical final momentum values will have different optimal portfolio allocations.[59] This path-dependence makes the problem ideally suited for sequence models like LSTMs or Transformers, which can learn to map complex path histories to optimal actions.

Multi-Signal Fusion and Decision Theory

A sophisticated agent will inevitably generate multiple, sometimes conflicting, trading signals.

A formal framework is needed to fuse these signals and make a final, optimal decision.

- **Bayesian Signal Fusion:** As discussed previously, Bayesian Networks provide a natural and coherent framework for this task. Each signal (e.g., from a momentum indicator, a mean-reversion model, or a sentiment analysis tool) is treated as a piece of evidence. The network then updates the posterior probability of a trading hypothesis (e.g., $P(\text{Price Up} | \text{Evidence})$).[49] This process transforms a collection of disparate signals into a single, unified probabilistic forecast.
- **Decision Theory and Optimal Sizing:** Given a probabilistic forecast, decision theory dictates how to act.
    - **Expected Utility Theory:** Instead of maximizing raw profit, an agent can be designed to maximize the expected utility of its wealth, which incorporates its risk aversion.
    - Kelly Criterion: This powerful formula provides the optimal fraction f^* of capital to allocate to a given bet to maximize the long-term logarithmic growth of wealth.60 The basic formula is:

      $$f^* = \frac{bp - q}{b}$$

      where p is the probability of winning, $q=1-p$ is the probability of losing, and b is the payout odds (e.g., if you risk 1 to win 2, $b=2$).60 If f^* is negative, the bet has a negative expected log-return and should not be taken. In practice, traders often use a "fractional Kelly" (e.g., half-Kelly or quarter-Kelly) to reduce the high volatility and drawdown risk associated with the full Kelly bet.60 The Kelly criterion can be extended to continuous distributions, where it relates the optimal leverage to the strategy's edge-to-volatility ratio (or Sharpe ratio), and to multi-asset portfolio optimization.61

Implementation Patterns for AI Agents

Translating the core theoretical frameworks into a functional, robust, and autonomous trading system requires careful consideration of implementation patterns, from the validation environment to the agent architecture itself. This section provides practical blueprints for these critical engineering tasks.

Advanced Backtesting and Validation Frameworks

Backtesting is the process of evaluating a trading strategy on historical data. For AI-driven and probabilistic strategies, simplistic backtesting approaches are insufficient and often dangerously misleading. The backtesting engine for an AI agent, particularly one employing reinforcement learning, is not merely a validation tool but serves as its training environment or "sandbox." The fidelity of this environment directly dictates the quality and real-world viability of the agent's learned policy.[63] An agent trained in an idealized environment that ignores market frictions will learn a naive policy that is destined to fail in live trading.

Vectorized vs. Event-Driven Architectures

Two primary paradigms exist for backtesting architecture.[65]

- **Vectorized Backtesting:** This approach leverages the power of array programming libraries like NumPy and pandas to process entire time series of data in large, vectorized chunks.[65] It is exceptionally fast, making it suitable for rapid prototyping and large-scale parameter sweeps of simpler, low-frequency strategies (e.g., daily rebalancing).[67] vectorbt is a leading example of a framework optimized for this paradigm.[67] However, its speed comes at the cost of realism. Vectorized backtesters struggle to model intra-bar price movements, order queue dynamics, partial fills, and market impact, typically assuming trades execute at the next bar's open or close price.[65]
- **Event-Driven Backtesting:** This architecture simulates the flow of time by processing market data sequentially, one "event" (e.g., a new price tick, a bar closing) at a time in a chronological loop.[69] While significantly slower and more complex to implement, it offers far higher fidelity.[69] It can realistically model various order types (market, limit, stop), slippage, commissions, and the interaction between the agent's orders and the market data stream.[65] This realism is non-negotiable for testing higher-frequency strategies and is the required paradigm for training and validating stateful, reactive AI agents. Backtrader and Zipline are classic open-source examples of this approach.[67]

Modeling Market Frictions

A high-fidelity backtester must accurately model the costs and imperfections of trading.

- **Slippage and Liquidity:** Slippage is the difference between the expected fill price of an order and the price at which it is actually executed.[71] It is influenced by market volatility, order size, and latency.[72] Backtesting frameworks must move beyond assuming perfect fills and incorporate slippage models, which can range from a simple fixed percentage to

more sophisticated variable models that account for the bid-ask spread, order book depth, and the order's size relative to available volume.[72] A crucial practice is to model liquidity constraints by capping simulated order sizes as a fraction of the historical trading volume (e.g., no more than 5% of average daily volume) to avoid the unrealistic assumption of infinite liquidity.[72]

- **Latency:** In high-frequency trading (HFT), delays measured in microseconds can determine profitability.[74] An event-driven backtester should model the various sources of latency, including the time for data to reach the agent (data feed latency), the time for the agent to process the data and generate a signal (processing latency), and the time for the order to travel to the exchange and be executed (network and exchange latency).[77]

Validation for Non-IID Time Series

Financial time series are not independent and identically distributed (IID), which invalidates standard machine learning validation techniques like random k-fold cross-validation. Using future data to train a model that predicts the past (lookahead bias) is a fatal flaw that leads to wildly inflated performance metrics.[78]

