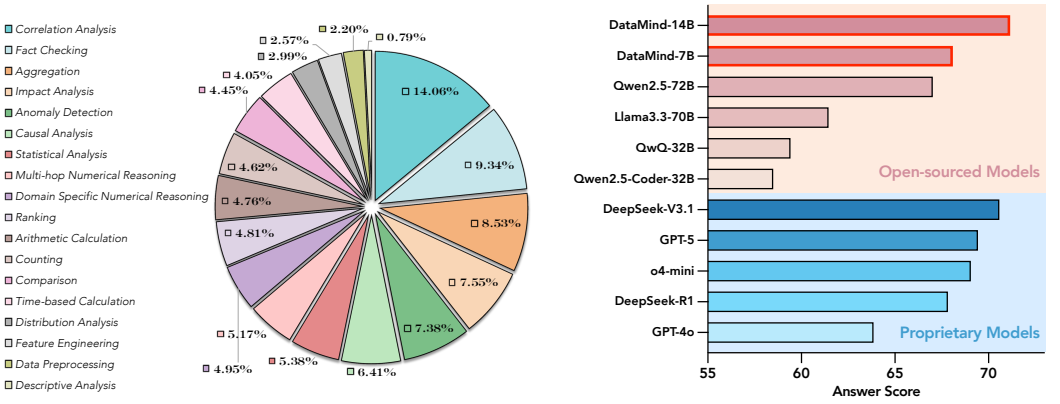
SCALING GENERALIST DATA-ANALYTIC AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Data-analytic agents are emerging as a key catalyst for automated scientific discovery and for the vision of Innovating AI. Current approaches, however, rely heavily on prompt engineering or multi-agent scaffolds over proprietary models, while open-source models still struggle with diverse-format, large-scale data files and long-horizon, multi-step reasoning that real-world analytics demands. This paper introduces DATAMIND, a scalable data synthesis and agent training recipe designed to construct generalist data-analytic agents. DATAMIND tackles three key challenges in building open-source data-analytic agents, including insufficient data resources, improper training strategy, and unstable code-based multi-turn rollout. Concretely, DATAMIND applies 1) a fine-grained task taxonomy and a recursive easy-to-hard task composition mechanism to increase the diversity and difficulty of synthesized queries; 2) a knowledge-augmented trajectory sampling strategy followed by model-based and rule-based filtering; 3) a dynamically adjustable training objective combining both SFT and RL losses; 4) a memory-frugal and stable code-based multi-turn rollout framework. Built on DATAMIND, we curate DATAMIND-12K, a high-quality trajectory set spanning diverse domains, task categories, and data file formats for data-analytic tasks. Trained on DATAMIND-12K, our DATAMIND-14B achieves state-of-the-art with an average score of 71.16% on multiple data analysis benchmarks, outperforming the strongest proprietary baselines DeepSeek-V3.1 and GPT-5. Our DATAMIND-7B also performs best among all open-source models with a score of 68.10%. We also list some empirical insights gained from our exploratory trials in the analysis experiments, aiming to provide actionable insights about agent training for the community. We will release DATAMIND-12K and DATAMIND-7B,14B for the community's future research.



(a) Task taxonomy used in DATAMIND for fine-grained and diverse query synthesis.

(b) Performance comparison between proprietary models and open-source models on multiple datasets.

Figure 1: **(a) Task Taxonomy.** We categorize data analysis tasks into 18 fine-grained categories to enhance the diversity of our synthesized queries. **(b) Performance Comparison.** Our DATAMIND-14B achieves the best compared with all proprietary models and open-source trained or untrained models.

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated formidable performance on a wide spectrum of reasoning tasks spanning math, code, and science (DeepSeek-AI et al., 2025; Kimi et al., 2025;

OpenAI, 2025a; Yang et al., 2025). As AI enters its *second half* (Yao, 2025), a surge of LLM Agentic benchmarks targeted in increasingly complex and domain-specific scenarios (Jimenez et al., 2024; Starace et al., 2025; Mialon et al., 2024; Phan et al., 2025; Wei et al., 2025a) is emerging. Among them, Automated Data Analysis (Hu et al., 2024; Jing et al., 2025; Liu et al., 2024; Majumder et al., 2025), an essential pillar of AI for scientific research, plays a critical role in realizing Innovating AI and has shown its promise to boost research efficiency and accelerate scientific discovery (Chen et al., 2025b; Schmidgall et al., 2025; Lu et al., 2024; Chai et al., 2025).

Data-Analytic Agents process, model, and compute data by generating code to discover useful information or regular conclusions, thereby furnishing users with insights to support decision-making. However, existing data-analytic agents (Zhang et al., 2023; Hong et al., 2025; Li et al., 2024; Sun et al., 2025; Guo et al., 2024) are overwhelmingly built on proprietary models via prompt engineering and rely on predefined workflows or multi-agent scaffolds. The few open-source trained models (Wu et al., 2025b;c; Su et al., 2024) can only perform simple table understanding tasks (tables compact enough to fit into the prompt) and can easily break down when confronted with diverse-format, large-scale data files and long-horizon, multi-step reasoning demanded by real-world tasks.

Challenges. In this work, we propose to train a generalist, open-source data-analytic agent. This endeavor entails several intrinsic challenges that must be addressed: 1) *Insufficient data resources.* Training a specialized agent demands a large-scale, high-quality collection of tasks and corresponding solution trajectories. However, publicly available data analysis benchmarks often only provide a limited test set for evaluation purposes and lack step-by-step trajectory annotations, making it infeasible to assemble an effective training corpus from off-the-shelf resources. 2) *Improper training strategy.* Current agent training strategies typically follow an SFT-then-RL paradigm. Yet, in a new scenario, it remains unclear how to stabilize long-horizon agent training and how to allocate training steps across SFT and RL to achieve optimal performance. 3) *Unstable code-based multi-turn rollout.* Data files and code interpreters involve intricate memory management. Parallel agentic rollout and multi-turn code generation with limited memory resources will further exacerbate this situation.

The DATAMIND Pipeline. In response to the above challenges, we introduce DATAMIND, a scalable data synthesis and agent training recipe designed to build generalist data-analytic agents. To construct a large-scale training corpus, we begin by harvesting a diverse collection of data files in various formats and domains from the Internet and open communities. Then, we apply a fine-grained task taxonomy (see Fig.1a) and a recursive easy-to-hard task composition mechanism to increase the diversity and difficulty of our synthesized queries. Next, we adopt a knowledge-augmented trajectory sampling strategy to improve both the validity and reliability of synthesized trajectories. A model-based judger performs self-consistency filtering on these trajectories, followed by rule-based checks. The judgment signal will also be fed back to the model to encourage refinement, enriching the thinking patterns present in the final training set. During training, we combine SFT loss and RL loss with a dynamic coefficient to schedule the relative weight of SFT versus RL across training steps, allowing us to balance exploitation and exploration to stabilize training. For parallel multi-turn rollout, we asynchronize agent generation and code execution and utilize a chunk-wise code maintenance method to reduce peak memory usage. Moreover, we sandbox each trajectory in an isolated environment with strict caps on execution time and memory usage, enabling stable code-based multi-turn rollout.

Results and Insights. Through the DATAMIND pipeline, we curate **DATAMIND-12K**, a high-quality training set that spans diverse task categories and data file formats for data-analytic tasks. When trained on DATAMIND-12K, our 14B model, **DATAMIND-14B**, achieves a new state-of-the-art with an average score of 71.16% on multiple data analysis benchmarks, outperforming the strongest proprietary baselines DeepSeek-V3.1 and GPT-5 and surpassing all open-source models by a substantial margin (see Fig.1b). Our **DATAMIND-7B** also performs best among all open-source models with a score of 68.10%. Our additional analysis studies yield two valuable insights for the community: 1) Self-consistency filtering is more non-trivial than the best trajectory selection; 2) SFT loss can be an effective stabilizer for RL training, but can also be the culprit of unstable training. 3) RL can narrow the performance gap between different base models, but can hardly reverse the order.

2 PROBLEM DEFINITION

A data analysis task u is typically represented as a quadruple $u = (q, f, d, a)$, comprising the user query q, the data file f, the data description d, and the answer a, where data file f may be provided in a variety of formats (`.csv`, `.xlsx`, `.sqlite`, etc.), and data description d is optional.

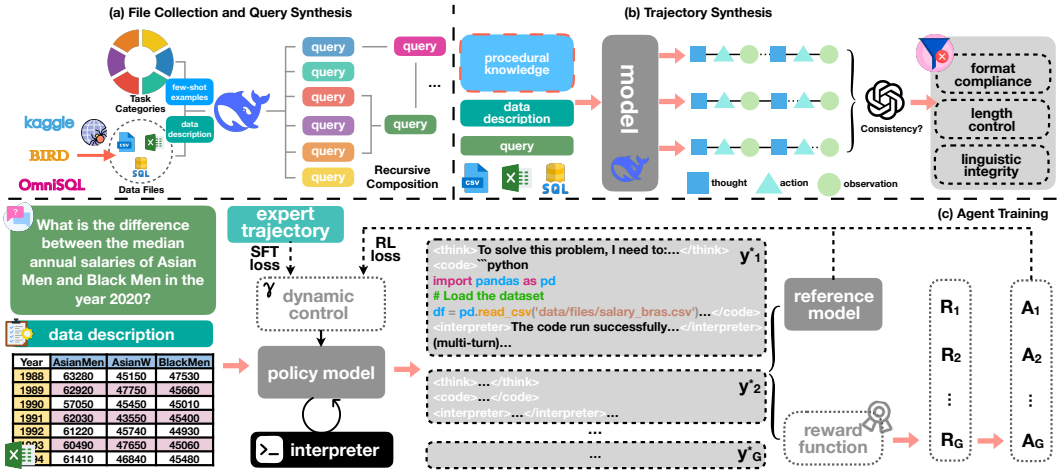


Figure 2: **The Pipeline of DATAMIND.** DATAMIND applies 1) a fine-grained task taxonomy and a recursive easy-to-hard task composition mechanism; 2) a knowledge-augmented trajectory sampling strategy followed by model-based and rule-based filtering; 3) a dynamically adjustable training objective including both SFT and RL losses; 4) a memory-frugal and stable code-based multi-turn rollout framework.

Our agent framework adheres to the prevailing `ReAct` (Yao et al., 2023) paradigm. Upon receiving a task, the agent is required to iterate multiple rounds of `Thought-Action-Observation` cycles and finally produce an answer. In the data analysis scenario, `Thought` denotes the agent's reasoning and reflection process conditioned on the current context; `Action` refers to the agent's invocation of code to process and compute over the data files or the generation of the final answer. The code may be written in Python or SQL, depending on the data file format; `Observation` consists of the execution feedback returned by the environment (i.e., Code Interpreter).

