# Reliable Data Science Analysis with Large Language Models via Multi-Agent Tools Orchestration

Extended Abstract

Mingyin Zou
Strategic Assessments and Consultation Institute, Academy of Military Science
Beijing, China
zoumingyin20@alumni.nudt.edu.cn

Guangrong You
Strategic Assessments and Consultation Institute, Academy of Military Science
Beijing, China
13910742660@163.com

Xiaomin Zhu
Strategic Assessments and Consultation Institute, Academy of Military Science
Beijing, China
xmzhu@nudt.edu.cn

Yanqing Ye
Strategic Assessments and Consultation Institute, Academy of Military Science
Beijing, China
yeyanqing09@alumni.nudt.edu.cn

Ji Wang
Laboratory for Big Data and Decision, National University of Defense Technology
Changsha, China
wangji@nudt.edu.cn

## ABSTRACT

While Large Language Models (LLMs) show promise for automating the labor-intensive process of data science analysis, their practical application is undermined by the generation of erroneous and unreliable code. We argue that this stems from treating LLMs as open-ended code generators—a task akin to answering an essay question. We propose a fundamental paradigm shift: our framework reframes the task as one of structured tool selection and parameterization, effectively turning the "essay question" into a sequence of "multiple-choice and fill-in-the-blanks" problems. This shift dramatically reduces the potential for error. Our contributions are twofold. First, a multi-agent framework orchestrates the workflow, breaking down complex tasks into verifiable steps. Second, we construct an auto-generated tool library that supports a novel mechanism, empowering LLMs to select tools based on full source code rather than descriptions.

## KEYWORDS

LLM, Code Generation, Multi-Agent, Data Science

**ACM Reference Format:**
Mingyin Zou, Guangrong You, Xiaomin Zhu, Yanqing Ye, and Ji Wang. 2026. Reliable Data Science Analysis with Large Language Models via Multi-Agent Tools Orchestration: Extended Abstract. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 3 pages. https://doi.org/10.65109/RPMN5314

## 1 INTRODUCTION

Data science analysis is pivotal for transforming raw data into knowledge, but it entails a complex, interdependent workflow [2, 10]. While Large Language Models (LLMs) promise to democratize this process, early code-generation approaches lacked reliability [6, 8, 9]. Subsequent multi-turn paradigms improved flexibility but still struggled with maintaining context [7, 12]. The current frontier has shifted to multi-agent systems such as AgentCoder [5] and AutoGen [11], which decompose tasks into specialized roles.

Despite this evolution, a fundamental vulnerability persists: an over-reliance on LLMs' fallible generative capabilities. LLMs are prone to "hallucinations" [1] and may lack domain-specific knowledge [6], rendering even sophisticated multi-agent systems fragile when they rely on unverified reasoning.

To address these challenges, we propose **MATO** (reliable data science analysis with **M**ulti-**A**gent **T**ools **O**rchestration), a framework that recasts LLMs from fallible code generators into reliable orchestrators of verified tools. (1) To improve reliability, we shift the core task from unconstrained code synthesis to structured tool invocation, where agents resolve subtasks by selecting functions from a verified library. (2) To bridge the domain-knowledge gap, we build a robust tool library via a self-expanding bootstrapping process and introduce a code-aware invocation mechanism. By including source code in the tool schema, agents can "read" the implementation to ensure accurate tool usage.

## 2 METHOD

We propose MATO, a framework that orchestrates four specialized agents to transform natural-language queries into reliable analysis results (see Figure 1). The workflow maintains a state $S_t = (D_t, V_t, C_t)$ (data, variables, and code) and uses a curated tool library $\mathcal{L}_T$.

**Dynamic Tool Library Curation.** To ensure broad domain coverage, we construct $\mathcal{L}_T$ around 16 fundamental data science task categories. We employ a bootstrapping process in which an LLM generates problems for each category and decomposes them into subtasks. For a subtask $s$, if an existing function $f \in \mathcal{L}_T$ matches under semantic similarity ($\text{Sim}(s, f) \geq \tau$), it is reused; otherwise, a new function is generated or adapted. All additions undergo



*Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus.* © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/RPMN5314

