KRAMABENCH: A BENCHMARK FOR AI SYSTEMS ON DATA INTENSIVE TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Discovering insights from a real-world data lake potentially containing unclean, semi-structured, and unstructured data requires a variety of data processing tasks, ranging from extraction and cleaning to integration, analysis, and modeling. This process often also demands domain knowledge and project-specific insight. While AI models have shown remarkable results in reasoning and code generation, their abilities to design and execute complex pipelines that solve these data-lake-to-insight challenges remain unclear. We introduce **KRAMABENCH**[1] which consists of 104 manually curated and solved challenges spanning 1700 files, 24 data sources, and 6 domains. **KRAMABENCH** focuses on testing the end-to-end capabilities of AI systems to solve challenges which require automated orchestration of different data tasks. **KRAMABENCH** also features a comprehensive evaluation framework assessing the *pipeline design* and *individual data task implementation* abilities of AI systems. Evaluating 8 LLMs with our single-agent reference framework DS-Guru, alongside open- and closed-source agentic systems, we find that while current single-agent systems may handle isolated data-science tasks and generate plausible draft pipelines, they struggle with producing working end-to-end pipelines. On **KRAMABENCH**, the best system reaches only 50% end-to-end accuracy in the full data-lake setting. Even with perfect retrieval, the accuracy tops out at 59%. Leading LLMs can identify up to 42% of important data tasks but can only fully implement 20% of individual data tasks.

1 INTRODUCTION

The goal of data science is to obtain insights from raw data. A data science workflow typically involves manually selecting data and designing pipelines that perform data wrangling, conduct data analyses, and extract findings, among other data tasks. These workflows (Figure 1) are expected to handle noisy, domain-specific data and scale to data lakes with tens to thousands of files, necessitating multi-step, data-dependent reasoning and coordination across data tasks (Guo et al., 2024; Shankar et al., 2025).

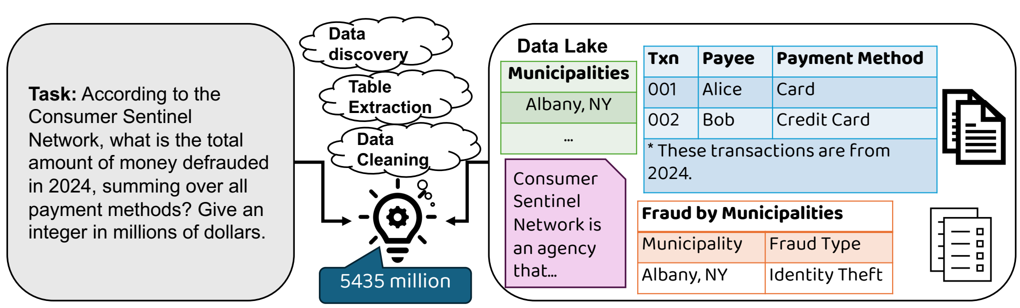

Figure 1: One of the tasks of **KRAMABENCH** based on a real data lake of 136 files in the legal discovery domain. Data file sample snippets are simplified.

[1]Assets available at `https://anonymous.4open.science/r/Kramabench-7D6D/`

Table 1: Comparing existing benchmarks. (– indicates partial satisfaction, e.g., not for all tasks)

Benchmarks	DS-1000 Lai et al. (2023)	ARCADE Yin et al. (2023)	DA-Code Huang et al. (2024)	DataSciBench Zhang et al. (2025a)	DSBench Jing et al. (2025)	BLADE Gu et al. (2024)	ScienceAgentBench Chen et al. (2025)	**Ours**
DS Tasks								
Data discovery	✗	✗	✗	✗	✗	✗	✗	✔
Multi-file integration	✗	✔	✔	✗	✔	✗	✗	✔
Data cleaning	✔	✔	✔	✔	✗	✗	✗	✔
Data preparation	✔	✔	✔	✔	✔	✔	✔	✔
Data analysis	✔	✔	✔	✔	✔	✔	✔	✔
Modeling	✔	✔	✔	✔	✔	✔	✔	✔
Abilities tested								
Data semantics	✗	✗	✗	✗	–	–	✗	✔
Domain knowledge	✗	✗	✗	✗	✗	✔	✔	✔
Multi-step reasoning	✔	✔	✔	✔	✔	✔	✔	✔
Evaluation								
Implementation	✔	✔	✔	✔	✔	✔	✔	✔
Pipeline design	✗	✗	✗	✗	–	–	✗	✔
End-to-end	✗	✗	✗	✗	✗	✗	✗	✔

While recent research has advanced individual components of these workflows such as code generation (Nam et al., 2024; Wang & Chen, 2023), tool use (Qin et al., 2024b;a), and natural language question answering (Zhang et al., 2024b; Pu et al., 2023), the challenge of designing and executing complete end-to-end data science pipelines remains underexplored.

Progress towards practical data-to-insight systems has been hindered by the lack of benchmarks that reflect the real-world complexity of these workflows. Existing benchmarks focus on isolated steps, such as code generation from fine-grained prompts (Lai et al., 2023; Zhang et al., 2025a; Huang et al., 2024; Yin et al., 2023), text-to-SQL (Lei et al., 2025; Zhang et al., 2024a), and modeling using curated input (Gu et al., 2024; Mitchener et al., 2025; Chen et al., 2025). We list these works in Table 1 and discuss more in Section 5. While immensely useful, these benchmarks do not capture the heterogeneity of data tasks and the accompanying reasoning demands of real-world data science involving large, domain-specific, and unclean input datasets.

To bridge this gap, we introduce **KramaBench** [2] [3], a benchmark designed to evaluate LLM-based systems on complex end-to-end data science pipelines. **KramaBench** consists of 104 tasks drawn from 1700 real-world files across 24 sources in 6 domains. All tasks are manually curated from fresh, domain-specific sources and paired with expert reference solutions grounded in accessible data. Each task is specified in natural language and requires systems to discover relevant data, perform data wrangling such as cleaning and normalization, and implement statistical or computational analyses to produce insights. To study public data's leakage into LLM training, we obscured the input of 20% of tasks through replacing real-world identifiers and numeric data with synthetic ones without changing the task structure. We hold them out for evaluation to prevent them from being trained on.

For each task, we provide reference sub-tasks that a system capable of solving the end-to-end task should be able to solve. Sub-tasks are also annotated with ground truth results and text descriptors. These assets facilitate our comprehensive evaluation framework with three settings. (1) The most important *end-to-end automation* setting assesses the ability to solve tasks without a human in the loop. (2) The *pipeline design* setting assesses the ability to reason and identify key components towards a successful pipeline design. (3) The *individual task implementation* setting assesses the ability to act on fine-grained descriptions of individual sub-tasks in a correct pipeline.

We evaluated **KramaBench** across eight models, along with three different configurations of DS-Guru and three other existing agentic systems (Hugging Face, 2025; OpenAI, 2025; Google, 2025). We conducted extensive ablations studies and failure analyses, taking advantage of our comprehensive evaluation framework and obscured inputs.

Through **KramaBench**, we observed multiple insights about where LLM systems are successful: (1) Agentic control flow is helpful with **KramaBench**'s challenges: a canonical single-agent system (*smolagents-single*) that iteratively search, plan, and repair achieve 47.23% end-to-end accuracy,

[2] We substantially improved upon an earlier version of this work.

[3] The name KramaBench is a reference to the "Vinyasa Krama" practice of Yoga

outperforming the strongest configurations of DS-Guru (22% overall), which uses a structured control flow. (2) a canonical multi-agent system (*smolagents-reflexion*, Shinn et al. (2023)) using an evaluator agent and reflections achieves 50.64% end-to-end accuracy. The mild improvement (+3.41%) indicates ample space for research on data-intensive agentic systems.(3) LLM systems can reason at a coarse level about the data operations required by a successful workflow and generate plausible pipelines, achieving 42% on pipeline design.

Our analyses also reveal some persistent challenges: (1) Retrieval from a data lake is problematic, but not the dominating obstacle. Supplying only the gold files improves overall accuracy by only 9-10% across systems using different retrieval mechanisms. (2) Weaknesses in fine-grained data-dependent reasoning cause models to fail. Systems even fail most of the time at implementing individual simple sub-tasks, capping at 19.75% when evaluated under the individual task implementation described above. (3) Agents often fail to achieve a holistic understanding of the data lake. We observe that the agents often overly rely on their prior knowledge (12%-16% performance fluctuation on obscured inputs), or assume clarifications will be given from a user (22% of failures).

2 The Design of **KramaBench**

Tasks in **KramaBench** are based on real-world data science challenges from six domains: archeology, astronomy, biomedical research, environmental science, legal insight discovery, and wildfire prevention. Each domain is associated with a data lake containing raw files in structured, semi-structured, or unstructured formats from multiple sources. Each **task** is a natural language description of a domain-specific data science problem. The goal of a system under test is to design and execute an end-to-end pipeline that takes the entire domain data lake as input and produces the correct output. In addition to the target answer, **KramaBench** provides the ground truth solution both in code and in annotated **sub-tasks**: natural language descriptions of smaller building-block operations that are essential elements within a full solution along with a prompt and their target answers. These finer-grained references enable the evaluations of pipeline design and individual task implementation.

2.1 Task Design and Validation

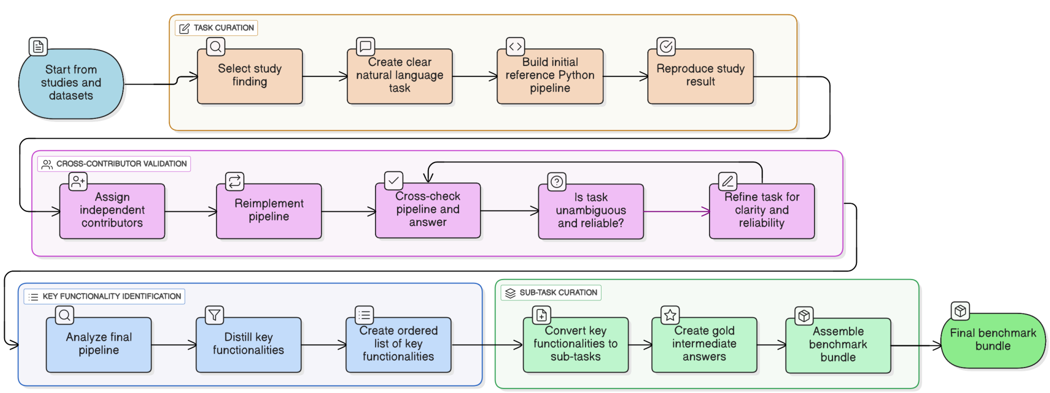

Figure 2: Workflow for task design and validation in **KramaBench**, detailing the curation, validation, and functional decomposition to ensure quality and consistency across tasks.

To curate tasks, we started with published studies and reports that (1) contain quantitative or graphical findings produced by data analysis, (2) are based on complete and publicly accessible datasets, and (3) require complex multi-step pipelines involving heterogeneous and noisy inputs. Grounding onto these studies and reports ensures that our tasks reflect real-world data science pipelines. We followed a 4-step workflow involving tight validations and repeated verifications of reference solutions to ensure the quality of tasks, reference solutions, and fine-grained annotations. We summarize the process in Figure 2.

Step 1: Task Curation. For each study or report, we reproduced its important findings using the associated datasets, transforming these findings into problem statements. Within the same domain,

Table 2: Detailed breakdown of per-domain tasks in **KRAMABENCH**. Hard tasks require multiple files or pipelines with more than three steps.

Domain	# Tasks (sub-)	%Hard Tasks	# Files (size)
Archeology	12 (71)	50.00%	5 (7.5MB)
Astronomy	12 (68)	50.00%	1556 (486MB)
Biomedical	9 (38)	66.66%	7 (175MB)
Environment	20 (148)	70.00%	37 (31MB)
Legal	30 (188)	53.33%	136 (1.3MB)
Wildfire	21 (120)	71.42%	23 (1GB)
Total	104 (633)	60.58%	1764 (1.7GB)

Table 3: Answer type and example questions.

Type	Metric score
String (exact)	Accuracy (0/1)
String (approximate)	ParaPluie paraphrase (0/1)
Numeric (exact)	Accuracy (0/1)
Numeric (approximate)	$1/(1+\text{RAE})$ (0-1)
List (exact)	F1 score (0-1)
List (approximate)	F1 score (if match > 0.9)

more tasks similar to the real-world ones are curated via integrating different data sources. The creator of each task supplies a concrete implementation of the pipeline.

Step 2: Cross-Contributor Validation. For each task, a different second contributor independently attempts to develop a solution. A third contributor compares the solution with the one in Step 1., resolves ambiguities in the problem statement, and checks in a reference pipeline. The execution time of the reference pipeline is also recorded.

