CHARTS AS EPISTEMIC ACTIONS FOR EXPERIENCE-DRIVEN AGENTIC DATA ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Data agents built on ReAct-style loops now execute complex analytical workflows over structured data, yet their intermediate reasoning still runs almost entirely through text. Human analysts instead move between numbers, tables, and charts, reading visual evidence to expose trends and anomalies that text summaries drop. We ask whether charts can serve as internal epistemic actions inside a data agent's reasoning loop, and how to teach an agent to use them. We first build PROBEBENCH, a suite of 200 controlled tasks under four matched interface conditions, and find a persistent gap between what agents can read and what they choose to use. Agents score as well from self-generated charts as from numeric output, yet plotting access alone does not translate into consistent use. Task slices show that charts lead on temporal, pattern, and anomaly reasoning, while text leads on precise reading and conditional verification. We then propose CHARTMIND, which distils two families of experience from conditioned trajectory rollouts. One family records how to select and verify analytical operations, and the other records when to invoke text, charts, or both. A trajectory judge scores outcomes and labels intermediate steps, and a validation gate admits only experience that improves held-out performance. On three held-out benchmarks with four LLM backbones, CHARTMIND raises analytical accuracy and visual evidence quality over matched baselines, and backbones that never drew charts begin using them, all without touching model parameters. [1]

1 INTRODUCTION

Data agents translate an analytical goal into a sequence of grounded actions and derive insights step by step. Recent progress in ReAct-style reasoning and acting loops (Yao et al., 2023) has enabled these agents to plan, execute, inspect, and revise increasingly complex workflows over structured data. Analytical report generation, long-horizon exploration, and exact-answer analysis all now sit within their reach (Lei et al., 2025; Liu et al., 2026; Jing et al., 2025). Such agents execute text-based analytical actions competently.

Human analysts, in contrast, do not think only in text. They move continually between numerical summaries, tables, and charts, using visual evidence to expose trends, distributions, and anomalies that a scrolling table hides. A well-chosen chart at the right moment can reshape the next step of analysis. Charts thus act as epistemic actions in the analyst's loop rather than as post-hoc report artefacts. We therefore ask whether an agent should also treat charts as intermediate evidence and learn when to invoke them.

Existing methods address only part of this decision. Chart-generation methods optimise visualisations requested by a user or an agent (Dibia, 2023; Yang et al., 2024; Namgoong et al., 2025; Lu et al., 2026; Wang et al., 2025), while ChartMimic benchmarks chart-to-code generation (Yang et al., 2025). Chart-understanding methods assume that a chart is already available and focus on extracting information from it (Masry et al., 2023; Han et al., 2023; Liu et al., 2023a;b; Zhang et al., 2024; Kaur et al., 2026; Das et al., 2026), while ChartBench benchmarks complex chart reasoning (Xu et al., 2023). Recent work compares different input representations (Zhou et al., 2025; Ho et al., 2026), routes table–query pairs across textual, visual, and fused paths (Xing et al., 2026), or builds

[1] The code is available in the supplementary materials.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

visual evidence on demand during report writing (Lin et al., 2026). We study how a data agent selects textual and chart evidence at intermediate analytical states.

To turn this open question into a concrete study, we build PROBEBENCH, a probe suite of 200 tasks synthesised from InsightBench (Sahu et al., 2025) with four matched interface conditions and a common ReAct backbone. Three findings emerge. First, charts alone already sustain end-to-end analysis. Second, plotting access alone does not translate into consistent use. Third, charts and text behave as complementary evidence across task slices, with charts stronger on temporal, pattern, and anomaly reasoning and text stronger on precise reading and conditional verification. These findings motivate a method that turns representation choice into a learned behaviour of the agent.

We propose CHARTMIND, an experience-driven method that teaches a data agent to use charts inside the analytical loop. For each PROBEBENCH, CHARTMIND generates diverse trajectories under different representation conditions and uses f_{score} and f_{attr} to score outcomes and label intermediate decisions. A builder agent distils the labelled trajectories into two experience families. Analysis experience $\mathcal{E}_{\mathrm{ana}}$ records how to select and verify analytical operations, and representation experience $\mathcal{E}_{\mathrm{repr}}$ records when and how to invoke text, charts, or a mixture. Candidate packages are accepted only when a validation gate confirms that held-out performance strictly improves. On downstream tasks the data agent reads a frozen package as external memory, so representation choice becomes part of the analytical policy without changing model parameters.

Our main contributions are as follows.

- **Probe study of chart use.** We construct PROBEBENCH with four matched interfaces and four backbones. The suite exposes a capability–use gap in current data agents and quantifies where charts and text serve as complementary evidence.
- **Experience-driven method for chart use.** We propose CHARTMIND. Given conditioned trajectory rollouts, it distils analysis and representation experience under outcome scoring and step-level attribution, and only experience updates that pass a validation gate enter the frozen package. CHARTMIND is the first framework that treats charts as internal epistemic actions of a data agent through learned experience.
- **Consistent gains on held-out benchmarks.** On DAComp-DA, DDR-Bench, and DSBench with four LLM backbones, CHARTMIND raises analytical accuracy and visual evidence quality over matched baselines. Backbones with weak representation-selection habits begin using charts, and ablations confirm the contributions of rollout diversity, validation gating, and both experience types.

2 CHARTS IN THE ANALYTICAL LOOP

This section asks whether charts carry genuine analytical value for data agents and how their utility differs from text. We design PROBEBENCH, a controlled probe suite, and evaluate representative agents under four matched interfaces, then report the suite's construction, evaluation protocol, and three empirical findings.