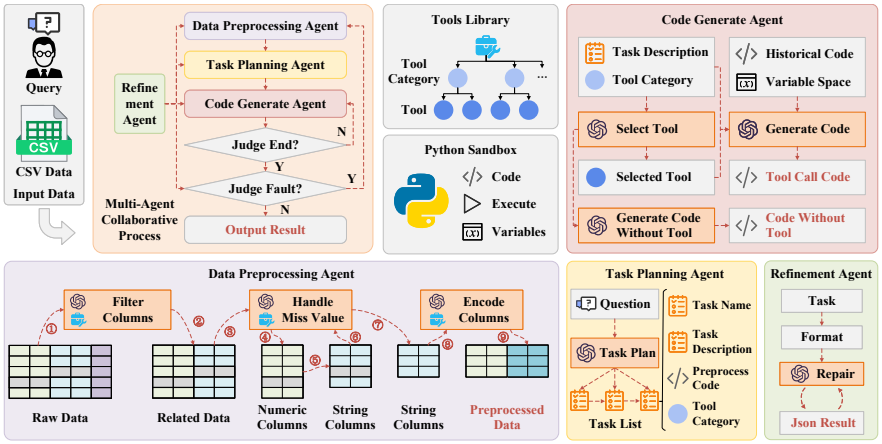


**Figure 1: The workflow of MATO. The process begins when a user submits a query and a CSV file. First, a Data Preprocessing Agent cleans the data by handling missing values and performing the required encoding. Next, a Task Planning Agent analyzes the query and produces a task plan. For each task, a Code Generate Agent produces code by selecting and invoking appropriate tools from a predefined library. The generated code is then validated for correctness and safety in a secure Python sandbox. If any preceding agent produces an incorrect output, a dedicated Refinement Agent intervenes to correct it, ensuring that the final result accurately addresses the user's request.**

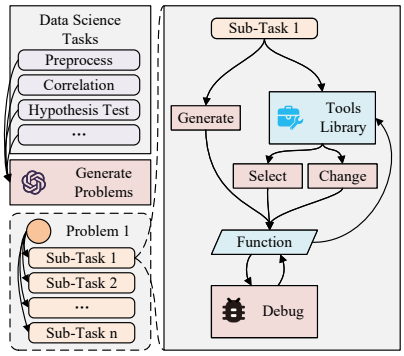


**Figure 2: The framework of dynamic Tool Library curation.**

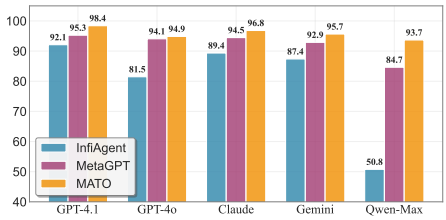


**Figure 3: Average ACC score under different frameworks.**

rigorous validation, resulting in 223 distinct, verified functions (Figure 2).

**Multi-Agent Workflow.** The process proceeds sequentially through four agents: **Data Preprocessing Agent ($\mathcal{A}_{DP}$):** $\mathcal{A}_{DP}$ cleans raw data $D_0$ using a plan-then-execute approach. It first generates a remediation plan $P_{DP}$ based on data metadata (Schema) and then iteratively generates and executes code to produce processed data $D_0'$ and code history $C_{DP}$. **Task Planning Agent ($\mathcal{A}_{TP}$):** $\mathcal{A}_{TP}$ decomposes the user query $Q$ into a sequence of executable sub-tasks $P = \langle p_1, \ldots, p_K \rangle$. Each task $p_k$ is assigned a specific category from $\mathcal{L}_T$, creating an abstract execution plan. **Code Generate Agent ($\mathcal{A}_{CG}$):** For each task $p_k$, $\mathcal{A}_{CG}$ selects the optimal tool $l^* \in \mathcal{L}_T$ based on semantic similarity. Uniquely, it employs a *code-aware invocation mechanism*: it reads the tool's source code to ensure correct parameterization in the current context $V_{k-1}$. If no tool matches (Sim $< \theta$), it falls back to generating code from scratch. This hybrid strategy balances reliability with flexibility. **Refinement Agent ($\mathcal{A}_R$):** $\mathcal{A}_R$ acts as a robustness module. If any agent's output violates the required schema or causes execution errors, $\mathcal{A}_R$ is triggered to rectify the output using the error context $O_{err}$ and schema constraints.

## 3 EMPERICAL ANALYSIS

**Setup.** We evaluate MATO on a curated version of the InfiAgent-DABench dataset [4], which we rectified for mathematical consistency across six core data science tasks (e.g., summary statistics and feature engineering). To demonstrate generalizability, we benchmark MATO against two baselines—InfiAgent (prompt-based) [4] and MetaGPT (multi-agent) [3]—across five state-of-the-art LLMs: GPT-4.1, GPT-4o, Claude-sonnet-4, Gemini-2.5-pro, and Qwen-max. Performance is assessed using execution accuracy (ACC).