Step 3: Key Functionality Identification. A data science problem can have multiple valid solution pipelines. However, certain data processing steps *must* exist in any correct pipeline. A simple example would be "*identifying the column containing the temperature to be* `Temp`". We draft a list of these key functionalities for each task using the reference pipeline via instruction-tuning GPT-o3 and manually polish the outputs to make sure the description of these sub-tasks do not depend on specific implementation choices. The semi-automation scripts are available at our repository.

Step 4: Sub-task Curation. We transform each sub-task description into a prompt via instruction tuning a local instance of Gemma3-27b and manual inspection. The example in Step 3 would be transformed to "*which column contains the temperature information*"? The target answers to each sub-task are manually verified using the reference pipeline.

Table 2 reports the statistics and difficulty distributions of the 6 domains and their tasks. We provide more detailed descriptions and an example of tasks, key functionalities, and sub-tasks in Appendix E.

2.2 EVALUATION MECHANISM

As discussed in Section 1, **KRAMABENCH** evaluates systems on three capabilities. In Figure 3, we provide an overview of these 3 areas. Our primary focus is (1) end-to-end automation.

(1) End-to-end Automation. For each task, the system output is given a **score** in $[0, 1]$ based on the reference target answer. The scoring schemes for each possible answer type address fuzzy matches and are discussed in Table 3. The string approximation metric uses the method introduced in (Lemesle et al., 2025). Our validation study (details in Subsection G.2) for this LLM-as-a-judge method shows 84% agreement between human annotators and the LLM. Given a domain workload W consisting of numerous tasks $T's$, the **total** score of a system $\mathcal{F}$ for W is $\text{Mean}_{T\in W}\text{score}(\mathcal{F}(T))$. The score of $\mathcal{F}$ for the entire benchmark suite is analogous.

Results under the following two less-automated evaluation settings provide insights into why a system may succeed or fail in the end-to-end automation setting and the abilities of a system to assist with a human-in-the-loop. Figure 8 describes the mapping of the following tasks.

(2) Pipeline Design. For each task, we assess the system generated pipeline using the key functionalities that *any correct pipeline* needs to contain in some form (from Step 3 of Subsection 2.1). We score the system with the fraction of key functionalities covered in the pipeline produced by the system. Coverage is evaluated via LLM-as-a-judge following the method in Tong & Zhang (2024) using the description obtained in Step 3.

(3) Sub-task Evaluation. We provide systems with the problem statements for sub-tasks and compare system outputs to human-curated target answers (as in Step 4 of Subsection 2.1) using the same scoring approach as in end-to-end evaluation. Full technical details of these evaluations are provided in Appendix G.

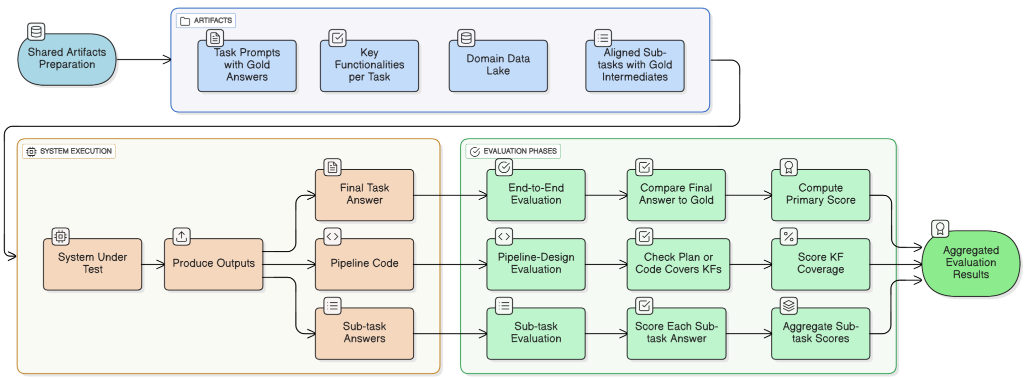

Figure 3: Overview of **KRAMABENCH**'s evaluation process.

2.3 REFERENCE IMPLEMENTATION

We introduce DS-Guru (Figure 4), a lightweight framework that serves as minimal scaffolding to enable a single out-of-the-box LLM to attempt the data science challenges in **KRAMABENCH**. DS-Guru has three variants. *No-context:* The LLM is invoked one-shot with the problem description and the names and paths of the files from the data lake, without any file contents. *One-shot:* The LLM is invoked one-shot with the problem description and sample snippets from each data file. *Few-shot:* The LLM is first invoked once with the task description and sample snippets, then re-invoked few-shot with execution results and error messages from the pipeline it implemented in the previous shot. With all variants, DS-Guru instructs the LLM to decompose the task into simpler tasks before attempting to implement each task provide the concrete pipeline implementation along with the answer.

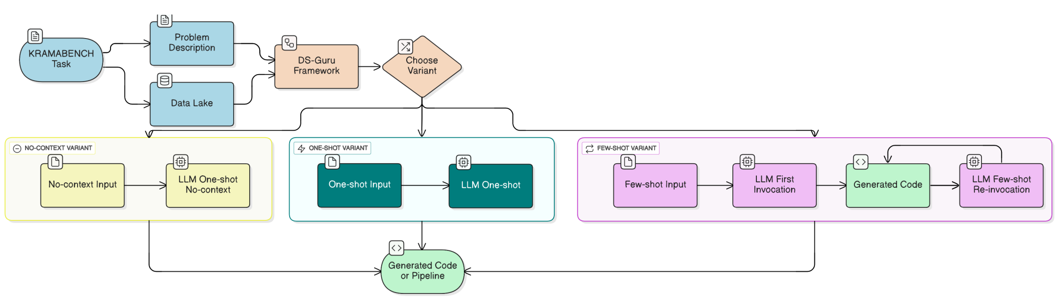

Figure 4: DS-Guru, a lightweight framework that scaffolds out-of-the-box LLMs with multiple variants to tackle **KRAMABENCH**.

DS-Guru succinctly addresses where out-of-the-box LLMs struggle with **KRAMABENCH**. (1) Realistic data lakes exceed LLM context windows. DS-Guru uses budgeted, type-annotated **one-pass sampling (OPS)** retrieval to make this step tractable. (2) Many data science tasks require many different data operations. DS-Guru uses chain-of-thought prompting (Wei et al., 2022) to encourage decomposition before code synthesis. (3) Code running on real-world uncurated data are subject to more sporadic errors compared to code for well-structured tasks. DS-Guru's multi-shot approach (Press et al., 2023) can help LLMs recover from such errors. More details on DS-Guru in Appendix B.

3 EXPERIMENTAL SETUP

We accessed all LLMs in different systems via OpenAI and Together APIs; pipelines generated by systems are executed locally.

DS-Guru: We combine each of the three variants (as in Subsection 2.3) of DS-Guru with six LLMs: GPT-o3, GPT-4o, Claude-3.5-Sonnet, Llama3.3, Deepseek-R1-70B, and Qwen2.5-Coder-32B (OpenAI, 2025; 2024; Anthropic, 2024; Meta AI, 2025; DeepSeek-AI et al., 2025; Hui et al., 2024), totaling to 18 concrete DS-Guru implementations.

smolagents Deep Research (smolagents DR): We use Hugging Face `smolagents` (Hugging Face, 2025) to evaluate deep research-style agentic systems on our benchmark under both single-agent and multi-agent settings. On the single agent setting (`smolagents-single DR`), `smolagent's` official single-agent deep research implementation using code actions (git), we report results with four different LLMs as the LLM: GPT-o3, GPT-4o, Claude-3.5-Sonnet, and Claude-3.7-Sonnet.

For the multi-agent setting, we implemented two representative architectures also using `smolagents`: (1) `smolagents-reflexion (Shinn et al., 2023)` which follows an actor → evaluator → reflection agent loop. (2) `smolagents-pdt` which follows a **P**lanner and Task **D**ecomposer → **T**ool Executor workflow (PDT), inspired by Fan et al. (2025). We provide more details in Appendix C. We view these systems representative of open-source "deep research" projects (e.g., Alibaba's Academy (2025)).

Closed-Source Deep Research Systems: OpenAI Deep Research (**OpenAI DR**, OpenAI (2025)) and **Gemini Pro-2.5 Agentic Mode (Gemini Agentic)** (Google DeepMind, 2025; Google, 2025) were evaluated manually through their web interfaces under the end-to-end automation setting. We made best efforts instructing them not to search online. However, this restriction was not enforceable.

Table 4: Comparison of different mechanisms across systems.

Systems	Retrieval mechanisms	Input modes	Control flow	Internet Access
DS-Guru	One-Pass Sampling (OPS)	Full, Trimmed, Oracle	Structured loops	Off
smolagents-single DR	Agentic retrieval	Full, Trimmed, Oracle	Agentic loops	Off
smolagents-reflexion DR	Agentic retrieval	Full, Trimmed, Oracle	Multi-agent	Off
smolagents-pdt DR	Agentic retrieval	Full, Trimmed, Oracle	Multi-agent	Off
OpenAI DR	Agentic retrieval	Trimmed, Oracle*	Agentic loops	On
Gemini Agentic	Agentic retrieval	Trimmed, Oracle*	Agentic loops	On

Human Baseline. We conducted a small-scale human study in which 9 data-science practitioners solved **KRAMABENCH** under the same conditions and requirements as LLM systems under test in the end-to-end, full input setting. Details can be found in Appendix F.

We evaluated six different systems, which differ in four important ways.

Retrieval Mechanisms. DS-Guru employs *One-Pass Sampling* (*OPS*) retrieval: a budgeted, type-annotated sample of each file in the data lake (schema summaries + a small row sample) is provided to the LLM once. *OPS* scales with data lake size but constrains the LLM's direct interaction to sampled views. DR systems employ *agentic retrieval*: the LLM plans the retrieval and issues file system tool calls to iteratively read, filter, and revisit sources, offering richer interaction but at a higher cost.

Input Modes. *Full*: ideally, the entire input lake is available to the system's retriever. *Oracle*: only the gold files are provided (no discovery), isolating non-retrieval failures (planning, reasoning, execution). *Trimmed*: to respect practical constraints, most notably the UI limit of $\leq$10 file uploads imposed by the closed-source DR systems, we supply the gold files plus a random subset of distractors up to the limit, testing discovery under budget. *Oracle**: for tasks where the gold set itself exceeds 10 files, we include the task by randomly sampling 10 gold files for upload.

Control flow describes whether a system has fully structured loops or an agentic workflow where agents decide the future courses of actions. Internet access describes whether systems have web search capabilities.

Cost of evaluation. For DS-Guru (few-shot, GPT-o3), evaluating end-to-end answers, pipeline design, and sub-task implementation took 4,501, 116,805, and 10,358 tokens respectively.

4 RESULTS AND TAKEAWAYS

Table 5 shows the performance of the systems under the *Full*, *Oracle*, and *Trimmed* input mode. We report only top-performing configurations here and present full results in Appendix A.

Agentic control flows drive the largest performance gains on KRAMABENCH. smolagents DR (Claude-3-7, max agentic iterations is 20) consistently outperforms DS-Guru across all domains (Table 9), reaching 50% overall score compared to the best DS-Guru variant (few-shot, GPT-o3; 22.08%). The DS-Guru (few-shot), which enables the LLM to catch implementation errors only moderately improves over DS-Guru (one-shot, GPT-o3), with 1.28% overall improvement. Our

Table 5: Results by domain for **KRAMABENCH** on DS-Guru and smolagents DR variations under three settings.