- **Time Series Cross-Validation (Rolling-Origin CV):** The standard method for time series involves creating multiple train-test splits that preserve the temporal order of the data.[79] In each fold, the model is trained on data up to a certain point in time and tested on the immediately following period. The training window can be either expanding (using all prior data) or rolling (using a fixed-size window).[78] Scikit-learn's TimeSeriesSplit provides a convenient implementation of this process.[79]
- **Purged and Embargoed K-Fold CV:** As detailed by Marcos López de Prado, even standard rolling-origin CV can suffer from information leakage when labels are generated from overlapping data points (e.g., a 20-day forward return label).[81] To combat this, he proposes two crucial modifications:
    1. **Purging:** Removing any training data points whose labels are derived from information that overlaps with the test period.
    2. Embargoing: Adding a small time gap between the end of the training set and the start of the test set to mitigate the effects of serial correlation.
    These techniques are the state-of-the-art for preventing backtest overfitting and achieving robust out-of-sample validation.81

Agent-Centric Validation (Reinforcement Learning)

Backtesting reinforcement learning (RL) agents presents unique challenges because the agent's actions actively influence the state of the environment (i.e., its portfolio and capital), a feedback loop that is absent in supervised learning.[82] A simple historical replay is insufficient. Robust validation requires either building a market simulator trained on historical data or using off-policy evaluation methods, which assess a new policy's performance based on data collected by a different policy. The Backtesting.py library, for example, can be used to backtest supervised ML models and provides a foundation that could be adapted for RL agent evaluation.[83]

The following table compares leading open-source Python backtesting frameworks.

Table 2: Python Backtesting Framework Comparison

Framework	Primary Architecture	Speed	Realism (Friction Modeling)	Key Features	Live Trading Integration	Best For
vectorbt [67]	Vectorized	Very High	Low (relies on user-defined heuristics)	Blazing-fast parameter sweeps, interactive dashboards, Numba optimization	No (designed for research)	Rapid prototyping, large-scale parameter optimization, portfolio-level research.
Backtrader [67]	Event-Driven	Low to Moderate	High (flexible commission/slippage models)	Mature, highly flexible, large community, multi-timeframe	Yes (via broker integrations)	Developing and testing complex, event-driven strategies;

				support		transition to live trading.
Zipline-reloaded [70]	Event-Driven	Low	Moderate to High	Pipeline API for factor analysis, originally from Quantopian	Yes (via forks like zipline-live)	Academic research, equity factor strategies, legacy Quantopian users.
Freqtrade [85]	Event-Driven	Moderate	High (customizable)	Crypto-focused, includes hyperparameter optimization, WebUI, Telegram integration	Yes (natively supported)	All-in-one solution for developing, backtesting, and deploying crypto trading bots.

Blueprint for an Autonomous Trading Agent

Synthesizing the preceding concepts, this section outlines a concrete architectural blueprint for an autonomous trading agent system, leveraging a multi-agent design for modularity and specialization.

Modular Agent Design

A multi-agent architecture is preferable to a monolithic design for managing complexity and promoting specialization.[86] Each agent is responsible for a distinct part of the trading

workflow, communicating via a message bus or an orchestration layer.

- **DataIngestionAgent:** Subscribes to market data feeds (e.g., via WebSocket), normalizes data formats, and publishes MarketEvent objects.
- **SignalFilteringAgent:** Subscribes to MarketEvents. Applies a suite of filters (e.g., Kalman, Butterworth) to generate cleaned signals and probabilistic state estimates (e.g., trend, velocity, uncertainty), publishing SignalEvent objects.
- **RegimeAnalysisAgent:** Also subscribes to MarketEvents. Employs HMMs, GARCH, and/or entropy measures to classify the current market regime, publishing RegimeEvent objects containing regime probabilities.
- **StrategySelectionAgent (Reasoning Agent):** The central decision-maker. It subscribes to SignalEvents and RegimeEvents. Based on the current regime and signal state, it selects an appropriate trading strategy (conceived as a "tool") and determines its parameters. It then generates a SignalEvent with a trading directive (e.g., "LONG MSFT").
- **RiskManagementAgent:** Subscribes to SignalEvents and portfolio state updates. It calculates the optimal position size based on the Kelly Criterion, forecast probabilities, and model uncertainty (e.g., from the Kalman filter's covariance matrix). It can veto or modify proposed trades, publishing an OrderEvent.
- **ExecutionAgent:** Subscribes to OrderEvents. It translates the order into the specific format required by the brokerage API, manages the order lifecycle (e.g., handling partial fills), and publishes FillEvent objects upon execution.

Orchestration and Reasoning with LLM Frameworks

Modern LLM agent frameworks provide powerful tools for orchestrating these specialized agents.