Given task u, let a `Thought-Action-Observation` loop be represented by (τ, α, o), respectively. Then the agent's historical trajectory h at time step t can be denoted as:

$$h_t = (u, \tau_0, \alpha_0, o_0, \tau_1, \alpha_1, o_1, \ldots, \tau_{t-1}, \alpha_{t-1}, o_{t-1}). \tag{1}$$

Conditioned on the history trajectory h_t, the agent with parameters θ will produce its next thought τ_t and action α_t according to the policy $\pi_\theta(\tau_t, \alpha_t | h_t)$ and will receive an observation o_t from the code interpreter after action α_t is executed. The whole trajectory terminates either when the agent emits an answer or when a predefined maximum number of rounds $\mathcal{T}$ is reached. **For simplicity, in the following sections**, we denote the input part provided to the agent (including q, f, and d) as x and the trajectory (including answer a) sampled from the agent as $y \sim \pi_\theta(\cdot|x)$.

3 SCALING DATA-ANALYTIC AGENT DATA

3.1 FILE COLLECTION AND QUERY SYNTHESIS

Data File Collection. First, we need a large amount of raw data files f to scale up the potential synthesized task volume. Fortunately, the Internet and the open community benchmarks already host a massive reservoir of such files. We first target Kaggle, which contains tens of thousands of `.csv` and `.xlsx` spreadsheets. Using the official Kaggle API[1], we crawl a diverse subset of files spanning multiple domains, and then discard files that *i)* can not be loaded, *ii)* are extremely small (< 20 rows) or large ($> 1,000$ rows), or *iii)* contain anomalous data types. After this pipeline, we retain $3,400$ `.csv` and 560 `.xlsx` files. For database files, we draw primarily from the training set of BIRD (Li et al., 2023b) and OmniSQL (Li et al., 2025a), both of which are high-quality corpora widely used in the Text-to-SQL field. Similarly, we sample from these sources and apply an analogous filtering pipeline, finally obtaining $1,954$ `.sqlite` files.

Query Categorization and Synthesis. To generate specific queries, we devise an automated script to extract meta-information d of each data file, such as table headers, column names, data types, and representative rows, and then feed these metadata into DeepSeek-V3 (DeepSeek-AI, 2024) to synthesize queries q. To ensure both **diversity** and **fine-grainedness** of the generated questions, we refer to and refine the taxonomy in Wu et al. (2025b) and classify the data analysis tasks into 18 fine-grained categories (see Fig.1a). For each category, we carefully curate $4 \sim 6$ exemplar queries

[1] https://www.kaggle.com/docs/api.

that vary in complexity and domains and serve as few-shot demonstrations. Under the guidance of these type-specific contexts, every data file is used to generate a diverse set of queries that span the full spectrum of the proposed taxonomy. To further elevate query **complexity**, we adopt a recursive easy-to-hard composition scheme that chains multiple task types, i.e., the output of one task is fed as input to the next. By iterating $2 \sim 5$ times, we progressively amplify the difficulty and create multi-hop analytic challenges that go well beyond the capability required by any single task type. The prompts for query synthesis can be found in Appx.G.2.

3.2 Expert Trajectory Sampling and Filtering.

Knowledge Augmented Trajectory Sampling. To guarantee the quality of the synthesized trajectories, we introduce a knowledge-augmented trajectory sampling framework. Initially, for each question category, we manually craft a high-level workflow k that encodes procedural knowledge and steers the model during trajectory synthesis. To further boost answer quality, we impose a self-consistency filter. We sample $\mathcal{N}$ independent trajectories per query and employ a judge model $\mathcal{M}$ powered by GPT-4o-mini (OpenAI, 2024b) to verify whether their final answers are consistent with reasoning rationales. Only trajectories that converge to the same answer are retained; among them, the judge model will also select the most concise and accurate one as our training instance y:

$$\{c, s, y\} = \mathcal{M}(\{y^i\}_{i=1}^{\mathcal{N}}), \quad \{y^i\}_{i=1}^{\mathcal{N}} \sim \pi_{\theta_{\text{expert}}}(\cdot|k, x), \quad y = \begin{cases} y^i \in \{y^i\}_{i=1}^{\mathcal{N}}, & s = 1 \\ \texttt{none}, & s = 0 \end{cases}, \tag{2}$$

where c is the chain-of-thought process of the judge model to reach the binary conclusion s of whether the sampled trajectories are consistent. We use DeepSeek-V3.1 (DeepSeek-AI, 2025) as our expert policy model $\pi_{\theta_{\text{expert}}}$. During implementation, we set $\mathcal{N} = 3$. The prompt used for trajectory sampling and the judge model $\mathcal{M}$ can be found in Appx.G.3 and Appx.G.4, respectively. We extract the final answer from the trajectory as the final synthesized answer a for the corresponding query q. However, this pipeline inherently biases us toward easier queries whose answers are more likely to coincide. To counteract this, we refine the high-level workflow knowledge k into more granular, step-by-step instructions for categories that exhibit low inter-trajectory consistency. Moreover, for trajectories that fail the consistency check, we feed the judge model's chain-of-thought back to the agent as external critique, prompting it to reflect and revise its reasoning path:

$$\{y^i_{\text{reflected}}\}_{i=1}^{\mathcal{N}} \sim \pi_{\theta_{\text{expert}}}(\cdot|k, x, \{y^i\}_{i=1}^{\mathcal{N}}, c), \quad \text{if } s = 0. \tag{3}$$

The reflected trajectories $\{y^i_{\text{reflected}}\}_{i=1}^{\mathcal{N}}$ will be fed into the judge model $\mathcal{M}$ again to conduct the consistency check and the trajectory selection in Eqn.2. This rescue loop not only salvages additional usable data but also enriches the diversity of thinking patterns embedded in the trajectory pool.

Rule-based Trajectory Filtering. In addition to discarding inconsistent trajectories, we apply three further rule-based filtering stages. **1) Format compliance**. We drop any trajectory that deviates from the `ReAct` format, ensuring that every remaining trajectory can be losslessly converted into our target training schema. **2) Length control**. We filter out trajectories whose final answer exceeds $1,024$ tokens, preventing the model from exploiting spurious hallucinations to artificially hit the correct string. **3) Linguistic integrity**. We remove trajectories containing garbled text or intermingled natural languages, eliminating samples that could destabilize the agent training. After the full filtering pipeline, we retain $11,707$ high-quality trajectories named as DataMind-12K.

4 Scaling Data-Analytic Agent Training

Dynamic Control Between SFT and RL. In this paper, we adopt a combined paradigm of Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL) for the agent training. Empirically, we observe that it is difficult to strike a balance between the two stages: the model needs to absorb sufficient knowledge from expert data during SFT, yet excessive imitation often rigidifies exploration during RL. Hence, following Zhang et al. (2025b), we employ a hybrid strategy that dynamically blends on-policy and off-policy learning, allowing the training procedure to flexibly trade off between exploitation of expert knowledge and continued exploration.

Given the training dataset $\mathcal{D}$, we express our SFT loss as:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\sum_{t=1}^{|y|} \mathbb{I}(y_t \notin o) \cdot \log \pi_\theta(y_t|x, y_{<t})\right], \tag{4}$$

where $\mathbb{I}(y_t \notin o)$ is an indicator function that masks out any tokens produced by the environment feedback, ensuring that the model is optimized only on the agent-generated portion of the trajectory. For RL, we use the Decoupled Clip and Dynamic Sampling Policy Optimization (DAPO) (Yu et al., 2025) algorithm, minimizing the following function:

$$\mathcal{L}_{\text{DAPO}}(\theta) = -\,\mathbb{E}_{(x,y)\sim\mathcal{D},\{y_i^*\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)} \left[\frac{1}{\sum_{i=1}^{G}|y_i^*|}\sum_{i=1}^{G}\sum_{t=1}^{|y_i^*|}\min\left(r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}\Big(r_{i,t}(\theta), 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}}\Big)\hat{A}_{i,t}\right)\right] \quad (5)$$
$$\text{s.t.} \quad 0 < \Big|\{y_i^* \mid \texttt{is\_equivalent}(y, y_i^*)\}\Big| < G,$$

where $\{y_i^*\}_{i=1}^{G}$ is a group of G trajectories sampled from the agent policy $\pi_{\theta_{\text{old}}}$ and y is the expert trajectory. Similar to SFT, any tokens emitted by the environment are discarded when computing the objective. $r_{i,t}(\theta)$ denotes the per-token importance-sampling ratio, and $\hat{A}_{i,t}$ is the advantage of the i-th response, obtained by normalizing the group-level rewards $\{R_i\}_{i=1}^{G}$:

$$r_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t}^* \mid x, y_{i,<t}^*)}{\pi_{\theta_{\text{old}}}(y_{i,t}^* \mid x, y_{i,<t}^*)}, \quad \hat{A}_{i,t} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^{G})}{\text{std}(\{R_i\}_{i=1}^{G})}. \quad (6)$$

The inequality in Eqn.5 serves as a filtering criterion that discards trajectories lacking optimization utility whose rewards are uniformly 0 or uniformly 1 to prevent spurious gradient updates.

Finally, unlike the conventional SFT-then-RL pipeline, we jointly optimize the agent by combining the SFT and RL objectives with a dynamically balanced weighting factor:

$$\mathcal{L}_{\text{Final}}(\theta) = \gamma\mathcal{L}_{\text{SFT}}(\theta) + (1-\gamma)\mathcal{L}_{\text{DAPO}}(\theta), \quad (7)$$

where $\gamma \in [0, 1]$ varies dynamically throughout training. In our implementation, γ is initialized to a large value so that the agent first acquires knowledge from expert data via the SFT loss, and is then annealed to a small value to encourage extensive exploration through RL. Please refer to §5.3 for our analysis of different γ settings. Importantly, for any trajectory that is filtered out by the inequality in Eqn.5, we will compute only the SFT loss. To increase the likelihood of producing eligible trajectories during the early stage of RL training, we perform a cold start using DATAMIND-12K before the process described above. We also analyze the effect of cold start for RL training in §5.3.

Void Turns Filtering. In multi-turn agentic training, the model can experience distributional drift due to external feedback and multi-turn compounding errors during multi-turn rollout, which will easily result in trajectory collapse, thereby destabilizing RL training (Xue et al., 2025; Baronio et al., 2025; Mai et al., 2025). We also observe this phenomenon in our experiments. To stabilize training, we directly mask out the entire loss contributed by trajectories that contain void turns. Here, a void turn is defined as an agentic loop that fails to produce a valid code snippet or answer.