System	Models	Domains: Archeology	Astronomy	Biomedical	Environment	Legal	Wildfire	**Overall**
		Full **Input Mode**						
Human baseline		66.67%	54.55%	100.00%	81.91%	74.19%	58.00%	71.07%
DS-Guru no-context	GPT-o3	25%	1.73%	3.50%	1.35%	3.35%	24.87%	9.64%
	GPT-4o	0.00%	1.41%	1.98%	0.45%	1.46%	1.45%	1.62%
	Claude-3-5	16.67%	1.62%	2.87%	1.17%	7.33%	13.63%	7.45%
DS-Guru one-shot	GPT-o3	25%	3.00%	8.63%	7.66%	19.15%	45.95%	20.80%
	GPT-4o	8.33%	1.40%	9.38%	2.60%	2.74%	19.39%	7.61%
	Claude-3-5	0.00%	4.15%	2.15%	6.21%	6.68%	34.99%	10.85%
DS-Guru few-shot	GPT-o3	25%	3.53%	8.95%	19.6%	13.89%	50.73%	22.08%
	GPT-4o	16.67%	2.76%	8.97%	2.60%	2.80%	17.18%	8.28%
	Claude-3-5	16.67%	1.52%	1.96%	11.21%	7.01%	39.16%	14.35%
smolagents-single DR	GPT-o3	**41.67%**	**16.67%**	33.33%	50%	50%	38.1%	41.36%
	GPT-4o	33.33%	0.00%	11.11%	35%	40%	38.1%	30.77%
	Claude-3-5	33.33%	0.00%	22.22%	60%	46.67%	**52.38%**	41.35%
	Claude-3-7	33.33%	**18.60%**	20.14%	**60.64%**	48.13%	59.67%	47.23%
smolagents-reflexion DR	Claude-3-7	**41.67%**	5.97%	**42.32%**	59.05%	59.27%	**60.26%**	**50.64%**
	GPTo3	16.67%	13.44%	15.26%	3.04%	14.25%	29.25%	14.69%
smolagents-pdt DR	Claude-3-7	25.00%	10.08%	2.22%	6.00%	10.22%	36.98%	15.92%
	GPTo3	16.67%	2.46%	4.13%	0.68%	6.87%	26.50%	10.17%
		Oracle **Input Mode**						
DS-Guru no-context	GPT-o3	17.83%	12.93%	19.48%	19.17%	9.94%	16.13%	14.93%
	GPT-4o	15.09%	9.15%	12.16%	11.26%	8.88%	7.15%	10.05%
	Claude-3-5	16.52%	10.63%	9.87%	12.51%	9.80%	0.00%	11.63%
DS-Guru one-shot	GPT-o3	23.90%	21.14%	18.29%	28.48%	18.49%	25.08%	22.85%
	GPT-4o	14.26%	10.58%	9.38%	20.37%	10.96%	19.21%	14.86%
	Claude-3-5	17.07%	10.24%	9.44%	22.27%	11.47%	17.93%	15.48%
DS-Guru few-shot	GPT-o3	27.78%	23.22%	19.56%	33.67%	35.14%	32.53%	31.92%
	GPT-4o	18.97%	19.29%	12.51%	27.14%	25.23%	26.07%	23.60%
	Claude-3-5	16.24%	14.02%	14.80%	33.83%	26.36%	25.02%	24.22%
smolagents-single DR	GPT-o3	**41.67%**	25%	44.44%	45%	44.83%	47.62%	44.45%
	GPT-4o	25%	25%	22.22%	20%	56.67%	38.1%	39%
	Claude-3-5	16.67%	25%	33.33%	25%	**66.66%**	66.66%	47%
	Claude-3-7	**41.67%**	**33.33%**	**77.78%**	**80%**	63.33%	**71.43%**	**59%**
smolagents-pdt DR	GPTo3	16.67%	0.40%	2.22%	2.66%	11.12%	29.00%	11.96%
		Trimmed **Input Mode**						
DS-Guru few-shot	GPT-o3	25.00%	3.17%	2.71%	17.02%	16.25%	49.42%	21.78%
smolagents-single DR	Claude-3-7	33.33%	**33.33%**	44.44%	**65%**	**63.33%**	**66.67%**	**57.85%**
OpenAI DR	GPT-o3-dr	**40%**	**33.33%**	**44.45%**	61.67%	50%	67.28%	52.18%
Gemini Agentic	Gemini-2.5-Pro	25%	16.67%	33.33%	25%	13.33%	24.87%	18.48%

detailed studies increased few-shot to 20 iterations yet still showed minor improvements (Appendix Table 13). This indicates that despite the heterogeneity of data operations, the core challenges are not isolated data operation implementation issues, but instead are to (1) explore and fix the design choices of the end-to-end pipeline; (2) iteratively understand the data and schema in a large data lake. Smolagent DR's agentic control flow helps address these challenges. Note that in the *Trimmed* setting (max 10 files per call), OpenAI DR reaches 52.18% overall, partly due to its web search capability. We refer the reader to Appendix F for detailed analysis of the human baseline.

In terms of cost, smolagents DR (Claude-3-7) averages 6.10 minutes per task—faster than OpenAI DR (10.35) but more than 10× slower than DS-Guru few-shot (0.76).

4.1 ABLATION STUDIES

Retrieval Mechanisms. Using *Oracle* input for DS-Guru improves the performance for the overall dataset across all domains and LLMs (by 9.98% on average and up to 20%) except for GPT-o3, Claude-3.5, and DeepSeek-R1 on wildfire (Table 5). These results under the design of DS-Guru show that supplying samples of the gold files can lead to more successful pipelines. We also studied the sensitivity of *OPS* against the sample size from each file. Table 6 shows that the performance of the system does not meaningfully increase with larger samples.

The benefits of the *Oracle* in smolagents DR shows the same trend (improvements of around 10%), suggesting that *agentic retrieval is not qualitatively closer to perfect retrieval than OPS in terms of* *file extraction*. Even with the *Oracle* input, the agentic smolagents DR with out-of-the-box LLMs still struggle to solve a lot of the tasks (59% overall with Claude-3.7). These results point to weaknesses in data-dependent reasoning (e.g., pipeline design), in addition to extracting the right files.

Table 6: DS-Guru (few-shot, 5 iterations, GPT-o3): performance and cost across rows sampled.

Rows Sampled	**10**	**50**	**100**	**150**
Overall Performance (%)	22.89	24.68	23.36	22.58
Tokens (Mean)	14,077.2	37,592.3	64,548.9	92,116.1

Table 7: End-to-end scores of various systems under obscured vs oracle inputs over the same tasks. Note that we sampled a subset of legal and wildfire respectively to curate obscured inputs for.

System	**Models**	**Combined**			**Legal**			**Wildfire**		
		Full	Oracle	Obscured	Full	Oracle	Obscured	Full	Oracle	Obscured
DS-Guru no-context	GPT-o3	12.54%	10.72%	11.15%	5.08%	6.45%	9.46%	20.00%	15.00%	13.07%
	GPT-4o	7.19%	2.52%	8.60%	4.37%	5.04%	9.83%	10.00%	0.002%	7.21%
	Claude-3-5	8.50%	4.85%	9.93%	6.99%	4.64%	11.30%	10.00%	0.00%	8.37%
DS-Guru one-shot	GPT-o3	12.73%	26.98%	8.99%	15.47%	24.89%	11.62%	10.00%	29.08%	5.99%
	GPT-4o	26.03%	27.45%	11.15%	12.06%	14.89%	10.27%	40.00%	40.00%	12.16%
	Claude-3-5	19.56%	18.82%	6.11%	5.03%	8.56%	7.35%	34.08%	29.07%	4.70%
DS-Guru few-shot	GPT-o3	7.08%	44.74%	20.40%	4.18%	40.41%	20.29%	10.00%	49.08%	20.52%
	GPT-4o	21.92%	34.21%	11.90%	8.85%	28.41%	15.47%	35.00%	40.00%	7.82%
	Claude-3-5	25.06%	23.02%	13.46%	6.04%	16.98%	12.18%	44.08%	29.07%	14.92%
Smolagents-single DR	GPT-o3	40%	45%	20%	50%	30%	30%	30%	60%	10%
	GPT-4o	50%	40%	30%	50%	40%	30%	50%	40%	30%
	Claude-3-5	55%	60%	30%	40%	50%	20%	70%	70%	40%

Data Leakage. To study to what extent different systems are solving tasks via external knowledge present in previous knowledge data instead of producing a reliable data pipeline, we manually curated **obscured inputs** for 20% of tasks in **KRAMABENCH**, where some data fields are changed such that a correct pipeline would still produce a correct solution, but a system relying on memorization cannot. For example, in a query spanning multiple locations, the real place names may be swapped for fictional ones, i.e., Los Angeles might be changed to “La-La Land."

For both smolagents-single DR and DS-Guru few-shot, the performance under the obscured input is 12-16% lower compared to the oracle input (Table 7). Interestingly, compared to *Full* input, *Obscured* input improved the performance for DS-Guru but significantly degraded smolagent for the legal workload. These observations and the stark difference between the *Full* and *Obscured* input performances on wildfire suggest two distinctive plausible explanations for our observations: (1) Prior knowledge could discourage attempts at data-dependent reasoning. (2) Prior knowledge could be serving as an unintended reward signal in agentic data-dependent reasoning, which possibly can either improve or reduce the performance of the system.

Cross-domain Accuracy Difference The per-domain accuracy of the best performing system (smolagents-reflexion DR) in **KRAMABENCH** varies as much as 33.33% on Astronomy and 80% on Environment. Our analysis of system traces show that the primary source of these cross-domain accuracy differences is the differences in the types of data task challenges that each domain emphasize on. We discuss this in more detail in Subsection D.1.

Diversity of Abilities Required from LLM Agents. Tested independently, both pipeline design (+19.50% GPT-o3) and sub-task implementation (+5.39% GPT-4o) substantially outperformed end-to-end automation. In addition, we observe that LLMs also have varying capability profiles: GPT-o3

Table 8: Lower automation settings evaluation results for **KRAMABENCH** on 18 methods.

Variant	**Automation setting**	**Models**					
		GPT-o3	GPT-4o	Claude-3.5	Llama3-3Instruct	DeepSeek-R1	Qwen2-5Coder
DS-GURU no-context	End-to-end automation	9.64%	1.62%	7.45%	1.19%	3.14%	3.72%
	Pipeline Design	40.60%	30.83%	31.06%	26.74%	18.94%	27.35%
	Sub-task Implementation	12.95%	9.27%	10.65%	8.28%	12.08%	7.52%
DS-GURU one-shot	End-to-end automation	20.80%	7.61%	10.85%	4.81%	6.35%	6.43%
	Pipeline Design	42.14%	19.75%	25.49%	19.24%	10.60%	22.19%
	Sub-task Implementation	17.24%	11.42%	10.12%	7.83%	11.37%	10.38%
DS-GURU few-shot	End-to-end automation	22.08%	8.28%	14.35%	4.48%	6.35%	9.98%
	Pipeline Design	41.58%	16.67%	29.46%	16.83%	6.44%	14.65%
	Sub-task Implementation	19.75%	13.67%	16.14%	8.87%	10.89%	12.09%

is strong at high-level pipeline design (42%) but weak at implementing those pipelines (20%); interestingly, it scores higher on end-to-end automation (22%) than on some implementation tasks. DeepSeek-R1 exhibits the opposite pattern (6.5% on pipeline design vs 11% on implementation). These patterns provide strong evidence that single-agent approaches are insufficiently reliable for real-world data science, as success depends on multiple heterogeneous skills, such as robust parsing of noisy inputs, query parsing and planning, identifying and performing data-cleaning/ transformation, coding, and iterative debugging.

4.2 Deeper Dive: Failure Analysis

In this subsection, we closely study two tasks requiring two distinct reasoning capabilities from LLM agents: (1) fine-grained data-dependent reasoning. (2) holistic understanding of a potentially domain-specific data lake.

Challenge 1: Fine-grained data-dependent reasoning.

Monthly Precipitation			
Year	Jan	Feb	Mar
2015	4.4	3.9	4.7
2016	5.0	6.2	3.5
2017	4.5	M	M

Water Body Testing			
Community	Sample Date	Beach Name	Violation
Chatham	2016-06-13	Bucks Creek	no
Brewster	2016-08-16	Cliff Pond (DCR) @ DYS	yes
Brewster	2016-05-24	Cliff Pond (DCR) @ Main	no

Figure 5: Data snippets for study cases. Multiple water testing entries for each location may exist.

environment-q17: What is the seasonal bacteria exceedance rate of Chatham's Bucks Creek Beach in the June, July, Aug of 2016? Impute missing values with median of the month in non-missing years.

To solve this query, a correct pipeline must analyze the data present in both files in Figure 5. DS-Guru uses *OPS* sampling, which may not see or realize the "M" buried in the data and deduce that "M" stands for missing values. Although few-shot prompting enables the agent to see relevant errors, the lack of an explicit agentic control flow results in the LLM not connecting the execution errors to these fine-grained data observations. By contrast, on every agentic iteration, smolagents DR conjectures what the important data are to look at next to ensure the correctness of the pipeline it has drafted. This conjecture guides its tool call-enabled retrieval step. It subsequently analyzes the tool call and pipeline execution results before the next iteration. This explicit *retrieve-revise-repeat* pattern tightly couples error feedbacks with data retrieval, which helps address the fine-grained data-dependent reasoning challenge and leads to working end-to-end pipelines.

Challenge 2: Holistic understanding of the input data and prior knowledge.

environment-q16: How many beaches remained safe to swimming from 2002 to 2023 inclusive?

environment-q16-3: How many beaches are there?

environment-q16-3 is an example sub-task for *environment-q16*, which also uses files in Figure 5. To solve the full task reliably, a system should be able to identify all beaches to start with. *environment-q16-3* prompts the system to carryout this identification and verifies the result.