- **LangChain / LangGraph:** LangChain's extensive library of integrations and its LangGraph extension make it a prime candidate for building the core workflow.[87] The trading process can be modeled as a directed graph where nodes represent agents or tools. This provides explicit control over the flow of information and allows for complex, stateful interactions, such as feedback loops.[89]
- **AutoGen:** Developed by Microsoft, AutoGen excels at facilitating complex, conversational collaboration between agents.[90] This pattern is ideal for scenarios where agents need to negotiate or deliberate. For instance, the StrategySelectionAgent could initiate a "conversation" with the RiskManagementAgent to determine an appropriate risk budget before finalizing a trade decision.[92]
- **CrewAI:** CrewAI offers a higher-level, role-based abstraction for multi-agent collaboration, which maps naturally to the specialized agent design proposed above.[93] It simplifies the process of defining agents with specific goals and tools, and orchestrating

their sequential or parallel execution.[89]

For a production system, a hybrid approach is often optimal. LangChain could provide the foundational data integrations and tool definitions, while AutoGen or CrewAI could manage the high-level reasoning and collaboration between agents.[94]

Table 3: AI Agent Orchestration Frameworks Comparison

Framework	Core Architecture	Agent Coordination Pattern	Tool Integration	State Management	Best For Financial Use Case
LangChain/LangGraph	Modular, graph-based (LangGraph)	Chains, Graphs, Routers	Extensive (600+ integrations) [93]	Explicit state passing in graphs	Building complex, production-grade data processing and strategy execution pipelines.[94]
AutoGen	Conversational, event-driven	Multi-agent conversation, hierarchical chats	Flexible, via function/tool registration	Contextual, conversation-driven	Simulating collaborative decision-making, such as a team of analyst agents debating a forecast.[90]
CrewAI	Role-based, built on LangChain	Task delegation, sequential/hierarchical crews	Inherits LangChain's ecosystem	Built-in short/long-term memory	Rapidly prototyping role-based agent teams for specific tasks (e.g.,

					research, analysis, reporting).[9] [4]

Risk-Aware AI Agents

A truly intelligent trading agent must be fundamentally risk-aware. This goes beyond simple stop-losses and involves architectural patterns that explicitly handle uncertainty.[95]

- **Uncertainty Propagation:** The system should operate on probability distributions, not just point estimates. A Bayesian forecasting model outputs a posterior distribution over future prices; this entire distribution, not just its mean, should be passed to the RiskManagementAgent. This allows for more nuanced risk assessment, such as calculating Value at Risk (VaR) or Expected Shortfall (ES) directly from the forecast distribution.
- **Hybrid Reactive-Deliberative Architecture:** The overall agent system can be viewed as a hybrid architecture.[96] The StrategySelectionAgent operates on a slower, deliberative timescale (e.g., minutes to hours), planning the overall strategy. In contrast, the ExecutionAgent must be reactive, capable of responding to microsecond-level price ticks to manage an order (e.g., adjusting a limit order price based on order book dynamics).

Multi-Agent Reinforcement Learning (MARL) for Coordination

For managing a portfolio of interacting strategies or assets, MARL provides a framework for agents to learn optimal coordination policies.[64] Instead of pre-programming their interaction rules, agents learn through trial and error to cooperate or compete to maximize a shared objective function (e.g., portfolio Sharpe ratio).[64] Frameworks like MAPS (Multi-Agent reinforcement learning-based Portfolio management System) use cooperative MARL, where agents are rewarded for creating diversified portfolios that collectively improve the risk-adjusted return of the entire system.[98]

Pitfalls & Anti-Patterns

The path to building a successful autonomous trading system is fraught with potential errors. Awareness of common pitfalls in filtering, backtesting, and agent design is critical for avoiding costly mistakes.

Filtering & Signal Processing

- **Phase Distortion & Lag:** Every filter, by its nature, introduces a time delay or lag. Causal filters, which only use past data, will always lag the real-time price. This can cause trading signals to be generated too late, after a significant portion of a price move has already occurred. The phase response of a filter quantifies this distortion across different frequencies, and it must be carefully analyzed when selecting a filter for a latency-sensitive strategy.
- **The "Repainting" Anti-Pattern:** This is one of the most dangerous pitfalls in signal generation. It occurs when using non-causal filters, like the standard two-sided Hodrick-Prescott filter, in a simulated or live environment.[20] These filters use future information to calculate the current value, meaning their historical output changes as new data becomes available. A strategy backtested on "repainting" data will appear deceptively profitable, as it is implicitly trading with perfect foresight. All signal generation logic must be strictly causal, using only data that would have been available at that point in time.
- **Complexity vs. Latency:** There is a direct trade-off between the sophistication of a filter and its computational cost. A complex filter like an Extended Kalman Filter has a computational complexity that scales with the size of the state vector (e.g., $O(L_x^2)$).[30] While this may be acceptable for a low-frequency strategy, it can introduce unacceptable processing latency in a high-frequency context where decisions are made in microseconds.[77]

Backtesting & Validation

- **Overfitting and Data Snooping:** These are the cardinal sins of quantitative research. Overfitting occurs when a model learns the noise in the historical data rather than the true underlying signal, leading to excellent backtest performance but poor live results.[82] Data snooping is the practice of repeatedly testing different ideas on the same dataset until one appears to work by chance.[82] As Marcos López de Prado argues, backtesting should not be used as a research tool; feature importance analysis should be used

instead. Every backtest performed should be logged, and performance metrics should be "deflated" to account for the number of tests run.[81]

- **Unrealistic Friction Modeling:** A common failure mode is to ignore or grossly underestimate the impact of market frictions. Backtests that do not model transaction costs (commissions, fees), slippage, and market impact can make a fundamentally unprofitable strategy appear highly lucrative.[72]
- **Lookahead Bias:** This subtle error occurs when information that would not have been available at the time of a decision leaks into the backtest.[65] Common sources include using data adjusted for corporate actions (e.g., splits, dividends) before the ex-date, or normalizing a data series using statistics (mean, standard deviation) calculated over the entire dataset, including future data.