Agentic Code-based Multi-turn Rollout. A stable environment plays a key role in stable on-policy RL training. In data-analytic agent training, massive concurrent file I/O and code execution can easily lead to environment crashes, especially with limited memory resources. To prevent this, we implement three optimizations: **1) Asynchronous interaction**. We asynchronize model generation and code execution for different data samples, which can decouple peak GPU and CPU memory demands and avoid simultaneous file I/O and code-execution spikes. **2) Chunked code maintenance**. We implement a light-weight, notebook-style code generation strategy. The model only needs to produce the code snippet required for the current reasoning step, effectively reducing generation latency. Furthermore, whereas conventional notebook systems maintain a global variable pool, which is memory-intensive, we retain only the textual code chunks. At runtime, we concatenate the active snippet with its predecessors, yielding the same global execution effect without the memory overhead. **3) Security Control**. To ensure secure code execution, we isolate the runtime environment for each trajectory, enforce per-trajectory limits on CPU time and peak memory, and filter any snippet containing insecure function calls before execution. Additionally, we provide an **automatic package-installation** mechanism that dynamically checks and installs uninstalled Python packages.

Reward Design. Our reward mainly comprises three components: **format reward** $r_{\textbf{format}}$, **answer reward** $r_{\textbf{answer}}$, and **length reward** $r_{\textbf{length}}$. The agent is required to enclose its reasoning process

Table 1: **Main Results.** ♣ indicates that the original paper does not report results for the corresponding model and we use their official data and code to train the model for reproduction. ‡ denotes that we directly download their official trained model for fair evaluation. The best results for each model group are highlighted in **bold**.

Backbone	Method	DABench		TableBench		BIRD		Avg.	
		pass@1	pass@3	pass@1	pass@3	pass@1	pass@3	pass@1	pass@3
Proprietary Models									
GPT-4o	ReAct	76.39	84.44	64.97	75.06	50.20	62.39	63.85	73.96
o4-mini	ReAct	79.12	86.77	71.03	80.15	57.04	66.88	69.06	77.93
DeepSeek-R1	ReAct	78.73	87.55	68.96	79.52	55.80	66.17	67.83	77.75
DeepSeek-V3.1	ReAct	**81.32**	**89.49**	**72.52**	**81.68**	57.89	**68.12**	**70.58**	**79.76**
GPT-5	ReAct	78.21	85.21	69.93	78.37	**60.17**	65.19	69.44	76.26
Open-source Models									
Qwen-2.5-Coder-32B	ReAct	73.15	81.32	61.11	72.26	41.20	60.17	58.49	71.25
QwQ-32B	ReAct	70.17	85.21	57.79	75.19	50.30	64.21	59.42	74.87
Llama-3.3-70B	ReAct	69.78	80.16	55.47	70.36	59.10	68.58	61.45	73.03
Qwen-2.5-72B	ReAct	**75.33**	**86.38**	**65.44**	**76.21**	**60.30**	**69.49**	**67.02**	**77.36**
Qwen-2.5 Coder-7B	ReAct	15.05	35.41	11.70	28.63	7.02	18.71	11.26	27.58
	TableLLM♣	36.71	71.98	41.01	70.36	11.99	16.75	29.90	53.03
	Table-R1♣	42.54	78.99	56.36	63.61	10.69	13.49	36.53	52.03
	OmniSQL‡	26.46	36.19	39.95	50.25	57.11	66.30	41.17	50.91
	SQL-R1‡	24.90	34.63	40.84	50.64	56.78	66.23	40.83	50.50
	DATAMIND	**77.30**	**87.94**	**67.60**	**79.39**	**59.41**	**69.88**	**68.10**	**79.07**
Qwen-2.5 Coder-14B	ReAct	71.21	83.27	56.96	69.97	41.76	59.91	56.64	71.05
	TableLLM♣	38.26	74.71	46.44	76.08	20.99	28.88	35.23	59.89
	Table-R1♣	45.33	79.38	50.38	58.91	11.80	14.08	35.84	50.79
	OmniSQL‡	26.46	39.30	41.98	52.67	58.80	67.41	42.41	53.13
	SQL-R1‡	27.24	40.47	41.22	51.02	58.02	66.62	42.16	52.70
	DATAMIND	**80.29**	**88.72**	**70.95**	**81.81**	**62.23**	**70.21**	**71.16**	**80.25**

within **<think>**...**</think>** tags, place any generated data-processing code between **<code>** and **</code>**, and wrap its final answer in **<answer>**...**</answer>**. The environment's execution results will be placed between <interpreter> and </interpreter>. For the answer reward, as many answers are descriptive and thus resist rule-based verification, we adopt a *model-as-judge* powered by GPT-4o-mini (OpenAI, 2024b). We engineer a dedicated LLM evaluation prompt detailed in Appx. G.4. Both r_{format} and r_{answer} are binary with only 0 and 1. To mitigate the risk of the agent hacking the answer reward by hallucinating excessive tokens, we further impose a length-based penalty to discourage overly verbose outputs. We define the length reward and the final reward as:

$$R = \begin{cases} r_{\text{length}} \cdot r_{\text{answer}}, & r_{\text{answer}} = 1 \\ 0, & r_{\text{format}} = 1, r_{\text{answer}} = 0 \\ -0.1, & r_{\text{format}} = 0, r_{\text{answer}} = 0 \end{cases}, r_{\text{length}} = \begin{cases} 1, & l \le l_{\text{min}} \\ \frac{l_{\text{max}} - l}{l_{\text{max}} - l_{\text{min}}} \cdot 0.5 + 0.5, & l_{\text{min}} < l \le l_{\text{max}} \\ 0.5, & l_{\text{max}} < l \end{cases} \tag{8}$$

We incentivize correct outputs. So as long as the predicted answer exactly matches the ground truth, the model will receive a high reward (≥ 0.5). The specific value is length-dependent: we award a full reward if the answer length l is shorter than l_{min}; it decays linearly to 0.5 when the length falls between l_{min} and l_{max}; any sequence longer than l_{max} incurs a fixed length penalty of 0.5. According to our observation, we set l_{min} and l_{max} to 256 and 1024 respectively during our experiments.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETTINGS

Datasets and Metrics. We evaluate our model on three datasets related to data analysis: **DABench** (Hu et al., 2024), **TableBench** (Wu et al., 2025b), and **BIRD** (Li et al., 2023b). Our evaluation protocol aligns with our answer reward method, where a judge model powered by GPT-4o-mini (OpenAI, 2024b) is used to evaluate the correctness of the final answer. We report both pass@1 and pass@3 scores for all the methods. Please refer to Appx.D for more details.

Models and Baselines. We compare our models with five strong proprietary models and four outstanding open-source models (see Tab.1). In addition, we select four open-source models that have been explicitly trained for data-analysis-related tasks: **TableLLM** (Wu et al., 2025b), **Table-R1**

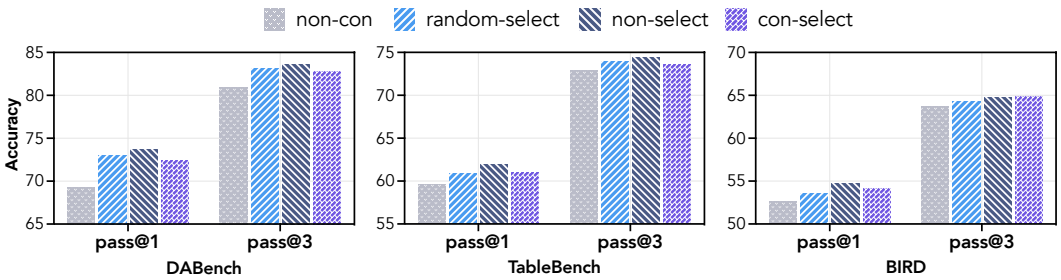


Figure 3: **Analysis on Self-Consistency Filtering and Best Trajectory Selection.** *Con-select* is our original setting, including self-consistency filtering and best trajectory selection by a judge model $\mathcal{M}$. *Non-select* uses all the sampled trajectories without the best selection. *Random-select* means randomly select a trajectory instead of the best selection. *Non-con* directly leverages all the synthesized trajectories without self-consistency filtering.

(Wu et al., 2025c), **OmniSQL** (Li et al., 2025a), and **SQL-R1** (Ma et al., 2025). We include **Qwen-2.5-Coder-7B** and **14B** (Hui et al., 2024) as our backbone models to compare different baselines. Detailed model information and reproduction protocols for all baselines are provided in Appx.E.

Training and Inference Setups. We use LlamaFactory (Zheng et al., 2024) for SFT training and verl (Sheng et al., 2025) for RL training. For SFT, our learning rate is $1e-5$ with a warmup ratio of 0.1 and a cosine decay schedule. Our global batch size is set to 16. For RL, we use a learning rate of $1e-6$. The batch size is 16 with a mini batch size of 2. The rollout temperature is 0.7, the top-p is 1.0, and the group size G is 4. We schedule γ via cosine decay, annealing from a peak of 0.9 to a valley of 0.05. At test time, we fix the temperature to 0.7, top-p to 0.95, and an inference batch size of 5 for all evaluations. The detailed hyperparameter information can be seen in Appx.F.

5.2 Main Results

As shown in Tab. 1, our 7B model, DataMind-7B, achieves the best among all open-source models with an average score of 68.10%. Our 14B model, DataMind-14B, attains an average score of 71.16% across all tasks, surpassing all proprietary models (including the latest GPT-5 and DeepSeek-v3.1) as well as all open-source alternatives. Moreover, our DataMind series models demonstrate robust mastery of diverse data formats and exhibit balanced performance across all datasets. By contrast, specialized models degrade sharply when confronted with unseen data. For example, OmniSQL-7B reaches 57.11% on BIRD, yet its performance on TableBench and DABench drops steeply. Note that to ensure a fair evaluation, we have converted all tables in these two benchmarks into `.sqlite` files. Nevertheless, SQL-oriented models still underperform. This observation indicates the breadth of query types and file formats covered by DataMind-12K. Furthermore, TableLLM and Table-R1 are limited to small-scale tables. When evaluated on DABench's large-scale tables, they fail to generalize, and their accuracy deteriorates even further on BIRD's multi-table analysis. These results highlight our model's capacity to handle complex tabular data, which can be attributed to the difficulty distribution embedded in DataMind-12K. Moreover, all trained baselines are exposed to significantly larger training corpora than ours (20K instances for TableLLM and Table-R1, and 2.5M for OmniSQL and SQL-R1, versus only 12K for DataMind), yet we outperform them even on their adept benchmarks. This gain is attributable to the high-quality reasoning trajectories curated in DataMind-12K and our stable training strategy. Our model also maintains a high pass@3 score, indicating that it preserves strong generation diversity while ensuring reliability.