The challenge with beach identification is that the "Beach Name" column encodes both the beach and sampling location. Cliff Pond (DCR) @ Main refers to the Main (street) sampling location of the Cliff Pond beach (Figure 5). Facing many near-duplicate files in the data lake, systems do not have a clear global schema or geographical domain knowledge that they could use to understand this encoding scheme. As a result, both DS-Guru and smolagents DR failed on this sub-task, despite smolagents DR's agentic control flow. This case highlights the need to incorporate prior knowledge and discover clarifications about under-specified conventions *from the data* (Mao et al., 2019).

Towards this end, we analyzed the traces of DS-Guru (few-shot, GPT-o3 & Claude 3.5) with the agentic system diagnosis framework proposed in Cemri et al. (2025). Respectively 24% (GPT-o3) and 43% (Claude 3.5) of all 104 tasks suffer from "failure to ask for clarification" and thus were not solved correctly. However, **KramaBench** expects that a human expert could solve the tasks without additional clarifications by exploring and understanding the data. In this example, a human could decipher the beach name conventions with common U.S. geographical knowledge. Reasoning models currently lack similar capabilities to gain holistic understandings of the input data and fail to incorporate prior knowledge.

5 Related Work

LLM-Powered Agentic Systems. There is a large and fast-growing literature on LLM-powered AI systems. These systems take on vastly different designs, such as vanilla LLM calls to frontier pre-trained reasoning models (OpenAI et al., 2024; DeepSeek-AI et al., 2025; Zhong et al., 2024), retrieval-augmented generation (Lewis et al., 2020), agentic workflow systems (Zhang et al., 2025b), chain-of-thought and iterative calls (Wei et al., 2022; Press et al., 2023), reflections (Ji et al., 2023) and task-time verifications (Tang et al., 2024a), structured knowledge representations (Jiang et al., 2024; Su et al., 2025; Wang et al., 2025), and data processing centric systems (Liu et al., 2024; Patel et al., 2025; Shankar et al., 2024). Recent work applies these techniques to data science tasks. For example, DocWrangler (Shankar et al., 2025) is an integrated development environment that helps the user optimize LLM prompts to construct data processing programs. DSAgent (Guo et al., 2024) is a framework that uses LLMs to understand user needs and build data science pipelines. Evaporate (Arora et al., 2023) helps users transform data into queryable tables. AutoPrep (Fan et al., 2025) constructs a data preparation program over a single table for a given question. Despite the progress, evaluating agent performance in real-world end-to-end setting remains a challenge.

Evaluations of LLM-Powered Agentic Systems. Benchmarks for question answering (QA) have shifted toward evaluating agentic solutions. These benchmarks require iterative retrieval, query parsing, planning, tool use, and temporal awareness. Recent works include FanOutQA (Zhu et al., 2024), MultiHop-RAG (Tang & Yang, 2024), CRAG (Yang et al., 2024), BrowseComp (Wei et al., 2025), which test end-to-end retrieval systems, MEQA (Li et al., 2024) for multi-hop reasoning with explanation chains, and MINTQA (He et al., 2024) for scaffolding long knowledge. These tasks differ from data science tasks, as they only require information retrieval and joins, but no data-intensive processing. Benchmarks such as DS-1000 (Lai et al., 2023), DA-Code (Huang et al., 2024), ARCADE (Yin et al., 2023), DataSciBench (Zhang et al., 2025a), DSEval (Zhang et al., 2024c) focus instead on implementing detailed instructions in general programming languages, specifically in data science tasks, differentiating themselves from other benchmarks like SWE-Bench (Jimenez et al., 2024), ML-Bench (Tang et al., 2024b), BigCodeBench (Zhuo et al., 2025). More recently, new benchmarks such as DSBench (Jing et al., 2025) and BLADE (Gu et al., 2024) have started to evaluate the ability to create an implementation plan. Benchmarks like ScienceAgentBench (Chen et al., 2025) and BixBench (Mitchener et al., 2025) evaluate using domain knowledge. Although such benchmarks assess specific capabilities, they fall short of capturing the full complexity of real-world data science pipelines.

6 Conclusion

KramaBench evaluates the capabilities of systems to generate data science pipelines over a data lake consisting of heterogeneous, unclean input. Our comprehensive experiments using 8 LLMs across 4 different agentic systems with **KramaBench** reveals although current systems are equipped with useful techniques such as agentic control flow and generic coding abilities, they are still far from solving real-world data science problems. Our analyses highlight several underexplored challenges such as effective retrieval, data-dependent reasoning, plan revision, and robust prior/domain knowledge integration as meaningful research directions towards practical automated data science systems.

Ethics Statement

We acknowledge the limitations of **KramaBench** regarding its scope, language, and cultural biases, and domain coverage, which stem from the human effort required for high-quality curation. All data included is publicly available, anonymized, or pseudonymized, with no personally identifiable information. The biomedical domain contains public data sourced from the cancer data commons

(CDC) – this data is pseudonymized and does not require confidential access nor specific approvals, with the only sensitive attribute included as part of the workload being the pseudonymized age of patients. We emphasize privacy as paramount and warn users of the benchmark against potential identification risks, which we deem unlikely, associated with this data source. In future iterations of our benchmark we aim at broaden domain diversity, include multilingual data, and integrate community contributions to reduce existing biases Furthermore, we comply with licensing practices of data sources. For data sources that are publicly available but have redistribution constraints, we do not modify or separately host these datasets. Instead, we point users of our benchmark to the original data sources.

Reproducibility Statement

We provide full artifacts—including code, data, workloads, and evaluation scripts—via our public repository at https://anonymous.4open.science/r/Kramabench-7D6D/. The main paper section 2 and Appendix D describe the process obtained to design and curate the task based on the datasets for each domain. Scripts to reproduce these steps can be found in the main repository. All datasets, benchmark frameworks, benchmark curation semi-automation scripts, reference pipelines and other accompanying annotations, and our reference system DS-Guru are available in our repository. The experimental analysis of different system under test in Section 4 can be reproduced using Python scripts also available in the public repository.

References

smolagents/examples/open_deep_research at main · huggingface/smolagents — github.com. https://github.com/huggingface/smolagents/tree/main/examples/open_deep_research. [Accessed 01-12-2025].

Magentic-One: A Generalist Multi-Agent System for Solving Complex Tasks - Microsoft Research — microsoft.com. https://www.microsoft.com/en-us/research/articles/magentic-one-a-generalist-multi-agent-system-for-solving-complex-tasks/. [Accessed 01-12-2025].

Alibaba DAMO Academy. Deepresearcher: An open-source multi-agent system for deep research. https://github.com/alibaba/DeepResearcher, 2025. Accessed: YYYY-MM-DD.

Anthropic. Introducing claude 3.5 sonnet. https://www.anthropic.com/news/claude-3-5-sonnet, 2024. Accessed: 2025-09-24.

Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. Language models enable simple systems for generating structured views of heterogeneous data lakes. *PVLDB*, 17(2):92–105, October 2023. ISSN 2150-8097. doi: 10.14778/3626292.3626294. URL https://doi.org/10.14778/3626292.3626294.

Julia Briden, Peng Mun Siew, Victor Rodriguez-Fernandez, and Richard Linares. Transformer-based atmospheric density forecasting. *Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference*, 2023. Free preprint available at https://arxiv.org/abs/2310.16912.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya G. Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? *CoRR*, abs/2503.13657, 2025. doi: 10.48550/ARXIV.2503.13657. URL https://doi.org/10.48550/arXiv.2503.13657.

National Interagency Fire Center. Statistics| National Interagency Fire Center — nifc.gov. https://www.nifc.gov/fire-information/statistics. [Accessed 21-05-2025].

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun. Scienceagentbench:

Toward rigorous assessment of language agents for data-driven scientific discovery, 2025. URL https://openreview.net/forum?id=6z4YKr0GK6.

F. Clette and L. Lefèvre. Silso sunspot number v2.0. https://doi.org/10.24414/qnza-ac80, 07 2015. Published by WDC SILSO - Royal Observatory of Belgium (ROB).

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, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Yongchao Dou, Emily A Kawaler, Daniel Cui Zhou, Marina A Gritsenko, Chen Huang, Lili Blumenberg, Alla Karpova, Vladislav A Petyuk, Sara R Savage, Shankha Satpathy, et al. Proteogenomic characterization of endometrial carcinoma. *Cell*, 180(4):729–748, 2020.

European Space Agency. Swarm Satellite Mission Data. https://earth.esa.int/eogateway/missions/swarm/data, 2013. Accessed: 2025-05-06.

Meihao Fan, Ju Fan, Nan Tang, Lei Cao, Guoliang Li, and Xiaoyong Du. Autoprep: Natural language question-aware data preparation with a multi-agent framework, 2025. URL https://arxiv.org/abs/2412.10422.

Federal Trade Commission. Age and Fraud Dashboard. https://public.tableau.com/app/profile/federal.trade.commission/viz/AgeandFraud/Infographic, 2025a. Accessed: 2025-05-06.

Federal Trade Commission. FTC Open Government Data Sets. https://www.ftc.gov/policy-notices/open-government/data-sets, 2025b. Accessed: 2025-05-06.

Federal Trade Commission. Debt Collection Dashboard. https://public.tableau.com/app/profile/federal.trade.commission/viz/DebtCollection/Infographic, 2025c. Accessed: 2025-05-06.

Raphael Fontes. Us election 2020 dataset. https://www.kaggle.com/datasets/unanimad/us-election-2020, 2020. Accessed: 2025-05-06.

Michael A Gillette, Shankha Satpathy, Song Cao, Saravana M Dhanasekaran, Suhas V Vasaikar, Karsten Krug, Francesca Petralia, Yize Li, Wen-Wei Liang, Boris Reva, et al. Proteogenomic characterization reveals therapeutic vulnerabilities in lung adenocarcinoma. *Cell*, 182(1):200–225, 2020.

Google. Agent mode in gemini code assist. https://developers.google.com/gemini-code-assist/docs/agent-mode, 2025. Accessed: 2025-09-24.

Google DeepMind. Gemini 2.5 pro. https://deepmind.google/models/gemini/pro/, 2025. Accessed: 2025-09-24.

Huw S Groucutt, Tom S White, Eleanor ML Scerri, Eric Andrieux, Richard Clark-Wilson, Paul S Breeze, Simon J Armitage, Mathew Stewart, Nick Drake, Julien Louys, et al. Multiple hominin dispersals into southwest asia over the past 400,000 years. *Nature*, 597(7876):376–380, 2021.

Ken Gu, Ruoxi Shang, Ruien Jiang, Keying Kuang, Richard-John Lin, Donghe Lyu, Yue Mao, Youran Pan, Teng Wu, Jiaqian Yu, Yikun Zhang, Tianmai M. Zhang, Lanyi Zhu, Mike A Merrill, Jeffrey Heer, and Tim Althoff. BLADE: Benchmarking language model agents for data-driven science, November 2024. URL https://aclanthology.org/2024.findings-emnlp.815/.

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. 2024.

Jie He, Nan Hu, Wanqiu Long, Jiaoyan Chen, and Jeff Z. Pan. Mintqa: A multi-hop question answering benchmark for evaluating llms on new and tail knowledge, 2024. URL https://arxiv.org/abs/2412.17032.

Yiming Huang, Jianwen Luo, Yan Yu, Yitong Zhang, Fangyu Lei, Yifan Wei, Shizhu He, Lifu Huang, Xiao Liu, Jun Zhao, and Kang Liu. Da-code: Agent data science code generation benchmark for large language models, 2024. URL https://doi.org/10.18653/v1/2024.emnlp-main.748.

Hugging Face. Open deep research — smolagents/examples/open_deep_research. https://github.com/huggingface/smolagents/tree/main/examples/open_deep_research, 2025. Accessed: 2025-09-24.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-Coder Technical Report. *arXiv preprint arXiv:2409.12186*, 2024. URL https://arxiv.org/abs/2409.12186.

Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. Towards mitigating LLM hallucination via self reflection. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1827–1843, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.123. URL https://aclanthology.org/2023.findings-emnlp.123/.

Zhouyu Jiang, Ling Zhong, Mengshu Sun, Jun Xu, Rui Sun, Hui Cai, Shuhan Luo, and Zhiqiang Zhang. Efficient knowledge infusion via KG-LLM alignment. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics*, pp. 2986–2999, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.176. URL https://aclanthology.org/2024.findings-acl.176/.