AI Agent Architecture

- **Monolithic Agent Design:** An anti-pattern is the attempt to build a single, "do-it-all" agent that handles data ingestion, signal processing, strategy logic, and execution.[96] This approach leads to tightly coupled, unmanageable code that is difficult to test, debug, and extend. A modular architecture with specialized, single-responsibility agents is far more robust and scalable.[86]
- **Brittle Tool Integration:** Agents can become too tightly coupled to the specific implementation details of their tools (e.g., a particular API or data format). A better pattern is to use abstract interfaces and standardized data models, allowing tools to be swapped or updated without breaking the agent's core logic.
- **Lack of Observability:** In a complex, asynchronous multi-agent system, tracing the chain of events and reasoning that led to a specific trading decision can be extremely difficult. This "black box" behavior is unacceptable in a production trading environment. The architecture must incorporate robust logging, tracing (e.g., using OpenTelemetry), and visualization tools to ensure that every decision is auditable and debuggable.[92]

References & Open-Source Toolkits

This section provides a curated list of seminal academic works and production-ready open-source frameworks to guide further research and practical implementation.

10 Seminal Research Papers & Books

1. **Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press.** The foundational and most-cited textbook on reinforcement learning, providing the theoretical underpinnings for RL-based agents.[99]
2. **De Prado, M. L. (2018). *Advances in Financial Machine Learning*. Wiley.** A modern classic that has reshaped quantitative finance by introducing rigorous methodologies for data structuring, labeling, feature importance, and, most critically, robust backtesting (e.g., purged and embargoed cross-validation) to prevent overfitting.[81]
3. **Hamilton, J. D. (1989). "A new approach to the economic analysis of nonstationary time series and the business cycle." *Econometrica*.** The seminal paper that introduced Markov-switching models, providing the basis for modern market regime detection techniques.[33]
4. **Engle, R. F. (1982). "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation." *Econometrica*.** The foundational paper for ARCH models, which revolutionized volatility modeling.[39]
5. **Gatev, E., Goetzmann, W. N., & Rouwenhorst, K. G. (2006). "Pairs trading: Performance of a relative-value arbitrage rule." *Review of Financial Studies*.** A classic and widely cited empirical study that validated the profitability of the distance-based pairs trading strategy.[54]
6. **Avellaneda, M., & Lee, J. H. (2010). "Statistical arbitrage in the U.S. equities market." *Quantitative Finance*.** Introduced a modern, factor-based approach to statistical arbitrage using Principal Component Analysis (PCA) to construct mean-reverting portfolios.[54]
7. **Epstein, E. L., et al. (2025). "Attention Factors for Statistical Arbitrage." *arXiv preprint*.** A state-of-the-art paper demonstrating how attention mechanisms, borrowed from deep learning, can be used to jointly learn arbitrage factors and trading policies, significantly outperforming traditional methods.[103]
8. **Moody, J., & Wu, L. (1997). "Optimization of trading systems and portfolios." *Neural Networks in Financial Engineering*.** An early and influential paper on applying reinforcement learning directly to optimize trading strategies, laying the groundwork for modern RL-based trading agents.[104]
9. **Deng, Y., et al. (2016). "Deep Direct Reinforcement Learning for Financial Signal Representation and Trading." *IEEE Transactions on Neural Networks and Learning Systems*.** A key paper demonstrating an end-to-end deep reinforcement learning approach that learns feature representations and trading policies directly from market data, without relying on traditional technical indicators.[83]
10. **Koller, D., & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.** The definitive textbook on PGMs, covering the theory and application of Bayesian Networks, Markov Networks, and their dynamic extensions, which

are essential for modeling complex dependencies in financial systems.

5 Production-Ready Open-Source Frameworks

1. **Backtesting: Backtrader**
   - An event-driven, feature-rich, and mature Python library. Its flexibility and focus on realism make it an ideal choice for developing and testing complex, stateful AI agent strategies before deploying them to live markets.[67] While vectorbt is faster for research, its vectorized nature is less suited for simulating the sequential decision-making of an agent.[67] For cryptocurrency trading, Freqtrade offers an excellent all-in-one solution with backtesting, optimization, and live trading capabilities out-of-the-box.[85]
2. **Probabilistic Programming: PyMC**
   - A powerful and user-friendly Python library for Bayesian modeling and inference.[106] It provides state-of-the-art MCMC (NUTS) and variational inference algorithms, making it essential for implementing custom Bayesian filters, probabilistic risk models, and Bayesian decision frameworks within an agent.[106]
3. **Probabilistic Graphical Models: pgmpy**
   - A dedicated Python library for working with Probabilistic Graphical Models. It provides a comprehensive toolset for defining, learning parameters for, and performing inference on Bayesian Networks and Dynamic Bayesian Networks, enabling the modeling of complex causal and probabilistic relationships between financial variables.[107]
4. **Agent Orchestration: LangChain**
   - The most mature and widely adopted framework for building applications with Large Language Models.[87] Its vast ecosystem of tool integrations, combined with the power of LangGraph for defining complex, stateful, and cyclical agent workflows, makes it the most robust choice for orchestrating the multi-agent financial system described in this report.[88]
5. **Quantitative Finance Toolkit: QuantConnect LEAN**
   - While a full platform, its open-source LEAN engine is an institutional-grade, multi-asset backtesting and live trading engine written in C# with full Python support.[108] It provides a battle-tested, high-performance foundation with realistic modeling of market frictions, making it a production-ready alternative to building an entire event-driven system from scratch.[108]