2.1 PROBE DESIGN

Task construction. To isolate the effect of intermediate representations on agent reasoning, we build PROBEBENCH from InsightBench (Sahu et al., 2025), which provides enterprise tables, open-ended questions, executable analyses, and target insights. Each valid insight defines a task, yielding 200 tasks in total. A task contains a raw CSV, dataset description, analytical question, reference analysis, and target insight. The agent sees the table, question, description, schema, and first three rows, then selects and executes operations to answer in natural language; the target insight is hidden for evaluation.

Analytical dimensions. For post-hoc slicing without label leakage, we annotate two dimensions. *Question type* covers numeric lookup, composition, ranking, group comparison, temporal change, relationship, anomaly, diagnosis, and prediction. *Cognitive load* covers information location, scan-and-select, comparison, pattern recognition, relational interpretation, multidimensional integration,

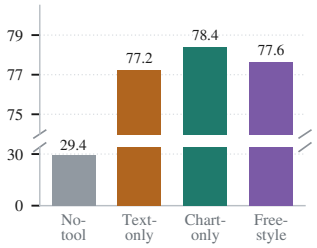

Figure 1: Judge scores under four matched PROBEBENCH interfaces. The score axis is split.

Table 1: Representation choice under **Free-style** across four backbones. *Both* counts tasks that use text and charts jointly.

LLM backbone	Score	Representation use			Interaction cost	
		Text	Chart	Both	Rounds	Tokens
Claude Sonnet 5	79.2	102	0	98	4.0	18.7k
GPT-5.6 Luna	76.2	198	0	2	3.1	6.8k
GPT-5.6 Sol	77.6	198	0	2	2.9	6.4k
DeepSeek V4.1 Flash	80.2	143	0	57	6.2	30.1k

and conditional reasoning. An LLM proposes the labels and a human verifies them; they never enter the agent context.

Agent interfaces. To hold everything but the analytical interface constant, all four settings share a vanilla ReAct agent with one general-purpose Bash tool. Each call starts a fresh process, files persist in a shared working directory, and the CSV is exposed read-only. We fix the model, initial context, interaction limit, and tool budget across conditions. **No-tool** removes execution tools. **Text-only** exposes stdout and stderr but forbids plotting. **Chart-only** exposes generated images and stderr while hiding stdout. **Free-style** returns stdout, stderr, and images and allows both text and charts.

Evaluation. To obtain comparable scores across settings, a shared LLM judge grades each answer against the target insight on a normalised $[0, 100]$ scale. Slice analyses use task-paired differences with clustered bootstrap intervals (Brown, 1995) and randomisation tests. Figure 1 and Table 1 summarise the two views used in the following analysis.

2.2 EMPIRICAL FINDINGS

Finding 1. Charts alone already sustain end-to-end analysis. To check whether charts carry enough evidence for analytical questions, we compare the four interfaces under the same backbone. Figure 1 shows that removing execution tools collapses the agent to a low score, while the Text-only and Chart-only interfaces achieve similar performance. Chart-only slightly outperforms Text-only, so the agent can read charts it just drew and produce answers comparable in quality to those obtained from numeric tables. Charts already satisfy the sufficiency condition for intermediate evidence, so the remaining question is whether agents actually use them.

Finding 2. Representation choice is model-dependent, not capability-driven. To probe whether the sufficiency of charts translates into use, we let each backbone freely choose between text and charts under the same Free-style interface. Table 1 shows large behavioural divergence. Claude Sonnet 5 combines text and charts on nearly half of the tasks and DeepSeek V4.1 Flash on more than a quarter, while GPT-5.6 Luna and GPT-5.6 Sol essentially fall back to text. The frequency of chart use does not track score. This shows that identical tool access yields inconsistent representation habits and confirms a capability–use gap. A method that closes this gap should shape when and how charts enter the loop, and cannot stop at exposing plotting tools.

Finding 3. Charts and text play complementary roles across analytical tasks. To locate where each representation helps, we slice results by question type and cognitive load. Figure 2 shows a clear pattern. Charts pull ahead on slices that demand visual pattern recognition, including temporal change, pattern recognition, and anomaly detection. Text pulls ahead on slices that demand precise reading, including numeric lookup, information location, and conditional verification. Free-style does not automatically inherit the stronger of the two on visual-pattern slices, so agents fail to switch to charts even where charts help most. Effective chart use therefore requires state-aware representation selection, and uniform access to both formats alone is not enough.

Summary of PROBEBENCH. PROBEBENCH shows that current data agents understand charts far better than they use them, and that charts and text play complementary roles under different task conditions. This gap and this complementarity together motivate a mechanism that learns state-dependent representation choice as part of the analytical policy, which we develop in Section 3.



Figure 2: Slice analysis of the four PROBEBENCH interfaces. **Top:** slices by question type. **Bottom:** slices by primary cognitive load. Markers show slice mean scores. The right column reports the Chart-only minus Text-only gap.

3 LEARNING TO USE CHARTS THROUGH EXPERIENCE

Section 2 leaves a concrete target. Charts already carry sufficient evidence, yet agents do not invoke them where they help, so representation choice has to become a decision the agent makes from the current analysis state. Reaching that target requires three things. The agent needs a place to hold representation knowledge, a way to acquire that knowledge from its own behaviour, and a way to consult it while analysing. CHARTMIND addresses these three needs in order. Section 3.1 stores experience as editable documents that keep analytical and representational knowledge side by side. Section 3.2 distils those documents from conditioned rollouts under outcome and step-level supervision, with a validation gate on every update. Section 3.3 lets the data agent consult the frozen documents inside its own analysis loop. Figure 3 shows the pipeline. Base model parameters stay unchanged throughout.