5.3 Analysis

Self-consistency filtering is more non-trivial than the best trajectory selection. In Fig.3, we analyze the impact of the self-consistency trajectory filtering and best trajectory selection strategies through SFT on the 7B model. It is evident that removing the self-consistency filtering (*non-con*) inflicts the most pronounced degradation on model performance: both pass@1 and pass@3 drop to varying extents across all datasets. This observation suggests that the quality of the answers produced by a trajectory is a critical guarantee of the trajectory's overall quality. Provided that the final answers are consistent, we observe that randomly selecting a single trajectory for training is not necessarily worse than explicitly choosing the best one, and it even yields a clear improvement on DABench. We hypothesize that the judge model's preference bias may potentially reduce trajectory diversity. This conjecture can be further evidenced by the pass@3 scores of *random-select*, which are on par

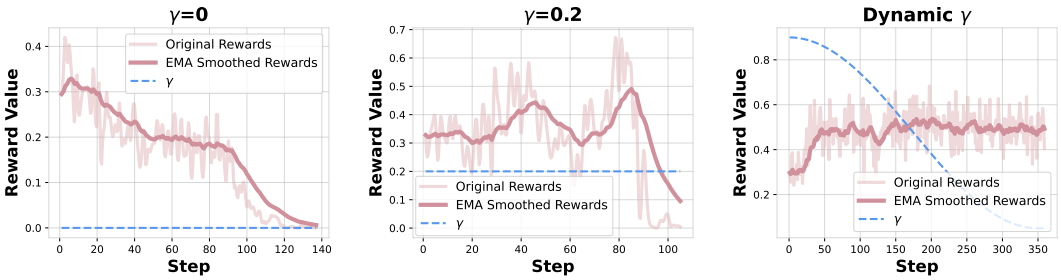


Figure 4: **The Influence of SFT Loss for RL Training.** $\gamma = 0$ denotes the absence of SFT loss, $\gamma = 0.2$ corresponds to a low SFT-loss weight, and dynamic γ indicates our naive setting.

with or superior to those of *con-select* across all three datasets. Moreover, the largest performance gains are obtained by including, without any selection, every trajectory that converges to a consistent answer. This pattern holds across all datasets and indicates that the diversity of reasoning patterns and problem-solving strategies embedded in the trajectories is more beneficial to the model's reasoning capability, which aligns with the findings in Guha et al. (2025), although we cannot fully rule out the contribution of the larger training volume introduced by this unfiltered approach.

SFT loss is an effective stabilizer for RL training. When our experiments are still in an exploratory phase, we use DATAMIND-12K to examine how the weight of the SFT loss in Eqn.7 influences the RL training on the 7B model without a cold start. In Fig.4, we plot the dynamics of the answer reward across training steps under different γ settings. As can be seen, when no SFT loss is imposed ($\gamma = 0$), the answer reward declines almost monotonically. We attribute this failure to two factors. First, the 7B model's limited multi-step reasoning capability makes it difficult to roll out high-quality trajectory groups for effective learning. Second, the heterogeneity of both data structures and code languages yields highly imbalanced trajectory distributions, resulting in unstable training. Raising γ to 0.2 can alleviate the problem to some extent. The answer reward initially rises despite large oscillations, yet the SFT loss remains too weak to prevent the policy from eventually drifting away and collapsing. Under our dynamic-γ schedule, the model first enjoys the stabilizing supervision of a strong SFT loss and after it matures, the SFT coefficient is gradually annealed to encourage exploration, yielding stable training during the whole process.

SFT loss can also be the culprit of unstable training. Although SFT loss serves as an effective stabilizer for RL training, we find that its persistent dominance throughout training can conversely trigger collapse. As shown in Fig.5, fixing γ at a high level also causes the answer reward to rise briefly, followed by a gradual decline. The underlying reason is that over-fitting to the SFT loss traps the policy in the rigid thinking patterns embedded in the expert trajectories, especially when these trajectories are synthesized from the same model, thereby crippling exploration. To corroborate this, we track the entropy of the policy during training and observe a pronounced entropy collapse phenomenon. In contrast, our dynamic-γ strategy can keep the policy entropy consistently at a relatively high level throughout training. Overall, we find the training process resembles raising a child. During early childhood, constant parental guidance (a large γ) is indispensable to keep the child from going astray. As the child grows up, excessive supervision stifles the child's innate drive for self-directed exploration. At that stage, judiciously letting go (a small γ) enables the child to discover their true capabilities through the feedback from the surrounding world.

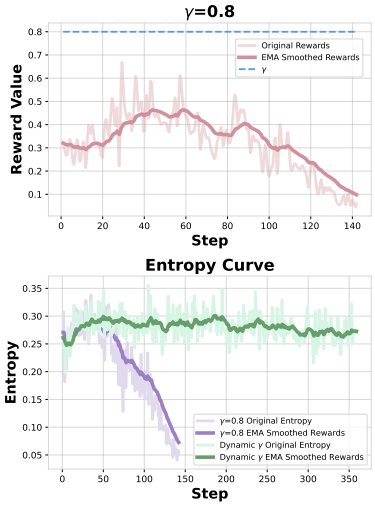


Figure 5: **Answer Reward and Entropy Dynamics** of different γ settings.

RL can narrow the performance gap between different base models, but can hardly reverse the order. Fig. 6 shows the impact of different degrees of cold start on the 7B model to subsequent RL training. For rapid empirical verification, we randomly sample $3,843$ training data from DATAMIND-12K (balanced on query types) and 240 test data (60 for each dataset) for evaluation. As the number of cold start training epochs increases, the marginal gain achieved by RL over the cold start checkpoint

(i.e., the slope of the dashed line) diminishes. This indicates that RL can narrow the performance gap between different base models (Liu et al., 2025). Nevertheless, although the gap is narrowed, post-RL performance remains positively correlated with the capability of the base model. This suggests that the bulk of knowledge is acquired during SFT, whereas RL primarily serves to unlock latent potential rather than explicitly push the model beyond its inherent capacity boundary (Yue et al., 2025a; Chu et al., 2025). Setting aside overfitting, the current trend suggests that a cross point may emerge in which a sufficiently strong cold start leaves no room for further improvement via RL. Whether such a point truly exists, and, if it does, what fundamental mechanisms (e.g., saturation of the policy space, diminishing exploratory signal, or intrinsic limitations of the reward model) render further RL ineffective, constitutes an important open question for future work.

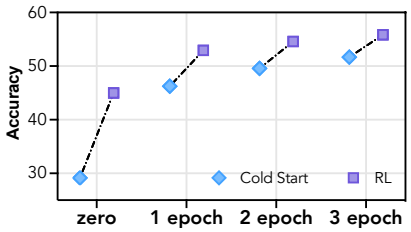


Figure 6: **Performance Gap Between Cold Start and RL** with varying cold start epochs.

6 Related Work

Agent Training. The earliest wave of LLM Agents (Wang et al., 2023; Xi et al., 2023) leverages the formidable reasoning capabilities of proprietary models (Qiao et al., 2023; Chen et al., 2025a; Yao et al., 2023; Zhou et al., 2023; Hong et al., 2024; Li et al., 2023a; Wu et al., 2023). As AI entered *the second half* (Yao, 2025), numerous benchmarks targeting complex, domain-specific agentic tasks are introduced (Mialon et al., 2024; Phan et al., 2025; Jimenez et al., 2024; Chan et al., 2025; Starace et al., 2025; Wei et al., 2025a), which expose the limitations of general-purpose agent architectures, elevating domain-specific agent training to a critical necessity. The release of Large Reasoning Models (OpenAI, 2024c; DeepSeek-AI et al., 2025; Kimi et al., 2025) marks the triumph of Reinforcement Learning (RL) for LLMs. Consequently, a surge of work has sought to adapt RL algorithms to various agent domains (Jin et al., 2025a; Song et al., 2025; Li et al., 2025c; Wu et al., 2025a; Feng et al., 2025; Qian et al., 2025; Li et al., 2025b). Yet these methods presuppose a strong backbone model; researchers are therefore compelled to synthesize copious post-training data to compensate for the backbone's deficiencies. To the best of our knowledge, we are the first to systematically investigate the scaling of agent post-training in the data-analytic scenario, aiming to provide actionable insights for data synthesis and RL-driven training in other complex agent fields.

Data-Analytic Agents and Benchmarks. Data Analysis Agents harness the reasoning capabilities and code-generation facility of LLMs to automate the end-to-end processing of data analysis tasks. Virtually all existing data analysis agents rely on closed-source models and are limited to prompt engineering. DS-Agent (Guo et al., 2024) incorporates human insights into data analysis tasks via case-based reasoning. AutoKaggle (Li et al., 2024) decomposes the data analysis pipeline into specialized sub-tasks through a multi-agent architecture. Data-Copilot (Zhang et al., 2023) and AgenticData (Sun et al., 2025) stabilize agent behavior by orchestrating operations within predefined workflows. Data Interpreter (Hong et al., 2025) further enlarges the agent's exploration space by introducing dynamic graph-based workflows. To foster progress in this domain, numerous data analysis datasets have been introduced (Hu et al., 2024; Jing et al., 2025; Liu et al., 2024; Zhang et al., 2025a; Majumder et al., 2025). Nevertheless, each adopts its own task formulation and evaluation protocol, and the majority primarily rely on human-annotated labels. In this paper, we propose a fully automated pipeline to synthesize data analysis questions and executable code trajectories. Leveraging this synthetic corpus, we train two generalist data-analytic agents with advanced performance.

7 Conclusion

This paper introduces DataMind, a scalable data synthesis and agent training recipe designed to build generalist data-analytic agents. Built on DataMind, we curate DataMind-12K, a high-quality training set that spans diverse task categories and data file formats for data-analytic tasks. Trained on DataMind-12K, we obtain DataMind-7B and 14B, two advanced data-analytic agents with superior performance on multiple benchmarks compared with various proprietary and open-source baselines. We also incorporate some empirical insights gained from our exploratory trials into the analysis experiments, aiming to provide actionable insights about agentic training for the community.

Ethics Statement

This study is carried out in strict accordance with ethical guidelines and best practices in research. All the data utilized are sourced and synthesized from publicly available resources, and no proprietary or confidential data are used. Throughout the paper, every mention or use of these data sources has been properly and accurately cited. We strongly urge all users to adhere to the highest ethical standards when using our training dataset, ensuring fairness, transparency, and responsibility in their research. Any usage of the dataset that may lead to harm or pose a detriment to society is strictly forbidden.

Reproducibility Statement

We have submitted all our training and evaluation code in the Supplementary Material. Due to OpenReview's file size limit, we only upload a $3,843$ subset of DataMind-12K. We will fully open DataMind-12K and our models DataMind-7B and 14B immediately after the double blind review process. The detailed training data synthesis and agent training methods can be found in §3 and §4. We have clearly reported the details of our evaluation datasets and metrics in §5.1 and Appx.D. The detailed information about the models and baselines we use, including the model versions of proprietary models and the reproduction details of baseline models, can be found in §5.1 and Appx.E. The code framework used and the training and inference hyperparameters are mentioned in §5.1 and Appx.F. All the prompts used in our paper are presented in Appx.G.