Works cited

1. milvus.io, accessed October 30, 2025,

https://milvus.io/ai-quick-reference/what-are-statespace-models-in-time-series-analysis#:~:text=State%2Dspace%20models%20(SSMs),that%20depend%20on%20that%20state.
2. What are state-space models in time series analysis? - Milvus, accessed October 30, 2025, https://milvus.io/ai-quick-reference/what-are-statespace-models-in-time-series-analysis
3. Kalman Filter in Python - GeeksforGeeks, accessed October 30, 2025, https://www.geeksforgeeks.org/python/kalman-filter-in-python/
4. How to Use Kalman Filters for Time Series Analysis in Python - Statology, accessed October 30, 2025, https://www.statology.org/how-to-use-kalman-filters-for-time-series-analysis-in-python/
5. Kalman Filter Python: Tutorial and Strategies - QuantInsti Blog, accessed October 30, 2025, https://blog.quantinsti.com/kalman-filter/
6. Bayesian Statistics in Finance: A Trader's Guide to Smarter Decisions, accessed October 30, 2025, https://www.interactivebrokers.com/campus/ibkr-quant-news/bayesian-statistics-in-finance-a-traders-guide-to-smarter-decisions/
7. Introduction to Bayesian Statistics in Finance and Algorithmic Trading, accessed October 30, 2025, https://blog.quantinsti.com/introduction-to-bayesian-statistics-in-finance/
8. Bayesian Inference in Quant Trading | QuestDB, accessed October 30, 2025, https://questdb.com/glossary/bayesian-inference-in-quant-trading/
9. Stock Market Prediction with Gaussian Naïve Bayes Machine Learning Algorithm, accessed October 30, 2025, https://www.researchgate.net/publication/353642287_Stock_Market_Prediction_with_Gaussian_Naive_Bayes_Machine_Learning_Algorithm
10. 10.3 Savitzky-Golay filter - Time Series, accessed October 30, 2025, https://timeseries.sci.muni.cz/index.php?pg=chapter-10--10-3-savitzky-golay-filter
11. Savitzky–Golay filter - Wikipedia, accessed October 30, 2025, https://en.wikipedia.org/wiki/Savitzky%E2%80%93Golay_filter
12. Savitzky Golay Filtering for Time Series Denoising | Nixtla, accessed October 30, 2025, https://www.nixtla.io/blog/polynomial_filtering
13. python - How to compare scipy noise filters? - Stack Overflow, accessed October 30, 2025, https://stackoverflow.com/questions/60093456/how-to-compare-scipy-noise-filters
14. Why and How Savitzky–Golay Filters Should Be Replaced | ACS ..., accessed October 30, 2025, https://pubs.acs.org/doi/10.1021/acsmeasuresciau.1c00054
15. Butterworth filter - Wikipedia, accessed October 30, 2025, https://en.wikipedia.org/wiki/Butterworth_filter
16. Butterworth Filter Design with a Low Pass Butterworth - Electronics Tutorials, accessed October 30, 2025,

https://www.electronics-tutorials.ws/filter/filter_8.html
17. Butterworth — Indicators and Strategies — TradingView — India India, accessed October 30, 2025, https://in.tradingview.com/scripts/butterworth/
18. Boosting: Why you Can Use the HP Filter - Cowles Foundation for Research in Economics, accessed October 30, 2025, https://cowles.yale.edu/sites/default/files/2022-08/d2212.pdf
19. Hodrick–Prescott filter - Wikipedia, accessed October 30, 2025, https://en.wikipedia.org/wiki/Hodrick%E2%80%93Prescott_filter
20. Hodrick-Prescott filter (HP) - Indicators - ProRealTime, accessed October 30, 2025, https://www.prorealcode.com/prorealtime-indicators/hodrick-prescott-filter-hp/
21. Denoising Autoencoders. A Denoising Autoencoder is a simply ..., accessed October 30, 2025, https://medium.com/@harishr2301/denoising-autoencoders-996e866e5cd0
22. Strong denoising of financial time-series - OpenReview, accessed October 30, 2025, https://openreview.net/forum?id=GG80jy9KI5
23. GRU Neural Network Based on CEEMDAN–Wavelet for Stock Price Prediction - MDPI, accessed October 30, 2025, https://www.mdpi.com/2076-3417/13/12/7104
24. Encoder–Decoder Based LSTM and GRU Architectures for Stocks ..., accessed October 30, 2025, https://www.mdpi.com/1911-8074/17/5/200
25. (PDF) A hybrid LSTM-GRU model for stock price prediction, accessed October 30, 2025, https://www.researchgate.net/publication/393492015_A_hybrid_LSTM-GRU_model_for_stock_price_prediction
26. arxiv.org, accessed October 30, 2025, https://arxiv.org/html/2508.20206v1
27. Filter then Attend: Improving attention-based Time Series ... - arXiv, accessed October 30, 2025, https://arxiv.org/abs/2508.20206
28. (PDF) Filter then Attend: Improving attention-based Time Series ..., accessed October 30, 2025, https://www.researchgate.net/publication/395032924_Filter_then_Attend_Improving_attention-based_Time_Series_Forecasting_with_Spectral_Filtering
29. SDformer: Transformer with Spectral Filter and Dynamic ... - IJCAI, accessed October 30, 2025, https://www.ijcai.org/proceedings/2024/0629.pdf
30. On Computational Complexity Reduction Methods for Kalman Filter ..., accessed October 30, 2025, https://www.researchgate.net/publication/336344733_On_Computational_Complexity_Reduction_Methods_for_Kalman_Filter_Extensions
31. Hidden Markov model - Wikipedia, accessed October 30, 2025, https://en.wikipedia.org/wiki/Hidden_Markov_model
32. A Markov Regime Switching Approach to Characterizing Financial ..., accessed October 30, 2025, https://medium.com/@cemalozturk/a-markov-regime-switching-approach-to-characterizing-financial-time-series-a5226298f8e1
33. Market Regime Detection using Hidden Markov Models in QSTrader ..., accessed October 30, 2025,