3.1 STORING EXPERIENCE AS EDITABLE DOCUMENTS

Representation choice depends on the analysis state, so the knowledge that governs it has to stay inspectable and revisable after training ends. We therefore keep experience outside the model as a document store that the agent reads during analysis and a builder revises during distillation. This choice also lets us audit what the agent learned and freeze it before any external evaluation.

Two-layer storage. To make experience readable, editable, and callable on demand, we organise it as a two-layer package. The storage layer is a filesystem tree in which an entry document indexes independent experience documents. Each document carries metadata fields `name`, `description`, and `category` together with its content, which supports direct reading, targeted editing, and versioning. The tool layer exposes `Read`, `Create`, `Edit`, and `Bash` for managing the package. The builder agent uses all four, while the data agent accesses experience only through a read-only retrieval interface, implemented as `kb_read` in downstream experiments. The data agent first fetches document metadata and then reads relevant content on demand, so context load stays low and grows with task complexity.

Analysis and representation experience. To separate what to compute from how to represent it, we index two experience families under different categories. Analysis experience $\mathcal{E}_{\text{ana}}$ holds reusable rules for selecting, organising, and verifying analytical operations. Representation experience $\mathcal{E}_{\text{repr}}$ holds rules for when intermediate evidence should be represented as text, charts, or a mixture, and for

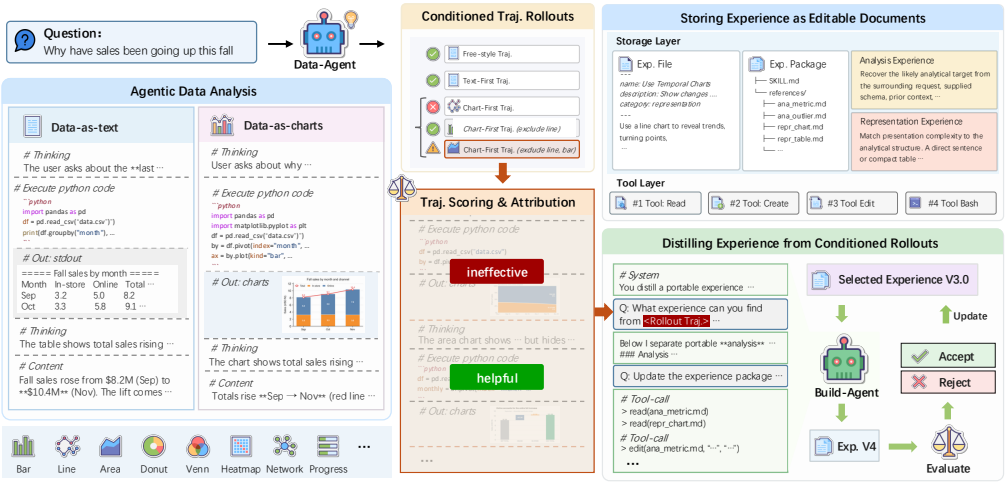

Figure 3: Overall pipeline of CHARTMIND. Conditioned rollouts diversify representation preferences over each PROBEBENCH. A trajectory judge produces outcome scores and step-level attributions. A builder agent distils analysis and representation experience under a validation gate.

how to consume the resulting evidence in later steps. The two families share the same storage and tool interface, and they are always updated and validated as one package. We write the v-th accepted version as $\mathcal{E}^{(v)} = \left(\mathcal{E}^{(v)}_{\text{ana}}, \mathcal{E}^{(v)}_{\text{repr}}\right)$.

3.2 DISTILLING EXPERIENCE FROM CONDITIONED ROLLOUTS

A single rollout per task reveals only the habit the backbone already has, and Section 2.2 shows that this habit is text-heavy for most backbones. Distillation from such data would reproduce the gap we want to close. We therefore force each task to be solved several times under different representation conditions, so the corpus contains chart-led trajectories even for backbones that would never produce them on their own, and outcome differences become attributable to representation choice.

Conditioned rollout bundle. To realise this, we generate multiple trajectories per task while holding the base agent, tools, and interaction budget fixed. Let $p_i = (D_i, q_i, z_i, y_i^\star)$ denote the i-th PROBEBENCH, where the four elements are the data, question, task context, and reference answer. A trajectory generated under representation condition $\rho \in \{\mathrm{U}, \mathrm{T}, \mathrm{C}_1, \mathrm{C}_2, \mathrm{C}_3\}$ is denoted by τ_i^ρ, and each task yields a bundle

$$\mathcal{B}_i = \left\{\tau_i^{\mathrm{U}}, \tau_i^{\mathrm{T}}, \tau_i^{\mathrm{C}_1}, \tau_i^{\mathrm{C}_2}, \tau_i^{\mathrm{C}_3}\right\}, \tag{1}$$

where τ_i^{U} is an unconstrained rollout, τ_i^{T} is a text-preferred rollout, and $\{\tau_i^{\mathrm{C}_k}\}_{k=1}^{3}$ are three chart-preferred rollouts. To keep the chart-preferred rollouts from collapsing onto a single visual habit, we enforce type exclusion across them. The second rollout cannot reuse any chart type used by the first, and the third cannot reuse any type used by the first two. This diversifies visual analyses without presuming that any chart type is intrinsically superior.

