Agentic Quant Backtesting Copilot: Do Agentic Pipelines Beat a Plain LLM for Quant Research?

Team: *JoJo Wang & whymath@uchicago.edu* | **Course Project One-Page Proposal**

Motivation & Research Question

Quant backtesting with Single-shot code generated by large language models (LLMs) is vulnerable to data leakage, timestamp misalignment, fragile code, and insufficient documentation. Do multi-agent pipelines improve *correctness*, *robustness*, and *productivity* for quantitative backtesting versus a plain LLM without tools or iteration?

Hypothesis

An multi-agent pipeline with **data or tools retrieval**, **code verification/repair**, and **execution loops** will yield (i) higher task-success and spec-compliance, (ii) lower error rates, and (iii) tighter dispersion of economic metrics (e.g., Sharpe) than a plain LLM baseline.

Agents, Data, and Tools

Agents & Roles (implemented with LangGraph):

- **Orchestrator:** plans/routes; enforces attempt budget; logs state.
- **Spec & Guard:** parses NL prompt; checks some important elements like dates/universe/frequency.
- **Retriever:** fetches vetted utilities .
- **Coder:** emits runnable module that calls utilities and a strategy registry.
- **Code Verification:** static checks the code prior to execution.
- **Runner:** executes on frozen data snapshot; computes metrics/artifacts.
- **Test-Result Verifier:** unit + economic checks (finite metrics, turnover bounds.....).
- **Code Fixing:** targeted patches from logs/check failures; deterministic seed increment.
- **Reporter:** Markdown summaries (metrics, diagnostics) or failure reports (reasons + logs).

Data & Tools:

- **Data:** Local data + running time data streaming from open API.
- **Tools: streaming data from open API; some metrics computation tools, or some feature engineering functions**

Task Suite

Candidate strategies: momentum (daily/weekly), mean-reversion, breakout, pair trading, volatility targeting, toy risk-parity, ATR stop/TP bandit, weekday mask, regime filter (MA cross).....

Baseline & Experimental Design

Conditions:

1. **Agentic System (ours)** with retrieval + verification/repair.
2. **LLM-Only Baseline:** single-shot code/report generation; *no tools*, *no retrieval*, *no repair loops*.

Success Criteria, Risks & Mitigations

Success: Agentic system improves Task-Success and Error Rate.
Risks and Mitigations: LLM fragility, runtime variance. we will use strict time-index tests and feature-window audits; low-temperature decoding + capped repair budgets(max loop).

Reflection & Contribution

We expect a clear, reproducible demonstration of *when* and *how* agentic pipelines provide measurable gains over plain LLM codegen in a real, scientific-style workflow (quant backtesting), aligning with the course's emphasis on careful and fair evaluation.