https://www.quantstart.com/articles/market-regime-detection-using-hidden-markov-models-in-qstrader/
34. Stock Market Prediction Using Hidden Markov Models, accessed October 30, 2025, https://users.cs.duke.edu/~bdhingra/papers/stock_hmm.pdf
35. Introduction to Markov-Switching Models | Aptech, accessed October 30, 2025, https://www.aptech.com/blog/introduction-to-markov-switching-models/
36. Market Regimes Explained: Build Winning Trading Strategies, accessed October 30, 2025, https://www.luxalgo.com/blog/market-regimes-explained-build-winning-trading-strategies/
37. Step-by-Step Python Guide for Regime-Specific Trading Using HMM ..., accessed October 30, 2025, https://blog.quantinsti.com/regime-adaptive-trading-python/
38. Volatility Models: ARCH and GARCH | Intro to Time Series Class Notes - Fiveable, accessed October 30, 2025, https://fiveable.me/intro-time-series/unit-14
39. ARCH and GARCH: Exploring Volatility in Finance | by Okan ..., accessed October 30, 2025, https://faun.pub/arch-and-garch-exploring-volatility-in-finance-d6f0f279abde
40. medium.com, accessed October 30, 2025, https://medium.com/codex/can-entropy-measures-reveal-hidden-insights-in-time-series-data-e9113e0e44a5#:~:text=In%20the%20context%20of%20time,is%20more%20regular%20and%20predictable.
41. Variance of entropy for testing time-varying regimes ... - DiVA portal, accessed October 30, 2025, https://uu.diva-portal.org/smash/get/diva2:1824880/FULLTEXT01.pdf
42. Information Entropy and Measures of Market Risk - MDPI, accessed October 30, 2025, https://www.mdpi.com/1099-4300/19/5/226
43. Price predictability at ultra-high frequency: Entropy-based ..., accessed October 30, 2025, https://arxiv.org/pdf/2312.16637
44. Price predictability at ultra-high frequency: Entropy-based ..., accessed October 30, 2025, https://arxiv.org/abs/2312.16637
45. Financial time series analysis based on effective phase transfer ..., accessed October 30, 2025, https://ideas.repec.org/a/eee/phsmap/v468y2017icp398-408.html
46. Using transfer entropy to measure information flows ... - EconStor, accessed October 30, 2025, https://www.econstor.eu/bitstream/10419/79614/1/722172478.pdf
47. Probabilistic Graphical Models Specialization [3 courses] (Stanford ..., accessed October 30, 2025, https://www.coursera.org/specializations/probabilistic-graphical-models
48. Market Analysis and Trading Strategies with Bayesian Networks, accessed October 30, 2025, https://c4i.gmu.edu/~pcosta/F15/data/fileserver/file/472072/filename/Paper_1570113749.pdf
49. Probabilistic Graphical Models: A New Way of Thinking in Financial Modelling, accessed October 30, 2025,