Outcome and step signals. Bundle-level outcomes alone would tell the builder which trajectory performed best but not which decisions contributed to its performance, so we attach two labels to every trajectory. The outcome scoring function $f_{\text{score}}(\tau_i^\rho \mid q_i, y_i^\star) \in [0, 1]$ evaluates the final answer against the reference. The step-level attribution function $f_{\text{attr}}(\tau_i^\rho \mid q_i, y_i^\star)$ labels each key step with a tag in $\{\texttt{Helpful}, \texttt{Ineffective}, \texttt{Harmful}\}$ and separates the contribution of analytical operations from that of representation choices. Together the two functions tell the builder which behaviours to retain and which to avoid.

Insight induction and package revision. To turn labelled trajectories into transferable rules, we split PROBEBENCH into construction and validation sets so that all trajectories from one task remain in the same split. At each iteration, we sample b tasks from the construction set and feed their bundles and labels to the builder. The builder first performs *insight induction*, comparing trajectories and their attributions to identify transferable analytical and representational patterns. It then performs *package revision*, reading the current package $\mathcal{E}^{(v-1)}$ and either merging new patterns into existing documents or creating fresh entries.

Validation-gated update. An unconstrained builder can overwrite a working document with a pattern that held only for a single batch, so every proposal must demonstrate transferable value. Let $Q(\mathcal{E})$ denote validation performance with package $\mathcal{E}$, and let $\widetilde{\mathcal{E}}^{(v)}$ be the proposal at iteration v. The update rule is

$$\mathcal{E}^{(v)} = \begin{cases} \widetilde{\mathcal{E}}^{(v)}, & Q(\widetilde{\mathcal{E}}^{(v)}) > Q(\mathcal{E}^{(v-1)}), \\ \mathcal{E}^{(v-1)}, & \text{otherwise.} \end{cases} \tag{2}$$

After multiple iterations, the accepted package $\mathcal{E}^\star$ is frozen for downstream use. The gate keeps validation performance monotonically nondecreasing and filters proposals that fit a single batch.

3.3 Consulting Experience inside the Analysis Loop

A frozen package is useful only if the agent consults it when a representation decision arises. We therefore attach the package to the running analysis loop as a read-only resource, so no extra training, retrieval model, or change to the core ReAct execution loop is needed.

Access to the frozen package. The data agent treats $\mathcal{E}^\star$ as external memory. It first receives document metadata, then reads the entries that match the question and the current analysis state. Reading costs a few tool calls and leaves the backbone untouched, so the same package serves every backbone we evaluate.

Representation choice at each step. Analysis experience $\mathcal{E}^\star_{\text{ana}}$ guides operation selection and verification, and representation experience $\mathcal{E}^\star_{\text{repr}}$ guides whether an intermediate result should be represented as text, charts, or both. Charts therefore enter the loop when the analysis state calls for visual pattern recognition, instead of appearing only as a final artefact.

4 Experiments

This section evaluates ChartMind on three held-out benchmarks that cover analytical report generation, long-horizon exploration, and exact-answer analysis. We describe the benchmarks and evaluation protocol, report main results by capability, and analyse the mechanism through ablations.

4.1 Benchmarks

To cover the main task forms of agentic data analysis, we adopt three benchmarks with complementary demands and use each benchmark's official protocol.

Analytical report generation. **DAComp-DA** (Lei et al., 2025) contains open-ended tasks over potentially multi-table SQLite databases and requires reports with analysis, conclusions, and visualisations. It is evaluated by Completeness, Accuracy, Insightfulness, Readability, Analytical Depth, Visualisation, and Overall.

Long-horizon exploration. **DDR-Bench** (Liu et al., 2026) evaluates long-horizon exploration without predefined queries in a 10-K setting over SEC disclosures. It is evaluated by Message-Wise accuracy, Trajectory-Wise accuracy, and Overall.

Exact-answer analysis. **DSBench** (Jing et al., 2025) derives exact-answer analytical tasks from ModelOff competitions with spreadsheet, text, and image inputs. It is evaluated by Task-level Accuracy, Competition-level Accuracy, and Overall.

4.2 Experimental Setup

Matched agent stack. The baseline agent and ChartMind share the model version, decoding configuration, base system instructions, data access, interaction budget, ReAct loop, and analytical tools within each backbone. ChartMind additionally receives the experience-access instructions, catalogue metadata, and read-only `kb_read` tool required to consult the frozen package.

Leakage prevention. All experience distillation uses ProbeBench only, and the three benchmarks are held out from experience construction, so downstream numbers measure transfer. Reported prior results are compared only when the split and protocol match.

Table 2: Analytical report generation on DAComp-DA. Higher is better on all metrics. CHARTMIND shares every configuration with the matched baseline except access to $\mathcal{E}^{\star}$.