References

Carlo Baronio, Pietro Marsella, Ben Pan, Simon Guo, and Silas Alberti. Kevin: Multi-turn RL for generating CUDA kernels. *CoRR*, abs/2507.11948, 2025. doi: 10.48550/ARXIV.2507.11948. URL https://doi.org/10.48550/arXiv.2507.11948.

Jingyi Chai, Shuo Tang, Rui Ye, Yuwen Du, Xinyu Zhu, Mengcheng Zhou, Yanfeng Wang, Weinan E, Yuzhi Zhang, Linfeng Zhang, and Siheng Chen. Scimaster: Towards general-purpose scientific AI agents, part i. x-master as foundation: Can we lead on humanity's last exam? *CoRR*, abs/2507.05241, 2025. doi: 10.48550/ARXIV.2507.05241. URL https://doi.org/10.48550/arXiv.2507.05241.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. Mle-bench: Evaluating machine learning agents on machine learning engineering, 2025. URL https://arxiv.org/abs/2410.07095.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning. *CoRR*, abs/2310.05915, 2023. doi: 10.48550/ARXIV.2310.05915. URL https://doi.org/10.48550/arXiv.2310.05915.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *CoRR*, abs/2503.09567, 2025a. doi: 10.48550/ARXIV.2503.09567. URL https://doi.org/10.48550/arXiv.2503.09567.

Qiguang Chen, Ming-Hsuan Yang, Libo Qin, Jinhao Liu, Zheng Yan, Jiannan Guan, Dengyun Peng, Yiyan Ji, Hanjing Li, Mengkang Hu, Yimeng Zhang, Yihao Liang, Yu Zhou, Jiaqi Wang, Zhi Chen, and Wanxiang Che. Ai4research: A survey of artificial intelligence for scientific research. *CoRR*, abs/2507.01903, 2025b. doi: 10.48550/ARXIV.2507.01903. URL https://doi.org/10.48550/arXiv.2507.01903.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. Agent-flan: Designing data and methods of effective agent tuning for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pp. 9354–9366. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.557. URL https://doi.org/10.18653/v1/2024.findings-acl.557.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. *CoRR*, abs/2501.17161, 2025. doi: 10.48550/ARXIV.2501.17161. URL https://doi.org/10.48550/arXiv.2501.17161.

DeepSeek-AI. Deepseek-v3 technical report. *CoRR*, abs/2412.19437, 2024. doi: 10.48550/ARXIV.2412.19437. URL https://doi.org/10.48550/arXiv.2412.19437.

DeepSeek-AI. Deepseek-v3.1 release, 2025. https://api-docs.deepseek.com/news/news250821.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, and et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization, 2025. URL https://arxiv.org/abs/2507.19849.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, and et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783. URL https://doi.org/10.48550/arXiv.2407.21783.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms, 2025. URL https://arxiv.org/abs/2504.11536.

Etash Kumar Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, Wanjia Zhao, John Yang, Shreyas Pimpalgaonkar, Kartik Sharma, Charlie Cheng-Jie Ji, Yichuan Deng, Sarah M. Pratt, Vivek Ramanujan, Jon Saad-Falcon, Jeffrey Li, Achal Dave, Alon Albalak, Kushal Arora, Blake Wulfe, Chinmay Hegde, Greg Durrett, Sewoong Oh, Mohit Bansal, Saadia Gabriel, Aditya Grover, Kai-Wei Chang, Vaishaal Shankar, Aaron Gokaslan, Mike A. Merrill, Tatsunori Hashimoto, Yejin Choi, Jenia Jitsev, Reinhard Heckel, Maheswaran Sathiamoorthy, Alexandros G. Dimakis, and Ludwig Schmidt. Openthoughts: Data recipes for reasoning models. *CoRR*, abs/2506.04178, 2025. doi: 10.48550/ARXIV.2506.04178. URL https://doi.org/10.48550/arXiv.2506.04178.

Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. Ds-agent: Automated data science by empowering large language models with case-based reasoning. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=LfJgeBNCFI.

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. Metagpt: Meta programming for A multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o.

Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Robert Tang, Xiangtao Lu, Xiawu Zheng, Xinbing Liang, Yaying Fei, Yuheng Cheng, Yongxin Ni, Zhibin Gou, Zongze Xu, Yuyu Luo, and Chenglin Wu. Data interpreter: An LLM agent for data science. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and

Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 19796–19821. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-acl.1016/.

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. Infiagent-dabench: Evaluating agents on data analysis tasks. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=d5LURMSfTx.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, An Yang, Rui Men, Fei Huang, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report. *CoRR*, abs/2409.12186, 2024. doi: 10.48550/ARXIV.2409.12186. URL https://doi.org/10.48550/arXiv.2409.12186.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning, 2025a. URL https://arxiv.org/abs/2503.09516.

Hangzhan Jin, Sicheng Lv, Sifan Wu, and Mohammad Hamdaqa. Rl is neither a panacea nor a mirage: Understanding supervised vs. reinforcement learning fine-tuning for llms, 2025b. URL https://arxiv.org/abs/2508.16546.

Liqiang Jing, Zhehui Huang, Xiaoyang Wang, Wenlin Yao, Wenhao Yu, Kaixin Ma, Hongming Zhang, Xinya Du, and Dong Yu. Dsbench: How far are data science agents from becoming data science experts? In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=DSsSPr0RZJ.

Team Kimi, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, and et al. Kimi k2: Open agentic intelligence, 2025. URL https://arxiv.org/abs/2507.20534.

Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, Victor Zhong, Caiming Xiong, Ruoxi Sun, Qian Liu, Sida Wang, and Tao Yu. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=XmProj9cPs.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: communicative agents for "mind" exploration of large language model society. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023a. URL http://papers.nips.cc/paper_files/paper/2023/hash/a3621ee907def47c1b952ade25c67698-Abstract-Conference.html.

Haoyang Li, Shang Wu, Xiaokang Zhang, Xinmei Huang, Jing Zhang, Fuxin Jiang, Shuai Wang, Tieying Zhang, Jianjun Chen, Rui Shi, Hong Chen, and Cuiping Li. Omnisql: Synthesizing high-quality text-to-sql data at scale. *CoRR*, abs/2503.02240, 2025a. doi: 10.48550/ARXIV.2503.02240. URL https://doi.org/10.48550/arXiv.2503.02240.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chen-Chuan Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can LLM already serve as A

database interface? A big bench for large-scale database grounded text-to-sqls. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023b. URL http://papers.nips.cc/paper_files/paper/2023/hash/83fc8fab1710363050bbd1d4b8cc0021-Abstract-Datasets_and_Benchmarks.html.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, Weizhou Shen, Junkai Zhang, Dingchu Zhang, Xixi Wu, Yong Jiang, Ming Yan, Pengjun Xie, Fei Huang, and Jingren Zhou. Websailor: Navigating super-human reasoning for web agent, 2025b. URL https://arxiv.org/abs/2507.02592.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yutao Zhu, Yongkang Wu, Ji-Rong Wen, and Zhicheng Dou. Webthinker: Empowering large reasoning models with deep research capability, 2025c. URL https://arxiv.org/abs/2504.21776.

Ziming Li, Qianbo Zang, David Ma, Jiawei Guo, Tuney Zheng, Minghao Liu, Xinyao Niu, Yue Wang, Jian Yang, Jiaheng Liu, Wanjun Zhong, Wangchunshu Zhou, Wenhao Huang, and Ge Zhang. Autokaggle: A multi-agent framework for autonomous data science competitions. *CoRR*, abs/2410.20424, 2024. doi: 10.48550/ARXIV.2410.20424. URL https://doi.org/10.48550/arXiv.2410.20424.

Xiao Liu, Zirui Wu, Xueqing Wu, Pan Lu, Kai-Wei Chang, and Yansong Feng. Are llms capable of data-based statistical and causal reasoning? benchmarking advanced quantitative reasoning with data. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pp. 9215–9235. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.548. URL https://doi.org/10.18653/v1/2024.findings-acl.548.

Zihan Liu, Zhuolin Yang, Yang Chen, Chankyu Lee, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acereason-nemotron 1.1: Advancing math and code reasoning through sft and rl synergy, 2025. URL https://arxiv.org/abs/2506.13284.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob N. Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery. *CoRR*, abs/2408.06292, 2024. doi: 10.48550/ARXIV.2408.06292. URL https://doi.org/10.48550/arXiv.2408.06292.

Peixian Ma, Xialie Zhuang, Chengjin Xu, Xuhui Jiang, Ran Chen, and Jian Guo. SQL-R1: training natural language to SQL reasoning model by reinforcement learning. *CoRR*, abs/2504.08600, 2025. doi: 10.48550/ARXIV.2504.08600. URL https://doi.org/10.48550/arXiv.2504.08600.

Xinji Mai, Haotian Xu, Xing W, Weinong Wang, Yingying Zhang, and Wenqiang Zhang. Agent RL scaling law: Agent RL with spontaneous code execution for mathematical problem solving. *CoRR*, abs/2505.07773, 2025. doi: 10.48550/ARXIV.2505.07773. URL https://doi.org/10.48550/arXiv.2505.07773.

Bodhisattwa Prasad Majumder, Harshit Surana, Dhruv Agarwal, Bhavana Dalvi Mishra, Abhijeetsingh Meena, Aryan Prakhar, Tirth Vora, Tushar Khot, Ashish Sabharwal, and Peter Clark. Discoverybench: Towards data-driven discovery with large language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=vyflgpwfJW.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

OpenAI. Hello gpt-4o, 2024a. https://openai.com/index/hello-gpt-4o/.

OpenAI. Gpt-4o mini: advancing cost-efficient intelligence, 2024b. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/.

OpenAI. Introducing openai o1-preview, 2024c. https://openai.com/index/introducing-openai-o1-preview/.

OpenAI. Introducing gpt-5, 2025a. https://openai.com/index/introducing-gpt-5/.

OpenAI. Introducing openai o3 and o4-mini, 2025b. https://openai.com/index/introducing-o3-and-o4-mini/.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, Michael Choi, Anish Agrawal, Arnav Chopra, Adam Khoja, Ryan Kim, Richard Ren, Jason Hausenloy, Oliver Zhang, Mantas Mazeika, Dmitry Dodonov, Tung Nguyen, , Milind Jagota, Ronak Pradeep, and et al. Humanity's last exam, 2025. URL https://arxiv.org/abs/2501.14249.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs, 2025. URL https://arxiv.org/abs/2504.13958.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. Reasoning with language model prompting: A survey. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 5368–5393. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.ACL-LONG.294. URL https://doi.org/10.18653/v1/2023.acl-long.294.

Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Eleanor Jiang, Chengfei Lv, and Huajun Chen. Autoact: Automatic agent learning from scratch for QA via self-planning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 3003–3021. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.acl-long.165.