https://www.researchgate.net/publication/280230530_Probabilistic_Graphical_Models_A_New_Way_of_Thinking_in_Financial_Modelling
50. Dynamic Bayesian Networks (DBNs) - GeeksforGeeks, accessed October 30, 2025, https://www.geeksforgeeks.org/artificial-intelligence/dynamic-bayesian-networks-dbns/
51. Dynamic Bayesian network modeling for longitudinal brain ..., accessed October 30, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC3254821/
52. arxiv.org, accessed October 30, 2025, https://arxiv.org/html/2403.12180v1
53. Factor Based Statistical Arbitrage in the U.S. Equity Market with a ..., accessed October 30, 2025, https://epublications.marquette.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1419&context=theses_open
54. (PDF) Dynamic modeling of mean-reverting spreads for statistical ..., accessed October 30, 2025, https://www.researchgate.net/publication/1747590_Dynamic_modeling_of_mean-reverting_spreads_for_statistical_arbitrage
55. Statistical Arbitrage: Asset clustering, market-exposure minimization, and high-frequency explorations., accessed October 30, 2025, http://stanford.edu/class/msande448/2016/final/group3.pdf
56. Momentum Trading for Beginners: The Complete Guide to Trading ..., accessed October 30, 2025, https://www.mindmathmoney.com/articles/momentum-trading-for-beginners-the-complete-guide-to-trading-strong-price-moves
57. Comprehensive Guide to Momentum Trading Strategies in 2024 ..., accessed October 30, 2025, https://bookmap.com/blog/exploring-momentum-trading-strategies-a-comprehensive-guide
58. optimal dynamic momentum strategies - Rady School of Management, accessed October 30, 2025, https://rady.ucsd.edu/_files/faculty-research/liu/Liu_Paper_MomentumStrategies.pdf
59. The Kelly Criterion and Its Application to Portfolio Management | by ..., accessed October 30, 2025, https://medium.com/@jatinnavani/the-kelly-criterion-and-its-application-to-portfolio-management-3490209df259
60. Hey Kelly, Optimize My Portfolio. What's the difference between an ..., accessed October 30, 2025, https://medium.com/@jlevi.nyc/hey-kelly-optimize-my-portfolio-64e835fbced9
61. Kelly vs. Markowitz Portfolio Optimization by Magnus Erik ... - GitHub, accessed October 30, 2025, https://raw.githubusercontent.com/scibrokes/kelly-criterion/d12b9b565b49f4b987470ca2cd853619f050594d/references/Kelly%20vs%20Markowitz%20Portfolio%20Optimization.pdf
62. A Survey of Deep Reinforcement Learning in ... - Atlantis Press, accessed October

30, 2025, https://www.atlantis-press.com/article/125999560.pdf
63. Developing A Multi-Agent and Self-Adaptive Framework with ... - arXiv, accessed October 30, 2025, https://arxiv.org/pdf/2402.00515
64. A Practical Breakdown of Vector-Based vs. Event-Based Backtesting, accessed October 30, 2025, https://www.interactivebrokers.com/campus/ibkr-quant-news/a-practical-breakdown-of-vector-based-vs-event-based-backtesting/
65. Vector Vs Event Based Backtesting | PDF | Day Trading | Simulation, accessed October 30, 2025, https://www.scribd.com/document/881304872/Vector-vs-Event-Based-Backtesting
66. Battle-Tested Backtesters: Comparing VectorBT, Zipline, and ..., accessed October 30, 2025, https://medium.com/@trading.dude/battle-tested-backtesters-comparing-vectorbt-zipline-and-backtrader-for-financial-strategy-dee33d33a9e0
67. Most Popular Python Backtesting Libraries - Krystian Safjan's Blog, accessed October 30, 2025, https://safjan.com/popular-backtesting-libraries/
68. Event-Driven Backtesting with Python - Part I | QuantStart, accessed October 30, 2025, https://www.quantstart.com/articles/Event-Driven-Backtesting-with-Python-Part-I/
69. That's it, I'm done with Zipline. I can't take it anymore. Does anyone ..., accessed October 30, 2025, https://www.reddit.com/r/algotrading/comments/q1hbhs/thats_it_im_done_with_zipline_i_cant_take_it/
70. Key Concepts - QuantConnect.com, accessed October 30, 2025, https://www.quantconnect.com/docs/v2/writing-algorithms/reality-modeling/slippage/key-concepts
71. Backtesting Limitations: Slippage and Liquidity Explained - LuxAlgo, accessed October 30, 2025, https://www.luxalgo.com/blog/backtesting-limitations-slippage-and-liquidity-explained/
72. Using Backtesting to Avoid Slippage in Equities Trading - Exegy, accessed October 30, 2025, https://www.exegy.com/avoiding-slippage-equities-trading-with-backtesting/
73. Understanding Algorithmic Trading and the Role of Latency, accessed October 30, 2025, https://www.sundancedsp.com/understanding-algorithmic-trading-and-the-critical-role-of-latency/
74. High Frequency/Low Latency : r/cpp_questions - Reddit, accessed October 30, 2025, https://www.reddit.com/r/cpp_questions/comments/1d9g16g/high_frequencylow_latency/
75. High-frequency trading - Wikipedia, accessed October 30, 2025, https://en.wikipedia.org/wiki/High-frequency_trading