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?, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

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?, 2025. URL https://arxiv.org/abs/2409.07703.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-tau Yih, Daniel Fried, Sida Wang, and Tao Yu. Ds-1000: a natural and reliable benchmark for data science code generation, 2023.

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. 2025. URL https://openreview.net/forum?id=XmProj9cPs.

Quentin Lemesle, Jonathan Chevelu, Philippe Martin, Damien Lolive, Arnaud Delhay, and Nelly Barbot. Paraphrase generation evaluation powered by an LLM: A semantic metric, not a lexical one. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the International Conference on Computational Linguistics*, pp. 8057–8087, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.538/.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2020.

Ruosen Li, Zimu Wang, Son Quoc Tran, Lei Xia, and Xinya Du. Meqa: A benchmark for multi-hop event-centric question answering with explanations. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024), Datasets and Benchmarks Track*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/e560a0b22e4432003d0dba63ff8dc457-Abstract-Datasets_and_Benchmarks_Track.html.

Chunwei Liu, Matthew Russo, Michael Cafarella, Lei Cao, Peter Baille Chen, Zui Chen, Michael Franklin, Tim Kraska, Samuel Madden, and Gerardo Vitagliano. A declarative system for optimizing ai workloads, 2024. URL https://arxiv.org/abs/2405.14696.

Yaoli Mao, Dakuo Wang, Michael Muller, Kush R. Varshney, Ioana Baldini, Casey Dugan, and Aleksandra Mojsilović. How data scientistswork together with domain experts in scientific collaborations: To find the right answer or to ask the right question? *Proc. ACM Hum.-Comput. Interact.*, 3(GROUP), December 2019. doi: 10.1145/3361118. URL https://doi.org/10.1145/3361118.

Massachusetts Department of Public Health. Water Quality at Massachusetts Swimming Beaches. https://www.mass.gov/lists/water-quality-at-massachusetts-swimming-beaches, 2025a. Accessed: 2025-05-06.

Massachusetts Department of Public Health. Water Body Testing Report — Massachusetts Environmental Public Health Tracking Network (MEPHTN). https://dphanalytics.hhs.mass.gov/ibmcognos/bi/?perspective=authoring&pathRef=.public_folders%2FMEPHTN%2Fenvironmental%2Fwater-body-testing&id=iB8503D8E63864870AC33EF393D858EB2, 2025b. Accessed: 2025-05-06.

Massachusetts Water Resources Authority. Beach Fact Sheets — Massachusetts Water Resources Authority (MWRA). https://www.mwra.com/harbor/download-environmental-data#beach-fact-sheets, 2025a. Accessed: 2025-05-06.

Massachusetts Water Resources Authority. Download Environmental Data — Massachusetts Water Resources Authority (MWRA). https://www.mwra.com/harbor/download-environmental-data, 2025b. Accessed: 2025-05-06.

Meta AI. Llama 3.3: Model cards and prompt formats. https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_3/, 2025. Accessed: 2025-09-24.

Ludovico Mitchener, Jon M Laurent, Benjamin Tenmann, Siddharth Narayanan, Geemi P Wellawatte, Andrew White, Lorenzo Sani, and Samuel G Rodriques. Bixbench: a comprehensive benchmark for llm-based agents in computational biology, 2025. URL https://arxiv.org/abs/2503.00096.

Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. Using an llm to help with code understanding. In *Proceedings of the International Conference on Software Engineering (ICSE)*, pp. 1–13, 2024.

National Centers for Environmental Information (NCEI). Wildfires - National Centers for Environmental Information (NCEI). https://www.ncei.noaa.gov/access/monitoring/wildfires/, 2025. Accessed: 2025-05-06.

National Interagency Fire Center. Fire Information — National Interagency Fire Center (NIFC). https://www.nifc.gov/fire-information, 2025. Accessed: 2025-05-06.

National Weather Service. Climate Data for Boston (BOX) Office — National Weather Service. https://www.weather.gov/wrh/Climate?wfo=box, 2025. Accessed: 2025-05-06.

NCEI.Monitoring.Info@noaa.gov. Monthly Climate Reports | National Centers for Environmental Information (NCEI) — ncei.noaa.gov. https://www.ncei.noaa.gov/access/monitoring/monthly-report/fire, 2025. [Accessed 21-05-2025].

NOAA Office of Satellite and Product Operations. NOAA Geostationary Operational Environmental Satellite (GOES) I-M and N-P Series Imager Data. NOAA National Centers for Environmental Information, 1994. URL https://doi.org/10.25921/Z9JQ-K976. Accessed: 2025-05-06.

OpenAI. Hello gpt-4o. https://openai.com/index/hello-gpt-4o/, 2024. Accessed: 2025-09-24.

OpenAI. Introducing deep research. https://openai.com/index/introducing-deep-research/, 2025. [Accessed 13-05-2025].

OpenAI. Introducing openai o3 and o4-mini. https://openai.com/index/introducing-o3-and-o4-mini/, 2025. Accessed: 2025-09-24.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg

Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

Natalia E. Papitashvili and Joseph H. King. Omni daily data. NASA Space Physics Data Facility, 2020a. URL https://doi.org/10.48322/5fmx-hv56. Accessed: 2025-05-06.

Natalia E. Papitashvili and Joseph H. King. Omni hourly data. NASA Space Physics Data Facility, 2020b. URL https://doi.org/10.48322/1shr-ht18. Accessed: 2025-05-06.

William E. Parker and Richard Linares. Satellite drag analysis during the may 2024 gannon geomagnetic storm. *Journal of Spacecraft and Rockets*, 61(5):1412–1416, September 2024. ISSN 1533-6794. doi: 10.2514/1.a36164. URL http://dx.doi.org/10.2514/1.a36164.

Liana Patel, Siddharth Jha, Melissa Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia. Semantic operators: A declarative model for rich, ai-based data processing, 2025. URL https://arxiv.org/abs/2407.11418.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/.

Xiao Pu, Mingqi Gao, and Xiaojun Wan. Summarization is (almost) dead. *arXiv preprint arXiv:2309.09558*, 2023.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, et al. Tool learning with foundation models. *ACM Computing Surveys*, 57(4):1–40, 2024a.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs, 2024b. URL https://openreview.net/forum?id=dHng2O0Jjr.

Robikscube. Zillow home value index (zhvi). https://www.kaggle.com/datasets/robikscube/zillow-home-value-index, 2021. Accessed: 2025-05-06.

Eleanor ML Scerri, James Blinkhorn, Huw S Groucutt, Mathew Stewart, Ian Candy, Ethel Allué, Aitor Burguet-Coca, Andrés Currás, W Christopher Carleton, Susanne Lindauer, et al. Hunter-gatherer sea voyages extended to remotest mediterranean islands. *Nature*, pp. 1–7, 2025.

Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G. Parameswaran, and Eugene Wu. Docetl: Agentic query rewriting and evaluation for complex document processing, 2024. URL https://arxiv.org/abs/2410.12189.

Shreya Shankar, Bhavya Chopra, Mawil Hasan, Stephen Lee, Björn Hartmann, Joseph M. Hellerstein, Aditya G. Parameswaran, and Eugene Wu. Steering semantic data processing with docwrangler. 2025. URL https://arxiv.org/abs/2504.14764.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

C. Siemes, J. de Teixeira da Encarnação, E. N. Doornbos, J. van den IJssel, J. Kraus, R. Pereštý, L. Grunwaldt, G. Apelbaum, J. Flury, and P. E. Holmdahl Olsen. Swarm accelerometer data processing from raw accelerations to thermospheric neutral densities. *Earth, Planets and Space*, 68:92, 2016. doi: 10.1186/s40623-016-0474-5. URL https://doi.org/10.1186/s40623-016-0474-5.

Hanchen Su, Wei Luo, Yashar Mehdad, Wei Han, Elaine Liu, Wayne Zhang, Mia Zhao, and Joy Zhang. LLM-friendly knowledge representation for customer support. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, Steven Schockaert, Kareem Darwish, and Apoorv Agarwal (eds.), *Proceedings of the International Conference on Computational Linguistics: Industry Track*, pp. 496–504, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-industry.42/.

Nan Tang, Chenyu Yang, Ju Fan, Lei Cao, Yuyu Luo, and Alon Y. Halevy. Verifai: Verified generative ai. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, 2024a.

Xiangru Tang, Yuliang Liu, Zefan Cai, Yanjun Shao, Junjie Lu, Yichi Zhang, Zexuan Deng, Helan Hu, Kaikai An, Ruijun Huang, Shuzheng Si, Sheng Chen, Haozhe Zhao, Liang Chen, Yan Wang, Tianyu Liu, Zhiwei Jiang, Baobao Chang, Yin Fang, Yujia Qin, Wangchunshu Zhou, Yilun Zhao, Arman Cohan, and Mark Gerstein. Ml-bench: Evaluating large language models and agents for machine learning tasks on repository-level code, 2024b. URL https://arxiv.org/abs/2311.09835.

Yixuan Tang and Yi Yang. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. In *Proceedings of the 1st Conference on Language Modeling (COLM)*, Philadelphia, PA, USA, 2024. URL https://openreview.net/forum?id=t4eB3zYWBK. COLM 2024.

Weixi Tong and Tianyi Zhang. CodeJudge: Evaluating code generation with large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 20032–20051, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1118. URL https://aclanthology.org/2024.emnlp-main.1118/.

U.S. Air Force and NOAA Space Weather Prediction Center. USAF 45-Day Ap and F10.7cm Flux Forecast. https://www.swpc.noaa.gov/products/usaf-45-day-ap-and-f107cm-flux-forecast, 2025. Accessed: 2025-05-06.

U.S. Census Bureau. National Population Totals: 2020–2023. https://www.census.gov/data/tables/time-series/demo/popest/2020s-national-total.html, 2025. Accessed: 2025-05-06.

U.S. Environmental Protection Agency. Air Quality System (AQS) Annual Data Download. https://aqs.epa.gov/aqsweb/airdata/download_files.html#Annual, 2025. Accessed: 2025-05-06.

U.S. Space Command. Two-line element sets (tles) from space-track.org. https://www.space-track.org/, 2025. Accessed: 2025-05-06.

Jianxun Wang and Yixiang Chen. A review on code generation with llms: Application and evaluation. In *IEEE International Conference on Medical Artificial Intelligence (MedAI)*, pp. 284–289. IEEE, 2023.

Xi Wang, Taketomo Isazawa, Liana Mikaelyan, and James Hensman. KBLam: Knowledge base augmented language model. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=aLsMzkTej9.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

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. *arXiv preprint arXiv:2504.12516*, 2025.

Wikipedia contributors. Metropolitan statistical area — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Metropolitan_statistical_area, 2025. Accessed: 2025-05-06.

Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, Lingkun Kong, Brian Moran, Jiaqi Wang, Yifan Ethan Xu, An Yan, Chenyu Yang, Eting Yuan, Hanwen Zha, Nan Tang, Lei Chen, Nicolas Scheffer, Yue Liu, Nirav Shah, Rakesh Wanga, Anuj Kumar, Wen-tau Yih, and Xin Luna Dong. Crag – comprehensive rag benchmark. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024), Datasets and Benchmarks Track*, 2024. doi: 10.48550/arXiv.2406.04744. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/1435d2d0fca85a84d83ddcb754f58c29-Abstract-Datasets_and_Benchmarks_Track.html. See arXiv:2406.04744 for the full version.

Pengcheng Yin, Wen-Ding Li, Kefan Xiao, Abhishek Rao, Yeming Wen, Kensen Shi, Joshua Howland, Paige Bailey, Michele Catasta, Henryk Michalewski, Oleksandr Polozov, and Charles Sutton. Natural language to code generation in interactive data science notebooks, July 2023. URL https://aclanthology.org/2023.acl-long.9/.

Jesse D. Young, Alexander M. Evans, Jose M. Iniguez, Andrea Thode, Marc D. Meyer, Shaula J. Hedwall, Sarah M. McCaffrey, Patrick Shin, and Ching-Hsun Huang. Large wildfire incident status summary (ics-209) report-generated data for the western united states, 2002–2016. Forest Service Research Data Archive, Fort Collins, CO, 2021. URL https://doi.org/10.2737/RDS-2021-0100.

Bin Zhang, Yuxiao Ye, Guoqing Du, Xiaoru Hu, Zhishuai Li, Sun Yang, Chi Harold Liu, Rui Zhao, Ziyue Li, and Hangyu Mao. Benchmarking the text-to-sql capability of large language models: A comprehensive evaluation. *arXiv preprint arXiv:2403.02951*, 2024a.

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, 2025a. URL https://arxiv.org/abs/2502.13897.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation, 2025b. URL https://openreview.net/forum?id=z5uVAKwmjf.