Method	LLM Backbone	Comp.	Acc.	Insi.	Read.	Depth	Vis.	Overall
OpenHands	GPT-5	60.98	40.30	49.39	35.51	69.80	21.40	46.99
	DeepSeek-V3.1	49.88	33.25	41.66	36.00	33.20	11.00	33.87
DA-Agent	GPT-5	64.23	43.81	56.89	43.59	76.80	27.44	50.84
	DeepSeek-V3.1	48.74	32.97	42.43	37.25	35.00	11.45	34.33
Baseline	Claude Sonnet 5	70.27	52.55	68.85	90.80	89.00	62.80	69.84
	GPT-5.6 Luna	66.21	46.61	65.45	83.83	80.74	7.66	54.32
	GPT-5.6 Sol	71.74	58.04	72.42	94.60	93.00	61.80	73.24
	DeepSeek V4.1 Flash	83.91	67.80	81.77	98.00	94.44	75.72	82.23
CHARTMIND	Claude Sonnet 5	73.31	54.12	70.20	91.17	89.74	68.91	72.37
	GPT-5.6 Luna	68.10	51.18	65.24	82.83	85.00	27.61	59.78
	GPT-5.6 Sol	72.38	60.32	74.08	95.94	93.54	63.03	75.81
	DeepSeek V4.1 Flash	84.11	70.41	84.36	98.10	95.03	78.34	83.94

Table 3: Long-horizon exploration on DDR-Bench (10-K). Msg. and Traj. denote Message-Wise and Trajectory-Wise accuracy.

Method	LLM Backbone	Msg.	Traj.	Overall
ReAct	Claude Sonnet 4.5	77.61	60.58	69.10
	GPT-5.1	37.12	44.25	40.69
	GPT-5.2	44.89	41.09	42.99
	DeepSeek V3.2	60.08	38.15	49.12
Baseline	Claude Sonnet 5	61.19	52.18	56.69
	GPT-5.6 Luna	44.52	45.11	44.82
	GPT-5.6 Sol	60.20	58.89	59.55
	DeepSeek V4.1 Flash	64.19	59.01	61.60
CHARTMIND	Claude Sonnet 5	64.01	60.31	62.16
	GPT-5.6 Luna	47.28	49.62	48.45
	GPT-5.6 Sol	63.25	59.25	61.25
	DeepSeek V4.1 Flash	65.13	60.43	62.77

Table 4: Exact-answer analysis on the DSBench Data Analysis subset. Task and Comp. denote Task-level and Competition-level accuracy.

Method	LLM Backbone	Task	Comp.	Overall
Code Interpreter	GPT-3.5	11.16	8.23	9.70
	GPT-4o	23.82	22.65	23.24
AutoGen	GPT-3.5	20.82	13.80	17.31
	GPT-4o	34.12	26.72	30.42
Baseline	Claude Sonnet 5	88.47	88.42	88.45
	GPT-5.6 Luna	75.11	73.84	74.47
	GPT-5.6 Sol	89.90	89.40	89.70
	DeepSeek V4.1 Flash	91.32	90.64	90.98
CHARTMIND	Claude Sonnet 5	90.31	89.93	90.12
	GPT-5.6 Luna	76.72	76.25	76.49
	GPT-5.6 Sol	89.94	89.65	89.80
	DeepSeek V4.1 Flash	91.43	90.93	91.18

Package freezing. We build a single package with GPT-5.6 Sol and freeze the accepted version before external evaluation. The same frozen package is then used by every evaluated backbone, so no downstream evaluation signal reaches the package.

4.3 MAIN RESULTS

Analytical report generation. Table 2 shows that CHARTMIND consistently improves report-style analysis across all four matched backbones, raising Overall by 1.71–5.46 points, with a mean gain of 3.07. The largest improvement appears in Visualisation, which increases by 7.48 points on average and by 19.95 points for GPT-5.6 Luna. Accuracy also rises for every backbone, by 2.76 points on average. These coupled gains indicate that the learned experience improves both analytical correctness and evidence presentation, rather than merely producing more polished reports.

Long-horizon exploration. On DDR-Bench, CHARTMIND improves Overall for every matched backbone by 1.17–5.47 points, yielding a mean gain of 2.99 points. Message-Wise and Trajectory-Wise accuracy increase consistently, by 2.39 and 3.60 points on average, respectively. The larger trajectory-level gains, particularly for Claude Sonnet 5 and GPT-5.6 Luna, show that the benefit extends beyond isolated responses and accumulates over the full analytical trajectory. Thus, the learned experience supports both local analytical decisions and coherent long-horizon exploration.

Exact-answer analysis. Table 4 shows consistent gains on DSBench despite the strong baseline performance. CHARTMIND improves Overall for all four backbones by 0.10–2.02 points, with a mean gain of 1.00 point. Task-level and Competition-level accuracy increase by 0.90 and 1.12 points on average, respectively. Gains are larger for Claude Sonnet 5 and GPT-5.6 Luna, while the smaller improvements on GPT-5.6 Sol and DeepSeek V4.1 Flash likely reflect their limited remaining

Table 5: Distillation procedure on DAComp-DA.

Variant	Acc.	Vis.	Overall
baseline	46.61	7.66	54.32
one-step distillation	49.31	18.75	56.94
w/o conditioned rollouts	47.43	8.43	57.11
w/o validation gate	46.93	12.43	56.73
CHARTMIND	51.18	27.61	59.78

Table 6: Experience types on DAComp-DA.

Variant	Acc.	Vis.	Overall
baseline	46.61	7.66	54.32
analysis only	50.31	10.43	58.74
representation only	49.31	20.75	57.31
CHARTMIND	51.18	27.61	59.78

headroom. These results demonstrate that richer analytical experience does not trade off exact-answer correctness.

4.4 ABLATION STUDIES AND MECHANISM ANALYSIS

Ablation protocol. To locate which mechanism inside CHARTMIND drives the gains, we run all ablations on DAComp-DA with the same backbone, ReAct loop, tool interface, interaction budget, and evaluator. We report Acc., Vis., and Overall, which measure analytical correctness, visual evidence quality, and aggregate performance. Each ablation changes only the component under study.