76. Trading Latency Optimization Guide - TradersPost Blog, accessed October 30, 2025, https://blog.traderspost.io/article/trading-latency-optimization-guide
77. What is the role of cross-validation in time series analysis? - Milvus, accessed October 30, 2025, https://milvus.io/ai-quick-reference/what-is-the-role-of-crossvalidation-in-time-series-analysis
78. Time Series Cross-Validation - GeeksforGeeks, accessed October 30, 2025, https://www.geeksforgeeks.org/machine-learning/time-series-cross-validation/
79. How to Perform Cross-Validation in Time Series - Statology, accessed October 30, 2025, https://www.statology.org/how-to-perform-cross-validation-in-time-series/
80. Advances in Financial Machine Learning · Reasonable Deviations, accessed October 30, 2025, https://reasonabledeviations.com/notes/adv_fin_ml/
81. Guide to Quantitative Trading Strategies and Backtesting, accessed October 30, 2025, https://www.pyquantnews.com/free-python-resources/guide-to-quantitative-trading-strategies-and-backtesting
82. Reinforcement Learning Framework for Quantitative Trading - arXiv, accessed October 30, 2025, https://arxiv.org/html/2411.07585v1
83. Trading with Machine Learning - Backtesting.py, accessed October 30, 2025, https://kernc.github.io/backtesting.py/doc/examples/Trading%20with%20Machine%20Learning.html
84. Automated Crypto Trading with Freqtrade and NostalgiaForInfinity ..., accessed October 30, 2025, https://alexbobes.com/crypto/automated-crypto-trading-with-freqtrade-and-nostalgiaforinfinity/
85. AI Agent Architecture: Key Components & Best Practices - Kanerika, accessed October 30, 2025, https://kanerika.com/blogs/ai-agent-architecture/
86. Built a Personal AI Financial Analyst with Python and LangChain For ..., accessed October 30, 2025, https://vardhmanandroid2015.medium.com/built-a-personal-ai-financial-analyst-with-python-and-langchain-for-crypto-currency-commodities-a4601a6c3839
87. LangChain Trading: Stock Analysis and LLM-Based Equity Analysis ..., accessed October 30, 2025, https://blog.quantinsti.com/langchain-trading-stock-analysis-llm-financial-python/
88. Comparison of Open-Source AI Agent Frameworks: Choosing the ..., accessed October 30, 2025, https://deepfa.ir/en/blog/open-source-ai-agent-frameworks-comparison
89. Getting Started | AutoGen 0.2 - Microsoft Open Source, accessed October 30, 2025, https://microsoft.github.io/autogen/0.2/docs/Getting-Started/
90. microsoft/autogen: A programming framework for agentic AI - GitHub, accessed October 30, 2025, https://github.com/microsoft/autogen
91. AutoGen - Microsoft Research, accessed October 30, 2025, https://www.microsoft.com/en-us/research/project/autogen/

92. Autogen vs LangChain vs CrewAI | *instinctools, accessed October 30, 2025, https://www.instinctools.com/blog/autogen-vs-langchain-vs-crewai/
93. LangChain vs CrewAI vs AutoGen: Choosing the Right Agentic AI ..., accessed October 30, 2025, https://gnxtsystems.com/langchain-vs-crewai-vs-autogen-choosing-the-right-ai-agent-framework-for-your-business/
94. AI Agent Architecture Secrets That Beat the 40% Failure Rate | Galileo, accessed October 30, 2025, https://galileo.ai/blog/ai-agent-architecture
95. AI Agent Architecture: Frameworks, Patterns & Best Practices, accessed October 30, 2025, https://www.leanware.co/insights/ai-agent-architecture
96. Deep Learning in Quantitative Finance: Multiagent Reinforcement ..., accessed October 30, 2025, https://blogs.mathworks.com/finance/2024/05/17/deep-learning-in-quantitative-finance-multiagent-reinforcement-learning-for-financial-trading/
97. MAPS: Multi-Agent reinforcement learning-based Portfolio ... - IJCAI, accessed October 30, 2025, https://www.ijcai.org/proceedings/2020/0623.pdf
98. Foundations of Reinforcement Learning with ... - Stanford University, accessed October 30, 2025, https://stanford.edu/~ashlearn/RLForFinanceBook/book.pdf
99. Must read papers for Reinforcement Learning : r/reinforcementlearning, accessed October 30, 2025, https://www.reddit.com/r/reinforcementlearning/comments/1is773d/must_read_papers_for_reinforcement_learning/
100. Advances in Financial Machine Learning - agorism.dev, accessed October 30, 2025, https://agorism.dev/book/finance/ml/Marcos%20Lopez%20de%20Prado%20-%20Advances%20in%20Financial%20Machine%20Learning-Wiley%20%282018%29.pdf
101. Advances in Financial Machine Learning | Wiley, accessed October 30, 2025, https://www.wiley.com/en-us/Advances+in+Financial+Machine+Learning-p-9781119482086
102. Attention Factors for Statistical Arbitrage - arXiv, accessed October 30, 2025, https://arxiv.org/abs/2510.11616
103. The Evolution of Reinforcement Learning in Quantitative Finance: A Survey - arXiv, accessed October 30, 2025, https://arxiv.org/html/2408.10932v3
104. Lesson 1: Introduction to Quantitative Trading and Freqtrade - DEV ..., accessed October 30, 2025, https://dev.to/henry_lin_3ac6363747f45b4/lesson-1-introduction-to-quantitative-trading-and-freqtrade-4ho5
105. Home — PyMC project website, accessed October 30, 2025, https://www.pymc.io/
106. pgmpy/pgmpy: Python library for causal inference and ... - GitHub, accessed October 30, 2025, https://github.com/pgmpy/pgmpy
107. QuantConnect.com: Open Source Algorithmic Trading Platform., accessed October 30, 2025, https://www.quantconnect.com/
108. Top Backtesting Software Comparison for 2025 - ChartsWatcher, accessed

October 30, 2025,
https://chartswatcher.com/pages/blog/top-backtesting-software-comparison-for-2025