Yang Zhang, Hanlei Jin, Dan Meng, Jun Wang, and Jinghua Tan. A comprehensive survey on process-oriented automatic text summarization with exploration of llm-based methods. *arXiv preprint arXiv:2403.02901*, 2024b.

Yuge Zhang, Qiyang Jiang, XingyuHan XingyuHan, Nan Chen, Yuqing Yang, and Kan Ren. Benchmarking data science agents, August 2024c. URL https://aclanthology.org/2024.acl-long.308/.

Tianyang Zhong, Zhengliang Liu, Yi Pan, Yutong Zhang, Yifan Zhou, Shizhe Liang, Zihao Wu, Yanjun Lyu, Peng Shu, Xiaowei Yu, Chao Cao, Hanqi Jiang, Hanxu Chen, Yiwei Li, Junhao Chen, Huawen Hu, Yihen Liu, Huaqin Zhao, Shaochen Xu, Haixing Dai, Lin Zhao, Ruidong Zhang, Wei Zhao, Zhenyuan Yang, Jingyuan Chen, Peilong Wang, Wei Ruan, Hui Wang, Huan Zhao, Jing Zhang, Yiming Ren, Shihuan Qin, Tong Chen, Jiaxi Li, Arif Hassan Zidan, Afrar Jahin, Minheng Chen, Sichen Xia, Jason Holmes, Yan Zhuang, Jiaqi Wang, Bochen Xu, Weiran Xia, Jichao Yu, Kaibo Tang, Yaxuan Yang, Bolun Sun, Tao Yang, Guoyu Lu, Xianqiao Wang, Lilong Chai, He Li, Jin Lu, Lichao Sun, Xin Zhang, Bao Ge, Xintao Hu, Lian Zhang, Hua Zhou, Lu Zhang, Shu Zhang, Ninghao Liu, Bei Jiang, Linglong Kong, Zhen Xiang, Yudan Ren, Jun Liu, Xi Jiang, Yu Bao, Wei Zhang, Xiang Li, Gang Li, Wei Liu, Dinggang Shen, Andrea Sikora, Xiaoming Zhai, Dajiang Zhu, and Tianming Liu. Evaluation of openai o1: Opportunities and challenges of agi, 2024. URL https://arxiv.org/abs/2409.18486.

Andrew Zhu, Alyssa Hwang, Liam Dugan, and Chris Callison-Burch. FanOutQA: A multi-hop, multi-document question answering benchmark for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 18–37, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-short.2. URL https://aclanthology.org/2024.acl-short.2/.

Terry Yue Zhuo, Vu Minh Chien, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen GONG, James Hoang, Armel Randy Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, Prateek Yadav, Naman Jain, Alex Gu, Zhoujun Cheng, Jiawei Liu, Qian Liu, Zijian Wang, David Lo, Binyuan Hui, Niklas Muennighoff, Daniel Fried, Xiaoning Du, Harm de Vries, and Leandro Von Werra. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. 2025. URL https://openreview.net/forum?id=YrycTjllL0.

A Extended Experiment Results

In this section, we supply the full evaluation results for which we presented a summary of in the main text due to space constraints.

Table 9: Results by domain for **KramaBench** on DS-Guru and smolagents DR with *Full* mode.

System	Models	Domains						
		Archeology	Astronomy	Biomedical	Environment	Legal	Wildfire	**Overall**
DS-Guru no-context	GPT-o3	25%	1.73%	3.50%	1.35%	3.35%	24.87%	9.64%
	GPT-4o	0.00%	1.41%	1.98%	0.45%	1.46%	1.45%	1.62%
	Claude-3.5	16.67%	1.62%	2.87%	1.17%	7.33%	13.63%	7.45%
	Llama3-3Instruct	0.00%	1.43%	1.70%	0.98%	1.37%	1.44%	1.19%
	DeepSeek-R1	0.00%	1.50%	2.49%	2.60%	1.61%	6.46%	3.14%
	Qwen2-5Coder	0.00%	1.37%	2.02%	1.07%	1.44%	13.68%	3.72%
DS-Guru one-shot	GPT-o3	25%	3.00%	8.63%	7.66%	19.15%	45.95%	20.80%
	GPT-4o	8.33%	1.40%	9.38%	2.60%	2.74%	19.39%	7.61%
	Claude-3.5	0.00%	4.15%	2.15%	6.21%	6.68%	34.99%	10.85%
	Llama3-3Instruct	0.00%	1.42%	10.38%	0.98%	5.48%	9.81%	4.81%
	DeepSeek-R1	0.00%	1.57%	3.39%	2.60%	8.30%	14.81%	6.35%
	Qwen2-5Coder	0.00%	1.36%	2.22%	12.59%	1.15%	16.48%	6.43%
DS-Guru few-shot	GPT-o3	25%	3.53%	8.95%	19.6%	13.89%	50.73%	22.08%
	GPT-4o	16.67%	2.76%	8.97%	2.60%	2.80%	17.18%	8.28%
	Claude-3.5	16.67%	1.52%	1.96%	11.21%	7.01%	39.16%	14.35%
	Llama3-3Instruct	0.00%	1.35%	6.98%	0.93%	2.15%	14.49%	4.48%
	DeepSeek-R1	8.33%	2.64%	2.87%	19.08%	8.39%	30.29%	6.34%
	Qwen2-5Coder	8.33%	2.40%	4.35%	12.64%	9.06%	16.48%	9.98%
smolagents DR	GPT-o3	**41.67%**	**16.67%**	33.33%	50%	50%	38.1%	41.36%
	GPT-4o	33.33%	0.00%	11.11%	35%	40%	38.1%	30.77%
	Claude-3-5	33.33%	0.00%	22.22%	**60%**	46.67%	**52.38%**	41.35%
	Claude-3-7	33.33%	**16.67%**	**44.44%**	**60%**	**63.33%**	**52.38%**	**50%**

Table 10: Results by domain for **KramaBench** on DS-Guru and smolagents DR with *Oracle* mode.

System	Models	Domains						
		Archeology	Astronomy	Biomedical	Environment	Legal	Wildfire	**Total**
DS-Guru no-context	GPT-o3	17.83%	12.93%	19.48%	19.17%	9.94%	16.13%	14.93%
	GPT-4o	15.09%	9.15%	12.16%	11.26%	8.88%	7.15%	10.05%
	Claude-3.5	16.52%	10.63%	9.87%	12.51%	9.80%	0.00%	11.63%
	Llama3-3Instruct	14.44%	12.17%	10.24%	10.35%	8.20%	8.06%	9.93%
	DeepSeek-R1	18.79%	8.53%	8.25%	12.71%	11.39%	8.90%	11.56%
	Qwen2-5Coder	10.24%	6.74%	7.71%	7.14%	1.52%	4.53%	6.62%
DS-Guru one-shot	GPT-o3	23.90%	21.14%	18.29%	28.48%	18.49%	25.08%	22.85%
	GPT-4o	14.26%	10.58%	9.38%	20.37%	10.96%	19.21	14.86%
	Claude-3-5	17.07%	10.24%	9.44%	22.27%	11.47%	17.93%	15.48%
	Llama3-3Instruct	8.92%	10.44%	4.45%	12.44%	8.64%	12.90%	10.23%
	DeepSeek-R1	16.78%	15.23%	8.06%	14.23%	11.89%	9.65%	12.64%
	Qwen2-5Coder	9.72%	11.57%	5.37%	15.13%	8.96%	13.22%	11.26%
DS-Guru few-shot	GPT-o3	27.78%	23.22%	19.56%	33.67%	35.14%	32.53%	31.92%
	GPT-4o	18.97%	19.29%	12.51%	27.14%	25.23%	26.07%	23.60%
	Claude-3-5	16.24%	14.02%	14.80%	33.83%	26.36%	25.02%	24.22%
	Llama3-3Instruct	15.57%	13.85%	11.63%	19.37%	15.57%	21.56%	17.11%
	DeepSeek-R1	22.29%	10.79%	9.65%	15.45%	11.75%	10.76%	13.37%
	Qwen2-5Coder	11.83%	14.91%	7.51%	18.39%	13.70%	18.51%	15.15%
smolagents DR	GPT-o3	**41.67%**	25%	44.44%	45%	44.83%	47.62%	44.45%
	GPT-4o	25%	25%	22.22%	20%	56.67%	38.1%	39%
	Claude-3-5	16.67%	25%	33.33%	25%	**66.66%**	66.66%	47%
	Claude-3-7	**41.67%**	**33.33%**	**77.78%**	**80%**	63.33%	**71.43%**	**59%**

B DS-Guru Details

The baseline system we provide, DS-Guru, follows a simple design. For each task, the system provides the backend LLM with an informative sample of data from each file in the data lake first as well as the task prompt. DS-Guru leverages instruction tuning to guide the LLM backend to provide a Python implementation of the task pipeline as well as a structured explanation of the steps to be taken. DS-Guru then executes the implementation and iterate with the LLM pipeline to debug and improve the pipeline by supplying outputs and error messages.

Table 11: Results by domain for **KramaBench** (*Trimmed* input lake). ⋆ marks web-browser on.

System	Metric	Archeology	Astronomy	Biomedical	Environment	Legal	Wildfire	Total
		Domains						
DS-Guru few-shot	Score	25.00%	3.17%	2.71%	17.02%	16.25%	49.42%	21.78%
(GPT-o3)	Avg. runtime/task (min)	0.47	0.49	0.43	0.83	1.44	0.81	0.76
smolagents DR	Claude-3-7	33.33%	**33.33%**	44.44%	**65%**	**63.33%**	**66.67%**	**57.85%**
	Avg. runtime/task (min)	2.22	5.13	40.38	3.72	2.12	2.11	6.10
OpenAI DR⋆	Score	**40%**	**33.33%**	**44.45%**	61.67%	50%	67.28%	52.18%
	Avg. runtime/task (min)	8.105	20.16	10.67	5.3	8.68	12.62	10.35
Gemini 2.5 Pro ⋆	Score	25%	16.67%	33.33%	25%	13.33%	24.87%	18.48%
	Avg. runtime/task (min)	0.64	2.44	3.49	2.3975	3.105	2.314	2.4835

Table 12: Cost-accuracy Tradeoff between different SUTs under *Full* input mode.

SUT	Overall Accuracy	Accuracy/ Runtime	Accuracy / 1k In Tokens	Accuracy / 1k Out Tokens
GPTo3 - Naive	4.4272%	0.0253%	3.6242%	2.0076%
GPTo3 - One Shot	14.3006%	0.0792%	0.3651%	8.0089%
GPTo3 - Few Shot	26.1561%	0.1123%	0.3571%	8.8960%
GPT4o - Naive	1.3532%	0.0081%	1.1068%	1.4968%
GPT4o - One Shot	9.8278%	0.0624%	0.7268%	15.6457%
GPT4o - Few Shot	11.8930%	0.0566%	0.3718%	9.9949%
Llama3_3Instruct - Naive	1.3755%	0.0077%	1.1041%	1.5074%
Llama3_3Instruct - One Shot	5.9167%	0.0313%	0.4362%	10.0508%
Llama3_3Instruct - Few Shot	9.3734%	0.0412%	0.3027%	9.9297%
DeepseekR1 - Naive	3.1111%	0.0546%	3.0126%	1.5262%
DeepseekR1 - One Shot	2.7946%	0.0300%	0.0697%	1.3719%
DeepseekR1 - Few Shot	6.0351%	0.0592%	0.1482%	2.9354%

The prompt used to instruct the LLM backend to provide a pipeline for the end-to-end task is presented below:

B.1 System Prompt

```
You are a helpful assistant that generates a plan to solve
the given request, and you'll be given:Your task is to answer
the following question based on the provided data sources.
Question: {query}
Data file names: {file_names}
The following is a snippet of the data files: {data}
Now think step-by-step carefully.
First, provide a step-by-step reasoning of how you would arrive
at the correct answer.
Do not assume the data files are clean or well-structured
(e.g., missing values, inconsistent data type in a column).
Do not assume the data type of the columns is what you see in
the data snippet (e.g., 2012 in Year could be a string, instead
of an int). So you need to convert it to the correct type if
your subsequent code relies on the correct data type (e.g.,
cast two columns to the same type before joining the two
tables).
You have to consider the possible data issues observed in the
data snippet and how to handle them.
Output the steps in a JSON format with the following keys:
- id: always "main-task" for the main task. For each subtask,
use "subtask-1", "subtask-2", etc.
- query: the question the step is trying to answer. Copy down
the question from above for the main task.
```

```
- data_sources: the data sources you need to check to answer
the question. Include all the file names you need for the main
task.
- subtasks: a list of subtasks. Each subtask should have the
same structure as the main task.
For example, a JSON object for the task might look like this:
{example_json}
You can have multiple steps, and each step should be a JSON
object. Your output for this task should be a JSON array of
JSON objects.
Mark the JSON array with {json_notation} to indicate the start
and end of the code block.
Then, provide the corresponding Python code to extract the
answer from the data sources.
The data sources you may need to answer the question are:
{file_paths}.
If possible, print the answer (in a JSON format) to each step
you provided in the JSON array using the print() function.
Use "id" as the key to print the answer.
For example, if you have an answer to subtask-1, subtask-2, and
main-task (i.e., the final answer), you should print it like
this:
print(json.dumps(
{{"subtask-1": answer1,
"subtask-2": answer2,
"main-task": answer
}}, indent=4))
You can find a suitable indentation for the print statement.
Always import json at the beginning of your code.
Mark the code with {notations} to indicate the start and end of
the code block.
```

B.2 Ablation Studies

We have started conducted ablation studies on key hyper-parameters, using the best-performing configuration of DS-Guru (i.e., self-correcting with GPT-o3). Here are our preliminary findings: The quality performance is positively correlated to token usage [1]. When varying the number of rows sampled per table, our result is consistent — success goes up as we sample more rows. We then observed a decrease at n=100, which is caused by the limited context window and our naive sampling algorithm. DS-Guru falls back to no data snippet when the prompt exceeds the context limit. DS-Guru showed consistent success across different numbers of maximum tries, with an initial slight increase. This potentially has two implications: (i) compile/runtime errors are not the major cause of failures; (ii) in a single-agent system, it may be difficult for the agent to get unstuck from a loop when fixing the error. We will discuss this in depth in failure analysis. We will update the paper to present these results and discuss them analytically under our 3-level evaluation framework. For reference, the full table of results is as follows: Varying the number of rows sampled in the input data snippet.