Shuofei Qiao, Zhisong Qiu, Baochang Ren, Xiaobin Wang, Xiangyuan Ru, Ningyu Zhang, Xiang Chen, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Agentic knowledgeable self-awareness, 2025. URL https://arxiv.org/abs/2504.03553.

Team Qwen. Qwq-32b: Embracing the power of reinforcement learning, 2025. https://qwenlm.github.io/blog/qwq-32b/.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using LLM agents as research assistants. *CoRR*, abs/2501.04227, 2025. doi: 10.48550/ARXIV.2501.04227. URL https://doi.org/10.48550/arXiv.2501.04227.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems, EuroSys 2025, Rotterdam, The Netherlands, 30 March 2025 - 3 April 2025*, pp. 1279–1297. ACM, 2025. doi: 10.1145/3689031.3696075. URL https://doi.org/10.1145/3689031.3696075.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2503.05592.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. Paperbench: Evaluating ai's ability to replicate ai research, 2025. URL https://arxiv.org/abs/2504.01848.

Aofeng Su, Aowen Wang, Chao Ye, Chen Zhou, Ga Zhang, Gang Chen, Guangcheng Zhu, Haobo Wang, Haokai Xu, Hao Chen, Haoze Li, Haoxuan Lan, Jiaming Tian, Jing Yuan, Junbo Zhao, Junlin Zhou, Kaizhe Shou, Liangyu Zha, Lin Long, Liyao Li, Pengzuo Wu, Qi Zhang, Qingyi Huang, Saisai Yang, Tao Zhang, Wentao Ye, Wufang Zhu, Xiaomeng Hu, Xijun Gu, Xinjie Sun, Xiang Li, Yuhang Yang, and Zhiqing Xiao. Tablegpt2: A large multimodal model with tabular data integration. *CoRR*, abs/2411.02059, 2024. doi: 10.48550/ARXIV.2411.02059. URL https://doi.org/10.48550/arXiv.2411.02059.

Ji Sun, Guoliang Li, Peiyao Zhou, Yihui Ma, Jingzhe Xu, and Yuan Li. Agenticdata: An agentic data analytics system for heterogeneous data, 2025. URL https://arxiv.org/abs/2508.05002.

Hanlin Wang, Chak Tou Leong, Jiashuo Wang, Jian Wang, and Wenjie Li. Spa-rl: Reinforcing llm agents via stepwise progress attribution, 2025. URL https://arxiv.org/abs/2505.20732.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A survey on large language model based autonomous agents. *CoRR*, abs/2308.11432, 2023. doi: 10.48550/ARXIV.2308.11432. URL https://doi.org/10.48550/arXiv.2308.11432.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents, 2025a. URL https://arxiv.org/abs/2504.12516.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I. Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution, 2025b. URL https://arxiv.org/abs/2502.18449.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Gang Fu, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Webdancer: Towards autonomous information seeking agency, 2025a. URL https://arxiv.org/abs/2505.22648.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen LLM applications via multi-agent conversation framework. *CoRR*, abs/2308.08155, 2023. doi: 10.48550/ARXIV.2308.08155. URL https://doi.org/10.48550/arXiv.2308.08155.

Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xeron Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, Tongliang Li, Zhoujun Li, and Guanglin Niu. Tablebench: A comprehensive and complex benchmark for table question answering. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pp. 25497–25506. AAAI Press, 2025b. doi: 10.1609/AAAI.V39I24.34739. URL https://doi.org/10.1609/aaai.v39i24.34739.

Zhenhe Wu, Jian Yang, Jiaheng Liu, Xianjie Wu, Changzai Pan, Jie Zhang, Yu Zhao, Shuangyong Song, Yongxiang Li, and Zhoujun Li. Table-r1: Region-based reinforcement learning for table understanding. *CoRR*, abs/2505.12415, 2025c. doi: 10.48550/ARXIV.2505.12415. URL https://doi.org/10.48550/arXiv.2505.12415.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan

Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huan, and Tao Gui. The rise and potential of large language model based agents: A survey. *CoRR*, abs/2309.07864, 2023. doi: 10.48550/ARXIV.2309.07864. URL https://doi.org/10.48550/arXiv.2309.07864.

Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. Simpletir: End-to-end reinforcement learning for multi-turn tool-integrated reasoning, 2025. URL https://arxiv.org/abs/2509.02479.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, and et al. Qwen2 technical report. *CoRR*, abs/2407.10671, 2024. doi: 10.48550/ARXIV.2407.10671. URL https://doi.org/10.48550/arXiv.2407.10671.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jian Yang, and et al. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

Shunyu Yao. The second half, 2025. https://ysymyth.github.io/The-Second-Half/.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *CoRR*, abs/2504.13837, 2025a. doi: 10.48550/ARXIV.2504.13837. URL https://doi.org/10.48550/arXiv.2504.13837.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, Xiangpeng Wei, Xiangyu Yu, Gaohong Liu, Juncai Liu, Lingjun Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Ru Zhang, Xin Liu, Mingxuan Wang, Yonghui Wu, and Lin Yan. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks, 2025b. URL https://arxiv.org/abs/2504.05118.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pp. 3053–3077. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.findings-acl.181.

Dan Zhang, Sining Zhoubian, Min Cai, Fengzu Li, Lekang Yang, Wei Wang, Tianjiao Dong, Ziniu Hu, Jie Tang, and Yisong Yue. Datascibench: An LLM agent benchmark for data science. *CoRR*, abs/2502.13897, 2025a. doi: 10.48550/ARXIV.2502.13897. URL https://doi.org/10.48550/arXiv.2502.13897.

Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li, Bolin Ding, and Jingren Zhou. On-policy rl meets off-policy experts: Harmonizing supervised fine-tuning

and reinforcement learning via dynamic weighting, 2025b. URL https://arxiv.org/abs/2508.11408.

Wenqi Zhang, Yongliang Shen, Weiming Lu, and Yueting Zhuang. Data-copilot: Bridging billions of data and humans with autonomous workflow. *CoRR*, abs/2306.07209, 2023. doi: 10.48550/ARXIV.2306.07209. URL https://doi.org/10.48550/arXiv.2306.07209.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. *CoRR*, abs/2403.13372, 2024. doi: 10.48550/ARXIV.2403.13372. URL https://doi.org/10.48550/arXiv.2403.13372.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang, Jing Chen, Ruipu Wu, Shuai Wang, Shiding Zhu, Jiyu Chen, Wentao Zhang, Ningyu Zhang, Huajun Chen, Peng Cui, and Mrinmaya Sachan. Agents: An open-source framework for autonomous language agents. *CoRR*, abs/2309.07870, 2023. doi: 10.48550/ARXIV.2309.07870. URL https://doi.org/10.48550/arXiv.2309.07870.

A The Use of Large Language Models

We affirm that Large Language Models are employed solely as an assisted tool to refine wording and sentence structure during our paper writing process. Their use in the experiments is strictly for scientific research purposes, and all such usage has been explicitly documented in our Experimental Settings and Reproducibility Statement. No other reliance on LLMs is involved in this work.

B Limitations

This paper still has some limitations that must be acknowledged: *a)* At present, we only incorporate reasoning-oriented data-analysis tasks; training, predictive, and data-visualization tasks are deliberately excluded and reserved as our important future work. *b)* Owing to computational constraints, our experimental backbone is restricted to the Qwen family, with model scale capped at 14B. Furthermore, not all mainstream benchmarks are covered in our evaluation suite. *c)* Limited by computational resources, we have not exhaustively evaluated all RL training algorithms; moreover, data scarcity constrains our RL runs to $\sim$ 350 steps. In future work, we will investigate more advanced RL strategies that enable stable, continual learning over substantially larger datasets.

C A More Detailed Related Work

Agent Training. The earliest wave of LLM Agents (Wang et al., 2023; Xi et al., 2023) leverages the formidable reasoning capabilities of proprietary models (Qiao et al., 2023; Chen et al., 2025a). At that time, researchers primarily boost agent performance through prompt engineering (Yao et al., 2023; Zhou et al., 2023; Hong et al., 2024; Li et al., 2023a; Wu et al., 2023). To equip open-source models with agentic skills, subsequent works introduce agent training (Chen et al., 2023; Zeng et al., 2024; Chen et al., 2024; Qiao et al., 2024) via SFT. Large-scale trajectory data, manually curated or synthetically generated by closed-source models, are used to instruct-tune open-source models. As AI entered *the second half* (Yao, 2025), numerous benchmarks targeting complex, domain-specific agentic tasks are introduced (Mialon et al., 2024; Phan et al., 2025; Jimenez et al., 2024; Chan et al., 2025; Starace et al., 2025; Wei et al., 2025a), which expose the limitations of general-purpose agent architectures, elevating domain-specific agent training to a critical necessity. However, extensive studies have shown that SFT tends to drive agent models into paradigm overfitting, severely compromising their dynamic generalization ability in sophisticated agent scenarios (Chu et al., 2025; Jin et al., 2025b; Qiao et al., 2025).

The release of Large Reasoning Models (OpenAI, 2024c; DeepSeek-AI et al., 2025; Kimi et al., 2025) marks the triumph of Reinforcement Learning (RL) for LLMs. GRPO-style algorithms (Shao et al., 2024; Yu et al., 2025; Yue et al., 2025b; Wang et al., 2025; Dong et al., 2025; Zhang et al., 2025b) enable models to autonomously explore while preserving robust generalization across diverse reasoning patterns. Consequently, a surge of work has sought to adapt GRPO-like algorithms to

various agent domains (Jin et al., 2025a; Song et al., 2025; Li et al., 2025c; Wu et al., 2025a; Feng et al., 2025; Qian et al., 2025; Li et al., 2025b; Wei et al., 2025b). Yet these methods presuppose a strong backbone model; researchers are therefore compelled to synthesize copious post-training data to compensate for the backbone's deficiencies in the target agent setting. To the best of our knowledge, we are the first to systematically investigate the scaling of agent post-training data and multi-turn RL training in the data-analytic scenario, aiming to provide actionable insights for data synthesis and RL-driven training in other complex agent fields.

Data-Analytic Agents and Benchmarks. Data Analysis Agents harness the reasoning capabilities and code-generation facility of LLMs to automate the end-to-end processing of data analysis tasks, constituting a critical component in the pursuit of autonomous scientific discovery (Chen et al., 2025b). Virtually all existing data analysis agents rely on closed-source models and are limited to prompt engineering. InfiAgent (Hu et al., 2024) pioneers the adoption of the `ReAct` (Yao et al., 2023) paradigm for tackling data analysis problems. DS-Agent (Guo et al., 2024) incorporates human insights into data analysis tasks via case-based reasoning. AutoKaggle (Li et al., 2024) decomposes the data analysis pipeline into specialized sub-tasks through a multi-agent architecture. Data-Copilot (Zhang et al., 2023) and AgenticData (Sun et al., 2025) stabilize agent behavior by orchestrating operations within predefined workflows. Data Interpreter (Hong et al., 2025) further enlarges the agent's exploration space by introducing dynamic graph-based workflows. To foster progress in this domain, numerous data analysis datasets have been introduced (Hu et al., 2024; Jing et al., 2025; Liu et al., 2024; Zhang et al., 2025a; Majumder et al., 2025; Wu et al., 2025b; Lei et al., 2025). Nevertheless, each adopts its own task formulation and evaluation protocol, and the majority primarily rely on human-annotated labels. In this paper, we propose a fully automated pipeline to synthesize data analysis questions and executable code trajectories. Leveraging this synthetic corpus, we train two generalist data-analytic agents with advanced performance.