**Results.** As shown in Figure 3, MATO consistently outperforms the baselines across all models and tasks. With GPT-4.1, MATO achieves a peak ACC of 98.43%, surpassing MetaGPT (95.28%) and InfiAgent (92.13%). Crucially, MATO demonstrates exceptional robustness with less capable models; for Qwen, it elevates accuracy to 93.7%, a dramatic improvement over InfiAgent (50.79%) and MetaGPT (84.65%). These results underscore MATO's ability to effectively structure and enhance the reasoning capabilities of diverse LLMs.

## REFERENCES


[1] Shihan Dou, Haoxiang Jia, Shenxi Wu, Huiyuan Zheng, Weikang Zhou, Muling Wu, Mingxu Chai, Jessica Fan, Caishuang Huang, Yunbo Tao, Yan Liu, Enyu Zhou, Ming Zhang, Yuhao Zhou, Yueming Wu, Rui Zheng, Ming Wen, Rongxiang Weng, Jingang Wang, Xunliang Cai, Tao Gui, Xipeng Qiu, Qi Zhang, and Xuanjing Huang. 2024. What's Wrong with Your Code Generated by Large Language Models? An Extensive Study. arXiv:2407.06153 [cs.SE] https://arxiv.org/abs/2407.06153

[2] Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Chenxing Wei, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Xiangru Tang, Xiangtao Lu, Xiawu Zheng, Xinbing Liang, Yaying Fei, Yuheng Cheng, Zhibin Gou, Zongze Xu, and Chenglin Wu. 2024. Data Interpreter: An LLM Agent For Data Science. arXiv:2402.18679 [cs.AI] https://arxiv.org/abs/2402.18679

[3] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: The Multi-Agent Framework. https://github.com/FoundationAgents/MetaGPT.

[4] Xueyu Hu, Ziyu Zhao, Shuang Wei, Ziwei Chai, Qianli Ma, Guoyin Wang, Xuwu Wang, Jing Su, Jingjing Xu, Ming Zhu, Yao Cheng, Jianbo Yuan, Jiwei Li, Kun Kuang, Yang Yang, Hongxia Yang, and Fei Wu. 2024. InfiAgent-DABench: Evaluating Agents on Data Analysis Tasks. arXiv:2401.05507 [cs.CL] https://arxiv.org/abs/2401.05507

[5] Dong Huang, Jie M. Zhang, Michael Luck, Qingwen Bu, Yuhao Qing, and Heming Cui. 2024. AgentCoder: Multi-Agent-based Code Generation with Iterative Testing and Optimisation. arXiv:2312.13010 [cs.CL] https://arxiv.org/abs/2312.13010

[6] Haolin Jin, Huaming Chen, Qinghua Lu, and Liming Zhu. 2025. Towards Advancing Code Generation with Large Language Models: A Research Roadmap. arXiv:2501.11354 [cs.SE] https://arxiv.org/abs/2501.11354

[7] Yubo Li, Xiaobin Shen, Xinyu Yao, Xueying Ding, Yidi Miao, Ramayya Krishnan, and Rema Padman. 2025. Beyond Single-Turn: A Survey on Multi-Turn Interactions with Large Language Models. arXiv:2504.04717 [cs.CL] https://arxiv.org/abs/2504.04717

[8] Nathalia Nascimento, Everton Guimaraes, Sai Sanjna Chintakunta, and Santhosh Anitha Boominathan. 2024. LLM4DS: Evaluating Large Language Models for Data Science Code Generation. arXiv:2411.11908 [cs.SE] https://arxiv.org/abs/2411.11908

[9] Mohamed Nejjar, Luca Zacharias, Fabian Stiehle, and Ingo Weber. 2024. LLMs for Science: Usage for Code Generation and Data Analysis. arXiv:2311.16733 [cs.SE] https://arxiv.org/abs/2311.16733

[10] Maojun Sun, Ruijian Han, Binyan Jiang, Houduo Qi, Defeng Sun, Yancheng Yuan, and Jian Huang. 2024. A Survey on Large Language Model-based Agents for Statistics and Data Science. arXiv:2412.14222 [cs.AI] https://arxiv.org/abs/2412.14222

[11] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. arXiv:2308.08155 [cs.AI]

[12] Chen Zhang, Xinyi Dai, Yaxiong Wu, Qu Yang, Yasheng Wang, Ruiming Tang, and Yong Liu. 2025. A Survey on Multi-Turn Interaction Capabilities of Large Language Models. arXiv:2501.09959 [cs.CL] https://arxiv.org/abs/2501.09959