Iterative distillation. One-step distillation improves Overall from 54.32 to 56.94 over the experience-free baseline, confirming the value of trajectory-derived experience. CHARTMIND further reaches 59.78, exceeding one-step distillation by 1.87 points in Acc. and 8.86 points in Vis. Iterative refinement therefore provides additional value beyond a single-pass summary, particularly for constructing visual evidence.

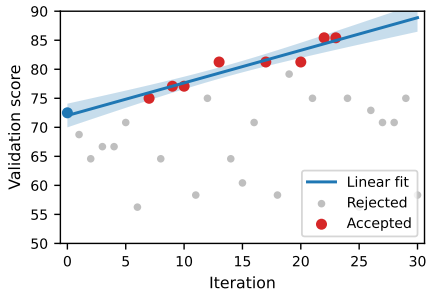

Figure 4: Validation trajectory during experience distillation. The gate admits only proposals that improve held-out performance.

Conditioned rollouts and validation gate. Using only unconstrained rollouts reduces Overall from 59.78 to 57.11, while removing the validation gate lowers it to 56.73. Both variants remain above the baseline but fall short of full CHARTMIND. Their larger drops in Vis. than in Acc. show that diverse representation-conditioned evidence and validation-based selection are particularly important for learning representation-aware analytical behaviour.

Two experience families. Both experience families improve the baseline, but in different ways. Analysis experience raises Acc. by 3.70 points and achieves the stronger single-family Overall, whereas representation experience produces a larger 13.09-point gain in Vis. Combining them yields the best Acc., Vis., and Overall, showing that analytical operation selection and evidence representation provide complementary rather than redundant guidance.

Optimisation dynamics. Figure 4 tracks the retained validation score across 30 candidate updates. The gate accepts eight proposals and raises the score monotonically from 72.50 to 85.42, while rejecting 22 others with scores as low as 56.25. Not every local edit transfers, and the gate stabilises optimisation by preventing harmful updates from entering the final package.

Trajectory behaviour on DAComp-DA. To connect the ablations to agent behaviour rather than only judge scores, we aggregate completed trajectories on DAComp-DA for the matched baseline and full CHARTMIND runs. For each backbone we report the mean number of tool-call rounds per task and the share of *analysis-phase* Bash calls—all but the final two invocations per task—that emit a valid chart image. Figure 5 shows that CHARTMIND increases interaction depth on every backbone. GPT-5.6 Luna rarely plots during analysis under the baseline but shifts towards chart use under CHARTMIND; this shift corresponds to the large Vis. gain in Table 2. GPT-5.6 Sol already exports figures near trajectory end under the baseline, so a raw all-call chart count overstates in-loop chart reasoning; the analysis-phase share stays low and does not consistently increase across backbones, indicating that CHARTMIND can improve visual evidence quality without uniformly raising plot frequency.

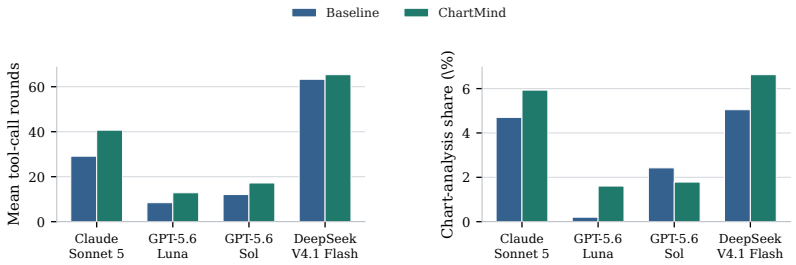

Figure 5: DAComp-DA trajectory statistics for the matched baseline and CHARTMIND. **Left:** mean tool-call rounds per completed task. **Right:** fraction of analysis-phase Bash calls (excluding the final two per task) whose observation includes a valid chart image.

5 RELATED WORK

Data agents. LLM-driven data agents follow a plan–execute–observe–revise loop over structured data. Representative systems couple reasoning with executable environments and cover heterogeneous analytical tasks. DS-Agent (Guo et al., 2024) retrieves historical cases inside an executable loop, Data Interpreter (Hong et al., 2025) organises tasks as dynamic directed acyclic graphs, AIDE (Jiang et al., 2025) models solution development as tree search over code, DS-STAR (Nam et al., 2025) adds sufficiency verification, AutoKaggle (Li et al., 2025) adopts multi-agent division of labour, DeepAnalyze (Zhang et al., 2025) trains an end-to-end data science model, and InsightBench (Sahu et al., 2025) extends evaluation to open-ended business insight generation. These systems keep the analytical loop text-centric and treat charts primarily as report artefacts. CHARTMIND instead makes charts first-class intermediate actions in the analytical loop.

Chart generation and understanding. One line of work optimises the production or reading of a single chart. LIDA (Dibia, 2023), MatPlotAgent (Yang et al., 2024), AMACE (Namgoong et al., 2025), MultiVis-Agent (Lu et al., 2026), and ChartOptimiser (Wang et al., 2025) improve chart generation through rendered feedback, multi-agent collaboration, and intent-aware design, while ChartMimic (Yang et al., 2025) benchmarks chart-to-code generation. UniChart (Masry et al., 2023) and ChartLlama (Han et al., 2023) advance chart reading through specialised training, while ChartBench (Xu et al., 2023) benchmarks complex chart reasoning. DePlot (Liu et al., 2023a), MatCha (Liu et al., 2023b), and TinyChart (Zhang et al., 2024) strengthen numerical analysis through derendering and programmatic reasoning. These methods assume that the chart or the plotting request is already fixed. CHARTMIND decides when a chart should exist at all inside an analytical trajectory.