Table 13: DS-Guru with GPT-o3: performance and cost across different numbers of iterations.

Number of Iterations	**5**	**10**	**15**	**20**
Overall Performance (%)	23.36	22.83	20.73	21.33
Tokens/Iteration (Mean)	64,548.9	72,926.3	70,845.1	72,301.7

Table 14: Runtime performance by number of sampled rows per file. Runtime is in seconds.

SUT	Archeology	Astronomy	Biomedical	Environment	Legal	Wildfire	Overall	Runtime
10 Rows	18.75	12.80	8.63	34.52	13.32	37.42	22.89	732.45
50 Rows	23.48	10.55	7.87	37.60	14.08	40.63	24.68	655.61
100 Rows	20.61	11.95	8.53	34.84	12.20	40.60	23.36	1374.82
150 Rows	21.08	10.58	8.64	31.68	13.09	39.22	22.58	802.90

Table 15: Performance by number of tries. Runtime is in seconds.

SUT	Archeology	Astronomy	Biomedical	Environment	Legal	Wildfire	Overall	Runtime
5 Tries	20.61	11.95	8.53	34.84	12.20	40.60	23.36	1374.82
10 Tries	19.86	11.60	8.71	36.66	10.79	37.86	22.83	575.88
15 Tries	20.47	7.00	8.72	36.84	9.51	31.47	20.73	721.95

C smolagents Agentic Baseline Details

In this section, we describe the single-agent and multi-agent baselines systems we evaluated on **KramaBench** more.

C.1 smolagents-single

For the single-agent baseline, we use the open source deep research implementation by `smolagents` (git). This agentic framework follows a canonical think → action → response loop with agentic actions expressed in code. In addition to code, the system is equipped with a text inspector capable of processing different common formats originally released with Microsoft Magentic One (mic). While the official implementation also equips the system with a web browser by default, we disabled the internet access to allow for direct comparison with DS-Guru.

C.2 smolagents-reflexion

Our first multi-agent baseline is Reflexion (Shinn et al., 2023). In addition to an *Actor* agent, Reflexion (Figure 6) introduces (1) An *Evaluator* agent which provides internal feedback by evaluating the outcome of each action. (2)A *Self-reflection* agent which provides external feedback with the outcome and the evaluation of the action. Compared to traditional reinforcement learning techniques, feedback in Reflexion are expressed with natural language and stored in agent memory to guide future actions.

C.3 smolagents-pdt

Our second multi-agent baseline is based on AutoPrep (Fan et al., 2025), a framework for natural language question answering over tabular data. We augmented AutoPrep with tools for parsing non-tabular data and the hierarchical task decomposition technique similar with DS-Guru to address the complexity of **KramaBench** tasks. The original AutoPrep pipeline involves three agents: (1) Planner (2) Programmer (3) Executor. With the augmentations we implemented, the system employs two agents respectively playing the roles of (1) ***Planner*** and *(task)* ***Decomposer*** (3) ***Tool Executor***. We implemented this approach also using `smolagents` and illustrate the architecture in Figure 7.

D Dataset Details

The six input domains with the associated studies that we used to design our benchmark tasks are:

- **Archeology**: the data files consists of chronological, archaeological, faunal, and botanical data supporting the presence of Holocene hunter-gatherers on the Maltese Islands in the Mediterranean from roughly 8000 years ago to 7500 years ago. The files were collected from the publicly available data associated with the papers Groucutt et al. (2021); Scerri et al. (2025).
- **Astronomy**: the data files consist of the OMNI dataset Papitashvili & King (2020a;b) that contains near-Earth solar wind, plasma, and magnetic field data, the Swarm dataset Siemes et al. (2016);

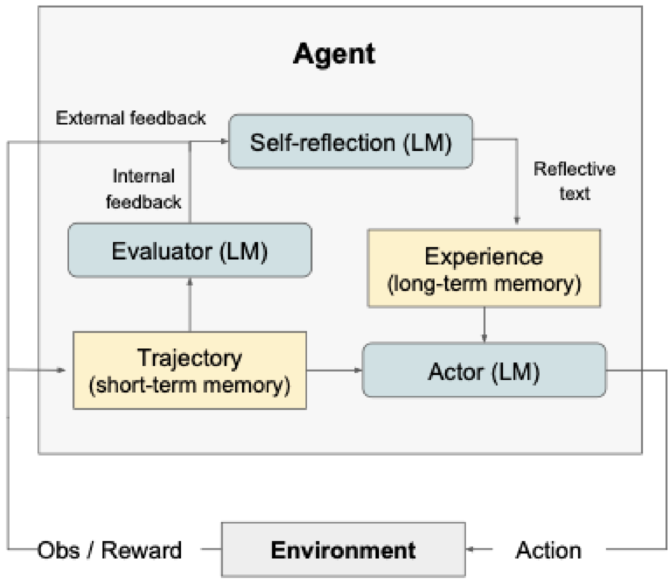

Figure 6: Architecture diagram of Reflexion. Reproduced from Figure 2(a) in Shinn et al. (2023).

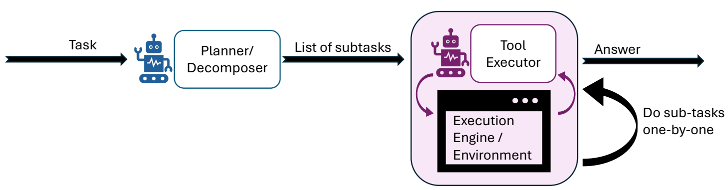

Figure 7: Architecture diagram of `smolagents-pdt`

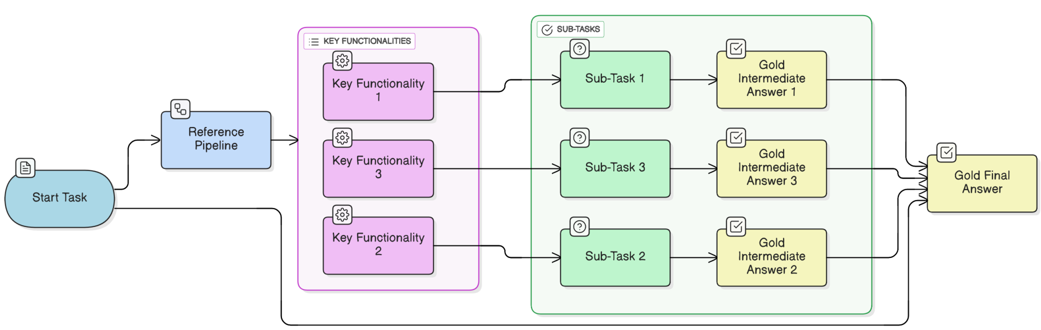

Figure 8: **KramaBench** maps system evaluation to both pipeline-design and sub-task-level evaluations, enabling analysis of why models succeed or fail beyond end-to-end performance.

European Space Agency (2013) that contains the magnetic field and geomagnetic field data, the SILSO Sunspot Number data Clette & Lefèvre (2015), Space-Track.org Two-Line Element Sets (TLEs) U.S. Space Command (2025), the National Oceanic and Atmospheric Administration (NOAA) Flux Forecast dataset U.S. Air Force & NOAA Space Weather Prediction Center (2025), and NOAA GOES Satellite dataset NOAA Office of Satellite and Product Operations (1994). The combination of these datasets has been used to analyze how activity from the Sun affects Earth's atmosphere, ocean currents, and weather by the authors of Briden et al. (2023); Parker & Linares (2024).

- **Biomedical**: the data files consist of the prote-ogenomic characterization of 95 prospectively collected endometrial carcinomas, respectively for 83 endometrioid and 12 serous tumors. Extensive analysis are done on these datasets to understand proteomic markers of tumor subgroups and regulatory mechanisms in the papers Dou et al. (2020); Gillette et al. (2020).
- **Environment**: the data files consist of beach water quality dataset from Massachusetts Environment Public Health Tracking (EPHT) Massachusetts Department of Public Health (2025b), the Massachusetts Bay beach dataset from Massachusetts Water Resources Authority (MWRA) Massachusetts Water Resources Authority (2025b), and the rainfall dataset from NOAA National Weather Service National Weather Service (2025), from 2002 to 2025. The data has been used in yearly reports Massachusetts Department of Public Health (2025a); Massachusetts Water Resources Authority (2025a) to uncover trends in beach water pollution and the correlation between rainfall and water quality.
- **Legal**: the datasets consists of 136 data files, accessible through the Federal Trade Commission (FTC) portal Federal Trade Commission (2025b) and Wikipedia Wikipedia contributors (2025), including information on merger filings, civil penalty actions, etc. The data is used in visualizations and dashboards that analyze nation-level debt collection and fraud detection, available at Federal Trade Commission (2025c;a).
- **Wildfire**: the datasets consists of NOAA wildfire dataset National Centers for Environmental Information (NCEI) (2025), National Interagency Fire Center (NIFC) Fire Information National Interagency Fire Center (2025), US Environmental Protection Agency (EPA) Air Quality Annual Data U.S. Environmental Protection Agency (2025), US Election 2020 Dataset Fontes (2020), Zillow Home Value Index Dataset Robikscube (2021), US Census 2020 U.S. Census Bureau (2025), and the Large wildfire Incident Status Summary Young et al. (2021) to understand wildfire incident location, cause, and consequences in the US from 2002 to 2016. This data has been used for analysis in the reports published by the NOAA and NIFC NCEI.Monitoring.Info@noaa.gov (2025); Center .

D.1 Cross-domain Accuracy Difference Analysis

In this subsection, we discuss the findings on likely causes of the differences in accuracy between different domains in **KramaBench**. We obtained these findings by manually analyzing the traces of smolagents-reflexion DR.

1. Archeology (33.33%) : In this domain, the system correctly solves questions answerable from a single table. However, errors occur for tasks requiring joining tables found in different files, because it treats multiple files as raw text instead of loading them as tables.
2. Astronomy (16.67%) : Astronomy tasks have the lowest average performance. In this domain, a large portion of the required input data is found in proprietary scientific formats (e.g., FORTRAN-style dat files). We observed that the agent struggles whenever it needed to load data from these files, e.g., SP3 orbit files or satellite products.
3. Biomedical (44.44%) : When working with biomedical data, the agent is reliable for shallow operations on a single sheet but fails to navigate large, multi-sheet workbooks and join data across sheets. Cross-sheet joins, especially between clinical and phosphoproteomics data, are problematic, and errors arise in correlation statistics due to sign miscalculations.
4. Environment (60.00%) : In the environmental domain, the system performs well on the relatively tasks involving clean CSV data, such as filtering, counting, and averaging. Unlike other tasks which struggle from data retrieval or understanding issues, the main issues arise from implementation/arithmetic mistakes, such as incorrect aggregation scopes or rounding errors, which explains the higher score.