D Datasets and Evaluation Details

We evaluate our model on three datasets related to data analysis. Here, we introduce the details and our evaluation protocols for each dataset:

- **DABench** (Hu et al., 2024). DABench evaluates LLMs in data analysis tasks across 257 challenges from 52 CSV files, covering 7 question categories. The original benchmark uses accuracy as the metric. The model's answer will be reformatted by an LLM to a specific structure and compared with the gold label using regular expression matching. Here, we directly utilize the *model-as-judge* to compare the predicted answer and the gold answer.
- **TableBench** (Wu et al., 2025b). TableBench is a real-world table reasoning benchmark spanning 18 fields and four major categories. The tables in TableBench are organized using `.json` files. So we first transform them into `.csv` files. Then, we filter the trend forecasting and chart generation questions because these questions do not have explicit gold answers. The original benchmark uses Rouge-L as the metric, and we apply *model-as-judge* instead.
- **BIRD** (Li et al., 2023b). BIRD is a widely used Text-to-SQL benchmark. We use it to evaluate our model's ability to analyze table-based databases. Since the BIRD test set requires official leaderboard submission, we adopt its validation set as our testbed for convenience. As SQL execution typically returns structured and often very large tables that are hard for a *model-as-judge* to assess, we instead materialize each result into a `.csv` file and perform an exact match comparison against the gold label.

We adopt accuracy as the final metric. For every model, we run three independent trials. We take the average score of the three trials as pass@1, while the union of the three trials is taken as pass@3 (i.e., success on any single trial counts as an overall success).

E Baselines and Reproduction Details

Models and Baselines. We compare our DataMind with five strong proprietary models: **GPT-4o** (`gpt-4o-2024-0806`) (OpenAI, 2024a), **o4-mini** (`o4-mini-2025-04-16`) (OpenAI,

2025b), **DeepSeek-R1** (`deepseek-r1-2025-0528`) (DeepSeek-AI et al., 2025), **DeepSeek-V3.1** (`deepseek-v3.1-nothinking`) (DeepSeek-AI, 2025), **GPT-5** (`gpt-5-2025-08-07`) (OpenAI, 2025a). We also include four outstanding open-source models: **QwQ-32B** (Qwen, 2025), **Qwen-2.5-Coder-32B** (Hui et al., 2024), **Llama-3.3-70B** (Dubey et al., 2024), and **Qwen-2.5-72B** (Yang et al., 2024). In addition, we select four open-source models that have been explicitly trained for data-analysis-related tasks: **TableLLM** (Wu et al., 2025b) and **Table-R1** (Wu et al., 2025c) are specialized for tabular reasoning, whereas **OmniSQL** (Li et al., 2025a) and **SQL-R1** (Ma et al., 2025) are optimized for Text-to-SQL generation. We include **Qwen-2.5-Coder-7B** and **14B** (Hui et al., 2024) as backbone models to compare different baselines. We use the Instruct version for all open-source models. Here we introduce the baselines we compare with and our reproduction details:

- **REACT** (Yao et al., 2023). For untrained models, including all proprietary models and open-source models, we test them using REACT agent format in a zero-shot manner. The detailed prompt is the same with the prompt in Fig.7.
- **TableLLM** (Wu et al., 2025b). TableLLM is trained on TableInstruct (Wu et al., 2025b), the training set of TableBench, with SFT. The data volume of TableInstruct is 19,661. We directly use TableInstruct to train Qwen2.5-Coder-7B and 14B for reproduction. As TableInstruct has three prompt modes (i.e., TCoT, SCoT, PoT), we test all of them and report the best results.
- **Table-R1** (Wu et al., 2025c). Table-R1 applies a region-enhanced SFT followed by a table-aware GRPO (Shao et al., 2024) training. The training data is also from TableInstruct for both SFT and RL stages. We follow the same data split and the training pipeline of Table-R1 to train Qwen2.5-Coder-7B and 14B for reproduction. As TableInstruct has three prompt modes (i.e., TCoT, SCoT, PoT), we test all of them and report the best results.
- **OmniSQL** (Li et al., 2025a). OmniSQL is trained on a large-scale synthesized text-to-SQL dataset SynSQL-2.5M with a data volume of 2.5M. Since it already has Qwen2.5-Coder-7B and 14B version models, we directly use them for evaluation.
- **SQL-R1** (Ma et al., 2025). SQL-R1 is further trained on OmniSQL with RL. The training data for RL is sourced from a 5K subset of SynSQL-2.5M. Since it already has open-source Qwen2.5-Coder-7B and 14B version models, we directly use them for evaluation.

F TRAINING AND INFERENCE DETAILS

We use LlamaFactory (Zheng et al., 2024) for SFT training and verl (Sheng et al., 2025) for RL training. For SFT, our learning rate is $1e-5$ with a warmup ratio of 0.1 and a cosine decay schedule. Our global batch size is set to 16. For RL, we use a learning rate of $1e-6$. The batch size is 16 with a mini batch size of 2. The rollout temperature is 0.7, the top-p is 1.0, and the group size G is 4. We schedule γ via cosine decay, annealing from a peak of 0.9 to a valley of 0.05. At test time, we fix the temperature to 0.7, top-p to 0.95, and an inference batch size of 5 for all evaluations. The detailed hyperparameters employed in DATAMIND are presented in Tab.2.

G PROMPTS USED IN OUR PAPER

G.1 TRAINING AND EVALUATION PROMPT

Training and Evaluation Prompt

You are an expert-level data analyst and statistician who solves any data challenge through rigorous logic, systematic planning, and deep investigation. Your primary task is to answer user questions by analyzing the provided data source. You can solve the given problem step by step by utilizing Python code execution (for CSV files) or SQL queries (for database files) to support your reasoning.

 # Problem-Solving Protocol

1. You should think through the problem logically, outlining your reasoning process in <think> and </think> tags.
2. After reasoning, write the appropriate code to execute your plan. Place your code between

Table 2: Detailed hyperparameters used in our paper.

Stage	Hyperparameter	Value
SFT	learning rate	1e-5
	lr scheduler type	cosine
	warmup ratio	0.1
	batch size	16
	training epoch	3
	cutoff length	8192
SFT+RL	learning rate	1e-6
	lr warmup style	constant
	lr warmup steps	20
	batch size	16
	mini batch size	2
	training epoch	1
	max prompt length	2048
	max response length	8192
	clip ratio low ε_{low}	0.2
	clip ratio high $\varepsilon_{\text{high}}$	0.28
	rollout temperature	0.7
	rollout topp	1.0
	rollout group size G	4
	γ scheduler type	cosine
	γ peak value	0.9
	γ valley value	0.05
	length reward l_{min}	256
	length reward l_{max}	1024
Inference	temperature	0.7
	topp	0.95
	batch size	5

<code> and </code> tags.
- For CSV files and Excel files, write Python code using libraries like pandas, numpy, sklearn, etc. to analyze the data. The format should be:
```python
<your python code here>
```
- For database files, you should only use the 'get_db_info' function to view the database structure information and 'execute_sql' function to execute sql queries, and save the results. Example:
```python
get_db_info()
```
or
```python
execute_sql(sql="a valid SQL query here", output_path="result.csv")
```
3. The execution results will be returned in <interpreter> and </interpreter> tags.
4. Every time you get the code execution result, you must conduct reasoning and analyze these results carefully between <think> and </think>. If the result indicates an error or unexpected behavior, explain the issue and rewrite the previous step code. If the result

indicates the code ran successfully, analyze whether the original problem has been fully solved.
- If it has been solved, explain your reasoning and then provide the final answer wrapped in <answer>...</answer>.
- If not, continue reasoning and provide the next step of code based on your previous correct code.
4. Whenever you're confident about the result, you can directly provide your answer to the question inside <answer>...</answer>.
- For CSV files and excel files, you should directly provide your answer. Make it concise and to the point. (e.g. <answer>The final answer is 3, ...</answer>)
- For database files, you must tell me the file name of the result CSV file. (e.g. <answer>The final answer is saved in the CSV file named 'result.csv'.</answer>)

 # CSV File and Excel File Analysis Notes
1. Load data from 'data/files' directory using the specified CSV or Excel filename. Temporary files can be saved to 'data/tmp'.
2. In your first step, you should use print() to inspect the data columns, the first 3 rows, and the type of the columns and so on to understand the data structure.
3. If you want to get the value of a variable in your code, you must print it out using print() (e.g., print(f"max: {{max}}")) to understand the current value and state of variables.
4. Only proceed to the next step of code if the current step is written correctly. Each step must build on the previous code.

 # Database File Analysis Notes
1. You can only use sqlite database engine to execute your SQL queries.
2. In your first step, use get_db_info() to inspect the database schema.
3. In your answer, you must provide the file name of the result CSV file. Make sure the answer file has been saved in the current directory.

 # Additional Notes:
1. Avoid including irrelevant commentary outside of the designated tags <think>, <code>, <interpreter>, and <answer>.
2. If the last step is not correct, you should first conduct a deep analysis of the previous step and then rewrite the code to fix the issue.
3. Keep your responses concise, structured, and directly tied to the original question.

 # Data Source
The data source path is '{data_source_path}'.

Figure 7: Prompt for Training and Evaluation.

G.2 Query Synthesis Prompt

Query Synthesis Prompt

You are a data analysis expert and assistant. Your task is to generate high-quality, insightful data analysis questions based on a given data file's metadata, headers, and column descriptions. You always consider the semantics of the data and produce questions that can support exploratory data analysis, business understanding, or hypothesis generation. Your output should be clear, structured, and directly usable for data analysis planning.

 # Objective
Based on the information from the data file and the specified question type, generate one unique and meaningful exploratory data analysis (EDA) question. The phrasing and structure of the question must closely follow the style of the provided examples be-

low. The goal is to generate questions that can reveal various types of insights from the dataset.