Representation choice for reasoning. Recent work studies how textual, tabular, or visual representations are used in reasoning. ChartAgent (Kaur et al., 2026) and Charts-of-Thought (Das et al., 2026) reason over existing charts, FRES (Zhou et al., 2025) and FormatMatters (Ho et al., 2026) compare different input representations, TableDART (Xing et al., 2026) routes table–query pairs across textual, visual, and fused paths, and EvidFuse (Lin et al., 2026) builds visual evidence on demand during report writing. CHARTMIND learns state-dependent representation choice from labelled analytical trajectories, so text and chart evidence are selected as the analysis unfolds.

6 CONCLUSION

We study whether charts can serve as internal epistemic actions inside a data agent's analytical loop. PROBEBENCH shows that current agents extract sufficient evidence from self-generated charts, yet plotting access alone does not translate into consistent chart use. Charts and text play complementary roles across task slices, with charts stronger on temporal, pattern, and anomaly reasoning and text stronger on precise reading and conditional verification. Motivated by these findings, CHARTMIND distils analysis experience $\mathcal{E}_{\text{ana}}$ and representation experience $\mathcal{E}_{\text{repr}}$ from conditioned trajectory rollouts using outcome scoring f_{score} and step-level attribution f_{attr}, and retains only experience updates that pass a validation gate. On three held-out benchmarks with four LLM backbones, CHARTMIND induces in-loop chart use, improves analytical accuracy, and lifts visual evidence quality without touching model parameters. These results position representation choice as part of the analytical policy rather than as a cosmetic output stage.

AI USE STATEMENT

OpenAI ChatGPT with GPT-5.6-series models was used to assist with translation, language polishing, and the organisation of author-written text. GPT-Image-2.5 was used only as a colour-palette reference; no generated image was included in the manuscript. LLMs are also part of the experimental pipeline described in the main text. They were not used to generate synthetic tables, questions, or reference answers. All AI-assisted content was reviewed and revised by the authors.

REFERENCES

Jason Brown. An introduction to the bootstrap. *Journal of Quality Technology*, 27(1):90–91, 1995. doi: 10.1080/00224065.1995.11979565. URL https://doi.org/10.1080/00224065.1995.11979565.

Amit Kumar Das, Mohammad Tarun, and Klaus Mueller. Charts-of-thought: Enhancing LLM visualization literacy through structured data extraction. *IEEE Transactions on Visualization and Computer Graphics*, 32(1):1427–1437, 2026. doi: 10.1109/TVCG.2025.3634813. URL https://doi.org/10.1109/TVCG.2025.3634813.

Victor Dibia. LIDA: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 113–126. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-demo.11. URL https://aclanthology.org/2023.acl-demo.11/.

Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. DS-Agent: Automated data science by empowering large language models with case-based reasoning. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2402.17453.

Yucheng Han, Chi Zhang, Xin Chen, Xu Yang, Zhibin Wang, Gang Yu, Bin Fu, and Hanwang Zhang. ChartLlama: A multimodal LLM for chart understanding and generation. *arXiv preprint arXiv:2311.16483*, 2023. URL https://arxiv.org/abs/2311.16483.

Xanh Ho, Yun-Ang Wu, Sunisth Kumar, Florian Boudin, Atsuhiro Takasu, and Akiko Aizawa. Format matters: The robustness of multimodal LLMs in reviewing evidence from tables and charts. *arXiv preprint arXiv:2511.10075*, 2026. URL https://arxiv.org/abs/2511.10075. Accepted to AAAI 2026.

Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Robert Tang, Xiangtao Lu, Xiawu Zheng, Xinbing Liang, Yaying Fei, Yuheng Cheng, Yongxin Ni, Zhibin Gou, Zongze Xu, Yuyu Luo, and Chenglin Wu. Data interpreter: An LLM agent for data science. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19796–19821, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1016. URL https://aclanthology.org/2025.findings-acl.1016/.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. AIDE: AI-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*, 2025. URL https://arxiv.org/abs/2502.13138.

Liqiang Jing, Zhehui Huang, Xiaoyang Wang, Wenlin Yao, Wenhao Yu, Kaixin Ma, Hongming Zhang, Xinya Du, and Dong Yu. DSBench: How far are data science agents from becoming data science experts? In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2409.07703.

Rachneet Kaur, Nishan Srishankar, Zhen Zeng, Sumitra Ganesh, and Manuela Veloso. ChartAgent: A multimodal agent for visually grounded reasoning in complex chart question answering. *arXiv preprint arXiv:2510.04514*, 2026. URL https://arxiv.org/abs/2510.04514. Accepted to ACL 2026.

Fangyu Lei, Jinxiang Meng, Yiming Huang, Junjie Zhao, Yitong Zhang, Jianwen Luo, Xin Zou, Ruiyi Yang, Wenbo Shi, Yan Gao, Shizhu He, Zuo Wang, Qian Liu, Yang Wang, Ke Wang, Jun Zhao, and Kang Liu. DAComp: Benchmarking data agents across the full data intelligence lifecycle. *arXiv preprint arXiv:2512.04324*, 2025. URL https://arxiv.org/abs/2512.04324.