Table 16: Detailed breakdown of per-domain tasks in KRAMABENCH. Reproduced from Table 3

Domain	# tasks	# subtasks	% Hard Tasks	# datasets	# sources	File size
Archeology	12	71	50.00%	5	2	7.5MB
Astronomy	12	68	50.00%	1556	8	486MB
Biomedical	9	38	66.66%	7	2	175MB
Environment	20	148	70.00%	37	3	31MB
Legal	30	188	53.33%	136	2	1.3MB
Wildfire	21	120	71.42%	23	7	1GB
Total	104	633	60.58%	1764	24	1.7GB

5. Legal (63.33%): In these tasks, the agent handles the straightforward pipelines well but struggles with loading data from messy files, i.e., that contain multi-row headers, partial subtotals, and metadata rows. Amongst the common errors that stem from these shortcomings, one example is that sum, means, and aggregations are only partial due to incorrect loading.

6. Wildfire (52.38%) : Within wildfire-related tasks, the system faces challenges with geospatial data and temporal/statistical reasoning. It struggles with GeoPackage layers and spatial joins, as well as with rolling-window aggregations for weather. However, text lookups and simple value comparisons work relatively well.

E TASK DETAILS

Across the 6 workloads, we supply 104 end-to-end data science pipelines. The table for the overall breakdown of the tasks over the workloads is reproduced at Table 16 for convenience. In this section, we use an example from the **archeology** workload to explain the organization of tasks.

Each workload is associated with a data lake consisting of tabular data and unstructured textual data.

```
archeology/input/:
    climateMeasurements.xlsx
    conflict_brecke.csv
    radiocarbon_database_regional.xlsx
    roman_cities.csv
    worldcities.csv
```

Before tasks in a workload are sent to the system under test, the system receives the directory where the data lake resides and may index it offline. When tasks are prompted, the system should not receive information on which files in the data lake the task pertains to. Each end-to-end task is specified with a high-level natural language prompt. Consider the following example of end-to-end task from the **archeology** domain:

```
 What is the average Potassium in ppm from the first and last
time the study recorded people in the Maltese area?  Assume
that Potassium is linearly interpolated between samples.
Round your answer to 4 decimal places.
```

For evaluating the performance of our systems, we use three artifacts:

1. The end-to-end ground truth answer used to calculate the overall end-to-end score.
2. A sequence of key functionalities, extracted from a manually verified reference implementation for the solution in Python.
3. A sequence of subtasks, natural language questions whose correct answer depends on correct code implementation of a key functionality.

The key functionalities are manually refined to correspond to the functionalities that should exist in any pipeline that produces the correct output. The sequence of key functionalities for the example end-to-end task above is the following:

```
1. Load the radiocarbon_database_regional.xlsx and
   climateMeasurements.xlsx and read the first worksheet
   of each.
2. Remove rows or columns that are entirely NaN or do not
   contain relevant information from both dataframes to
   ensure clean numeric processing.
3. Convert both chronologies to calendar years:  for the
   radio-carbon table get the year as 1950 minus the
   'date'
4. Convert both chronologies to calendar years:  for the
   climate table get the year as 1950 minus the rounded
   'Age_ky.1' (in thousands of years) multiplied by 1000.
5. Determine the span of human presence in the Maltese
   area by taking the minimum and maximum 'year' in the
   radio-carbon dataframe.
6. For every integer year within the human presence
   span, locate the closest earlier and later rows in the
   climate dataframe and linearly interpolate (or directly
   return) the Potassium value 'K' and collect all these
   values.
7. Compute the mean of the collected Potassium values.
```

For each key functionality, we supply a **subtask** associated with the key functionality. Each subtask is annotated with the ground truth subtask answer. These subtasks are used to verify the code implementation capabilities of systems under test. Note that among correct pipeline implementations for the end-to-end task, key functionalities may be ordered or composed differently. The subtasks associated to the end-to-end example task are:

```
1. Which files contain information about Potassium in ppm
   and the maltese people?
2. What are the indices (0-indexed) in rows in the climate
   measurement dataframe that must be cleaned?
3. What are the calendar years in the radiocarbon table?
4. What are the calendar years in the climate table?
5. What are the minimum and maximum years of radiocarbon
   dating for the Malta region?
6. What are the Potassium values for each integer year
   between -7580 and -4050 (included)?  If the value is
   not available, use interpolatation between the closest
   earlier and later values.
7. What is the mean potassium value for the years between
   -4462 and -4055?  Use 4 decimal places.
```

F Human Baseline Details

In this section, we summarize how we conducted the human baseline and discuss the results and implications. To contextualize LLM performance on KramaBench, we conducted a human data science study involving nine participants. Each participant was assigned a subset of benchmark tasks and asked to solve them under the same data directory structure, resource constraints, and assumptions provided to our LLM agents. For every assigned task, participants produced:

- a complete, reproducible end-to-end solution in a Jupyter notebook,
- a detailed log of their active time, broken down into data exploration, pipeline design, coding, and debugging, and
- both draft-stage notes and a final clean solution, enabling direct comparison to LLM workflows and error modes.

Incorrect pipeline design (46%): The largest category. These errors occur when, for example, experts mis-specified a join, aggregation rule, grouping key, or filtering logic. This suggests that the most cognitively demanding part of real-world data science is pipeline design, rather than implementation.

Lack of domain knowledge (24%): Many tasks contain implicit domain assumptions (e.g., definitions of "violation," mapping categorical labels). Experts often produced internally consistent but mismatched interpretations. This shows that even humans struggle with domain-specific task semantics.

Incorrect inputs (12%): Tasks often require gathering information across multiple similarly named or structurally similar files, and even humans sometimes use wrong inputs. These errors reflect the challenge of navigating multi-file datasets.

Incorrect answer format (9%): Some errors are due to having the final outputs in the wrong representation (e.g., units, rounding, formatting), which did not match the one requested by the task.

Library/version issues (9%): Minor inconsistencies (e.g., pandas handling) that changed intermediate results enough to fail strict correctness checking.

Interpretation and implications. Multi-file, multi-step pipelines are inherently error-prone—even for trained experts. Humans have difficulty navigating a vast data lake, which we see as an opportunity for LLM-powered systems to quickly search through the lake and identify the target files. Having a reliable retriever could greatly improve accuracy. Ambiguity and assumed domain knowledge are a real factor in real-world data tasks. One possible way for future agentic data-science systems to combat this issue is to ask clarification questions and invite user input. Another approach is to branch out on possible solutions by clearly stating the assumptions. Pipeline design is the bottleneck. Nearly half of all errors (45.45%) are due to incorrect pipeline logic, highlighting that the core challenge is understanding what transformations to perform, not coding them. Overall, most human errors stemmed from misinterpreting ambiguous tasks, selecting the wrong files, or designing incorrect pipelines—challenges that mirror the dominant failure modes of LLM agents. This confirms that KRAMABENCH captures genuinely difficult, real-world data-to-insight tasks where even trained data scientists struggle with pipeline reasoning, multi-file navigation, and implicit domain assumptions.

G EVALUATION DETAILS

Considering the broad nature of data science tasks, and the challenges in correctly evaluating their design and implementation, **KRAMABENCH** evaluates systems on three capabilities. From the most to the least automated: (1) End-to-end automation (2) Pipeline design (3) Sub-task implementation.

We are primarily interested in systems that can solve end-to-end data science tasks fully correctly, which drives our main evaluation metric to be the result from the end-to-end automation setting.

G.1 MAIN METRIC: END-TO-END AUTOMATION SETTING

Each task in **KRAMABENCH** has a manually validated target output and is scored from [0,1]. Since pipelines might be composed of steps with varying nature, we identify six possible answer types for the target output. summarized and discussed in Table 3. For each answer type, we choose a scoring scheme normalized to the range $[0, 1]$, also shown in Table 3. When tested, the **total** score of system

Table 17: Answer type and example questions

Type	Example	Metric	Scoring
String (exact)	The name of a file to load.	Accuracy	0/1
String (approximate)	The month when an experiment started.	ParaPluie paraphrase detection (Lemesle et al., 2025)	0/1
Numeric (exact)	Counting the number of entries satisfying a predicate.	Accuracy	0/1
Numeric (approximate)	Prediction of a future experiment observation.	Relative Absolute Error (RAE) $\lvert\hat{y} - y\rvert/\lvert y\rvert$	$1/(1 + \text{RAE})$
List (exact)	Names of columns to filter data.	F1 (exact match)	F1 score
List (approximate)	Regression coefficients for different variables.	F1 score (approximate match > 0.9)	F1 score

F for a workload W is defined solely based on the end-to-end correctness as

$$\frac{\sum_{T \in W} \text{score}(F(T))}{|W|}$$

Each T is a task belonging to workload W, and $|W|$ is the number of tasks in workload W. The overall score for the entire benchmark suite is defined analogously.

G.2 LLM-as-a-judge Validation

To assess the validity of the evaluation for `String (approximate)` and `List (approximate)` with `String (approximate)` list members conducted via instruction tuning an LLM, we performed a small scale human-LLM evaluator agreement study. We asked three human reviewers to manually evaluate the equivalence between the reference solutions and the answers generated by 12 different SUTs (the three variants of DS-Guru across four different LLM backends). We run the LLM-as-a-judge evaluation pipeline three times. We report the Cohen's Kappa values for inter-human agreement, human-LLM agreement and inter-LLM calibration (Table 18). The possible values range from -1 (complete misalignment) to 1 (complete alignment). The results show very high inter-human agreement (~95% on average) and moderately high human-LLM agreement (~84% on average), indicating that our usage of LLM-as-a-judge provides meaningful evaluation results.

Table 18: Inter-Human, inter-LLM, and human-LLM agreement on approximate answer evaluation.

Inter-Human Agreement			Inter-LLM Agreement			Human–LLM Agreement		
Rater 1	**Rater 2**	**K**	**Rater 1**	**Rater 2**	**K**	**Rater 1**	**Rater 2**	**K**
Human_0	Human_1	0.949	LLM judge_0	LLM judge_1	1	Human_0	LLM judge_0	0.870
Human_0	Human_2	0.950	LLM judge_0	LLM judge_2	1	Human_1	LLM judge_0	0.867
Human_1	Human_2	0.949	LLM judge_1	LLM judge_2	1	Human_2	LLM judge_0	0.818

G.3 Additional Evaluation Settings

A system that cannot provide fully correct end-to-end results may still be helpful for end-users via assisting them in the process of data pipeline design and implementation. Motivated by the goal of assessing this type of helpfulness of systems, we conduct evaluations under two less-automated settings. In Section 4 detailing our experiments, we report these results as micro-benchmarks in Table 8.

Pipeline Design: This setting evaluates how many essential functions a system-generated pipeline includes. Here, we ask the system to provide an end-to-end pipeline implemented in Python that solves an end-to-end task. For evaluation, we manually curated an explicit list of key functionalities that any correct solution must implement for each task. We evaluate whether the generated pipeline code covers each functionality using the LLM evaluation method proposed in Tong & Zhang (2024). The score for a single task is computed as

$$\frac{\sum_{f \in KF(T)} \text{Judge}(f, P)}{|KF(T)|}$$

Here, $KF(T)$ denotes the set of human-annotated key functionalities for task T, $|KF(T)|$ is the number of those functionalities, f represents a single functionality, P is the pipeline the system generated under test, and Judge is a binary decision from an LLM-based evaluator indicating wether P contains the key functionality f. The overall score across a workload/the entire benchmark is the average of the individual task scores.

Sub-task Implementation: This setting evaluates the system's ability to correctly implement simpler, lower-level functionalities and individual data tasks required to solve the entire challenge when explicitly prompted. We provide the system with problem statements of sub-tasks generate in Step 4 of the benchmark curation. Each sub-task corresponds to a key functionality and represents an intermediate step within the full end-to-end pipeline, operating over the gold subset of the data lake. We assess sub-task performance by comparing the system's intermediate outputs to human-annotated references, using an evaluation approach similar to the end-to-end automated method described earlier in this section.

H SUMMARY OF LLM USAGE

In this section, we summarize our usage of LLMs in compliance with the conference policy. We used LLMs for the following purposes

1. LLMs were used for the semi-automated generation of fine-grained annotations for the benchmark. However, contributors manually improved and verified all annotations. This is described in detail in Subsection 2.1.
2. LLMs are an integral part of the systems we evaluated. Their roles in the systems are described in detail in Subsection 2.3 and Section 3.
3. LLM-as-a-judge were used to evaluade string paraphrases and code coverage. This is described in detail in Appendix G.
4. LLMs were used to generate better documentations in our repository.

In addition to these research-level involvement of LLMs, we also used LLMs for table formatting and paraphrasing some sentences already written by authors in favor of brevity.