—

Here is some meta information about the data file:

Description:
{description.replace('#', '###')}

Header:
{meta_info['head']}

Columns:
{meta_info['columns']}

Columns Type:
{meta_info['type']}

Columns Range:
{meta_info['range']}

Columns Unique Values:
{meta_info['unique']}

Row Number:
{meta_info['row_count']}

Column Number
{meta_info['column_count']}

—

EDA Question Types with Short Descriptions
1. **Aggregation** - Summarize data values to understand overall patterns or trends, such as calculating averages, totals, or maximums. Often used to quantify general behavior across a dataset.
2. **Ranking** - Identify and compare items based on specific metrics to determine their relative standing, often highlighting the highest, lowest.
3. **Counting** - Determine the number of items that meet specific conditions or criteria. This typically involves filtering data and counting matching entries.
4. **Comparison** - Compare values across data points to identify differences, similarities, or extremes, often focusing on identifying the highest, lowest, or range between values.
5. **Domain Specific** - Analyze data within a specific field or context using domain knowledge to interpret results, answer specialized questions, or derive insights meaningful to that area.
6. **Causal Analysis** - Analyzing relationships beyond correlation, often requiring controlled experiments or advanced statistical methods to infer causality.
7. **Statistical Analysis** - Apply statistical measures (e.g., median, standard deviation, variance, growth rate) to summarize, describe, or evaluate patterns and variability within the data.
8. **Correlation Analysis** - Measure the strength and direction of the relationship between two quantitative variables, typically using a correlation coefficient to assess how closely the variables move together.
9. **Arithmetic Calculation** - Perform basic mathematical operations (e.g., addition, subtraction, multiplication, division) to compute totals, differences, or projections based on the given data.
10. **Descriptive Analysis** - Provide an overview of the dataset by explaining its structure,

key columns, and any observable patterns or trends. Focus on summarizing what the data shows without drawing causal inferences.
11. **Impact Analysis** - Analyze relationships between variables to determine how one factor influences another. This involves identifying trends, correlations, or causations within the data to assess impact over time or across categories.
12. **Fact Checking** - Retrieve and verify multiple related facts across different data points to answer a question. This requires connecting and cross-referencing information from various parts of a dataset.
13. **Anomaly Detection** - Identify data points that significantly differ from the expected pattern or norm, potentially indicating errors, outliers, or unusual behavior.
14. **Multi-hop Numerical Reasoning** - Perform numerical reasoning that requires combining multiple pieces of information or steps. This often involves intermediate calculations or logical sequencing to reach the final answer.
15. **Time-based Calculation** - Analyze data across time periods to identify trends, changes, or cumulative values, often involving comparisons between different time intervals or calculating growth rates over time.
16. **Distribution Analysis** - Analyze how data values are distributed for a given variable or across groups. This includes assessing normality (e.g., via the Shapiro-Wilk test), skewness, kurtosis, or comparing distributions between groups using statistical tests (e.g., the Mann-Whitney U). It helps in understanding the shape, spread, and symmetry of the data, and whether it meets assumptions required for other analyses.
17. **Feature Engineering** - Create, transform, combine, or extract variables to enhance data quality or modeling potential. This includes generating new columns, deriving ratios or indicators, aggregating related values, or reformatting data to reveal deeper patterns or prepare for predictive analysis.
18. **Comprehensive Data Preprocessing** - Perform a sequence of data cleaning and preparation steps to ensure the dataset is ready for analysis or modeling. This includes handling missing values, transforming data types, encoding categorical variables, normalizing or scaling numerical features, and correcting inconsistencies. The goal is to produce a clean, well-structured, and analysis-ready dataset.

—

Analysis Question Type Focus

Primary Question Type:
The question should primarily focus on the given type:
{question_info['question_type']}

—

Requirements:
- Return exactly one data analysis question per execution.
- The question should mainly focus on the specified question type:
{question_info['question_type']} - Use clear, concise, and practical language suitable for data analysts.
- The question should be grounded in the context and structure of the data file.
- **The phrasing and style of the question must closely mirror the examples provided**. This includes using similar formats, tones, and syntactic patterns.
- **Follow the same linguistic conventions as the examples**. For example, if the examples use formulations like "What is the difference between...", "How many...", or "Which of the following...", then your generated question must follow similar patterns to ensure stylistic consistency.
- **Output format must include**:
- The question inside '<question>...</question>'
- A short description inside '<description>...</description>'
- The question type inside '<type>...</type>'

—

Template Enforcement
You are strictly required to follow the templates provided for the question type below:
{get_question_template(question_info['question_template'])}

- You must use the template to generate the corresponding question.
- Generate the question by using the template with relevant columns, values, or descriptions from the dataset.
- Do not introduce new question styles, structures, or alternative phrasings.

This is a hard constraint, not a suggestion.

—

Here are some examples:
{question_info['question_example']}

—
Now, generate a high-quality EDA question:

Figure 8: Prompt for Query Synthesis.

G.3 Trajectory Sampling Prompt

Trajectory Sampling Prompt

You are a Data Analysis Assistant who can solve the given problem step by step with utilizing a code execution tool to support your reasoning.

1. You should think through the data analysis problem logically, outlining your reasoning process in the <reasoning>...</reasoning> tags.

2. After reasoning, you can write Python code if necessary and use print() statements to inspect key values to support your reasoning. Remember to place your Python code inside <code>```python ... ```</code> tags. You may use libraries like pandas, numpy, sklearn, etc.

3. User will execute the code and return results in <interpreter>...</interpreter>. Every time you get the code execution result, you must conduct reasoning and analyze these results carefully between <reasoning> and </reasoning>, following this process:
Whether the result indicates an error or unexpected behavior, explain the issue and rewrite the previous step code, or the result indicates the code ran successfully, analyze whether the original problem has been fully solved.
- If it has been solved, explain your reasoning and then provide the final answer wrapped in <answer>...</answer>.
- If not, continue reasoning and provide the next step of code based on your previous correct code.

4. Whenever you're confident about the result, you can directly provide your answer to the question inside <answer>...</answer>.

5. Format Example:
- Code Format:
<reasoning>

Your reasoning here, step by step, explaining your thought process and how you will approach the problem.
</reasoning>

<code>
```python
Your Python code here, using print() statements to inspect variables.
```
</code>

- Answer Format:
<reasoning>
Your final reasoning here. Summarize the solution to the question.
</reasoning>

<answer>
Your final answer to the question, keep your answer as brief as possible while ensuring it is complete, concise, and clearly stated. **No longer than 1024 words.**
</answer>

6. Important Notes:
- To read data files, load from the '{file_dir}' using the file '{question['filename']}'.
- **If you want to get the value of a variable in your code, you must print it out using print() (e.g., print(f"max: {{max}}"))** to understand the current value and state of variables. Try to use one or two print() statements in one single step to avoid overwhelming the user with too many outputs.
- Don't use visualization libraries like matplotlib or seaborn, as the user will not be able to see the plots.
- Only proceed to the next step of code if the current step is written correctly. Each step must build on the previous code.
- Avoid including irrelevant commentary outside of the designated tags <reasoning>, <code>, <interpreter>, and <answer>.
- Keep your responses concise, structured, and directly tied to the original question.

7. When beginning to solve a problem, you are encouraged to follow these two initial steps as a general guideline (not a rigid rule):
- Step 1: Load the Excel file specified in the question using pandas.read_excel() from the path '{file_dir}/{question["filename"]}'. Then, inspect the dataset structure by printing:
- the first 3 rows using df.head(3)
- the list of column names using df.columns
- optionally, the data types using df.dtypes if helpful for understanding column types.
- Step 2: Analyze the question to determine which columns or rows in the dataset are relevant. Then apply appropriate operations to those columns/rows to address the problem.
- For complex tasks, you may adapt or break the second step into smaller sub-tasks and solve them incrementally. This approach helps manage complexity and ensures accurate reasoning throughout the analysis.
- Error Handling: If your code contains a bug or raises an exception during execution, first focus on debugging and resolving the issue before proceeding with solving the main problem.

{get_few_shot(question['question_type'])}

Figure 9: Prompt for Trajectory Sampling.

G.4 JUDGE MODEL PROMPT

Judge Model Prompt (Self-consistency)

You are a precision evaluation system. Your task is to determine whether three AI-generated answers are equivalent and fully correct. Use the following evaluation criteria:

1. Semantic Equivalence: For descriptive questions, all answers must express the same meaning, even if phrased differently.
2. Numerical Agreement: For numerical/calculation questions, any numeric values must be within 3% of each other. Convert all units to be comparable before comparing. If a number is derived through a formula, check that the process is logically sound.
3. Completeness: The final answer must fully address all aspects of the original question. Do not ignore implicit sub-questions.
4. Source Traceability: Identify which answer (1, 2, or 3) most clearly and accurately represents the final synthesized answer.

Respond using:
- <reasoning>...</reasoning>. Please use this tag to think through the evaluation process step by step, explaining your reasoning and how you will choose the best answer. Be careful to consider all three answers.
- <correct>...</correct>. Use 'yes' or 'no' to indicate whether all three answers are semantically equivalent and fully correct. It's fine if the differences are justifiable.
- <number>...</number>. Use 1, 2, or 3 to choose the single best answer that should be used as the final response. If you cannot determine a clear best answer, choose the one that is most complete and accurate.

Question
question

Answer 1
answer_1

Answer 2
answer_2

Answer 3
answer_3

Evaluate standard:
1. Are all three answers semantically equivalent and/or numerically consistent ($\leq$3% difference)?
2. Do all three answers fully resolve the original question without omissions?
3. Which single answer is the best basis for a final polished response?

Respond in this format:
<reasoning>...</reasoning>
<correct>yes</correct> or <correct>no</correct>
<number>...</number>

Figure 10: Prompt for Consistency Judge Model.

Judge Model Prompt (Reward)

You are a fair and professional evaluator. Your task is to assess how closely an AI assistant's answer matches the provided ground truth for a given question. You are to provide a numerical score for how well the response answers the question based on the ground truth

answer.
Your evaluation should focus on the assistant's answer to the question. Begin your evaluation by comparing the assistant's answer with the ground_truth answer. Identify and correct any mistakes. Be as objective as possible.

Evaluate the correctness (0 for incorrect, 1 for correct) of the predicted answer to the question:

Question: {question}

Predicted answer: {pred_answer}

Ground truth answer: {ground_truth}

Rules for judgment:
1. For numerical questions, any result within 3% of the ground truth answer is considered correct. Please compare abs(Predicted answer)/abs(True answer) with 3% to make your decision.
2. For multiple-choice questions, an exact match is required.
3. The answer should be clear and complete.
4. The calculation process alone is not considered correct.

Wrap your reasoning inside <thought></thought> and wrap the accuracy score inside <score></score> tags. Keep your reasoning concise, no more than 3-5 clear and informative sentences. Avoid repetition or unnecessary elaboration. Only output the reasoning and score using the required tags. Follow the output format as shown in the example below:

<thought>The predicted answer is 115624, which exactly matches the ground truth. The relative error is 0, well within the 3% threshold. The answer is clear, correct, and directly responds to the question.</thought>
<score>1</score>

Figure 11: Prompt for Reward Judge Model.