Ziming Li, Qianbo Zang, David Ma, Jiawei Guo, Tuney Zheng, Minghao Liu, Xinyao Niu, Yue Wang, Jian Yang, Jiaheng Liu, Wanjun Zhong, Wangchunshu Zhou, Wenhao Huang, and Ge Zhang. AutoKaggle: A multi-agent framework for autonomous data science competitions. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=09LEjbLcZW.

Huanxiang Lin, Qianyue Wang, Jinwu Hu, Bailin Chen, Qing Du, and Mingkui Tan. EvidFuse: Writing-time evidence learning for consistent text-chart data reporting. *arXiv preprint arXiv:2601.05487*, 2026. URL https://arxiv.org/abs/2601.05487.

Fangyu Liu, Julian Eisenschlos, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Wenhu Chen, Nigel Collier, and Yasemin Altun. DePlot: One-shot visual language reasoning by plot-to-table translation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 10381–10399. Association for Computational Linguistics, 2023a. doi: 10.18653/v1/2023.findings-acl.660. URL https://aclanthology.org/2023.findings-acl.660/.

Fangyu Liu, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Yasemin Altun, Nigel Collier, and Julian Eisenschlos. MatCha: Enhancing visual language pretraining with math reasoning and chart derendering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12756–12770. Association for Computational Linguistics, 2023b. doi: 10.18653/v1/2023.acl-long.714. URL https://aclanthology.org/2023.acl-long.714/.

Wei Liu, Peijie Yu, Michele Orini, Yali Du, and Yulan He. Hunt instead of wait: Evaluating deep data research on large language models. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.02039.

Jinwei Lu, Yuanfeng Song, Chen Zhang, and Raymond Chi-Wing Wong. MultiVis-Agent: A multi-agent framework with logic rules for reliable and comprehensive cross-modal data visualization. *arXiv preprint arXiv:2601.18320*, 2026. URL https://arxiv.org/abs/2601.18320. Accepted to SIGMOD 2026.

Ahmed Masry, Parsa Kavehzadeh, Xuan Long Do, Enamul Hoque, and Shafiq Joty. UniChart: A universal vision-language pretrained model for chart comprehension and reasoning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14662–14684. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.906. URL https://aclanthology.org/2023.emnlp-main.906/.

Jaehyun Nam, Jinsung Yoon, Jiefeng Chen, Raj Sinha, Jinwoo Shin, and Tomas Pfister. DS-STAR: Data science agent for solving diverse tasks across heterogeneous formats and open-ended queries. *arXiv preprint arXiv:2509.21825*, 2025. URL https://arxiv.org/abs/2509.21825.

Hyuk Namgoong, Jeesu Jung, Hyeonseok Kang, Yohan Lee, and Sangkeun Jung. AMACE: Automatic multi-agent chart evolution for iteratively tailored chart generation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21472–21487. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1089. URL https://aclanthology.org/2025.emnlp-main.1089/.

Gaurav Sahu, Abhay Puri, Juan Rodriguez, Amirhossein Abaskohi, Mohammad Chegini, Alexandre Drouin, Perouz Taslakian, Valentina Zantedeschi, Alexandre Lacoste, David Vazquez, Nicolas Chapados, Christopher Pal, Sai Rajeswar Mudumba, and Issam Hadj Laradji. InsightBench: Evaluating business analytics agents through multi-step insight generation. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2407.06423.

Yao Wang, Jiarong Pan, Danqing Shi, Zhiming Hu, Antti Oulasvirta, and Andreas Bulling. ChartOptimiser: Task-driven optimisation of chart designs. *arXiv preprint arXiv:2504.10180*, 2025. URL https://arxiv.org/abs/2504.10180.

Xiaobo Xing, Wei Yuan, Tong Chen, Quoc Viet Hung Nguyen, Xiangliang Zhang, and Hongzhi Yin. TableDART: Dynamic adaptive multi-modal routing for table understanding. *arXiv preprint arXiv:2509.14671*, 2026. URL https://arxiv.org/abs/2509.14671. Accepted to ICLR 2026.

Zhengzhuo Xu, Sinan Du, Yiyan Qi, Chengjin Xu, Chun Yuan, and Jian Guo. ChartBench: A benchmark for complex visual reasoning in charts. *arXiv preprint arXiv:2312.15915*, 2023. URL https://arxiv.org/abs/2312.15915.

Cheng Yang, Chufan Shi, Yaxin Liu, Bo Shui, Junjie Wang, Mohan Jing, Linran Xu, Xinyu Zhu, Siheng Li, Yuxiang Zhang, Gongye Liu, Xiaomei Nie, Deng Cai, and Yujiu Yang. ChartMimic: Evaluating LMM's cross-modal reasoning capability via chart-to-code generation. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2406.09961.

Zhiyu Yang, Zihan Zhou, Shuo Wang, Xin Cong, Xu Han, Yukun Yan, Zhenghao Liu, Zhixing Tan, Pengyuan Liu, Dong Yu, Zhiyuan Liu, Xiaodong Shi, and Maosong Sun. MatPlotAgent: Method and evaluation for LLM-based agentic scientific data visualization. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11789–11804. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.701. URL https://aclanthology.org/2024.findings-acl.701/.

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

Liang Zhang, Anwen Hu, Haiyang Xu, Ming Yan, Yichen Xu, Qin Jin, Ji Zhang, and Fei Huang. TinyChart: Efficient chart understanding with program-of-thoughts learning and visual token merging. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1882–1898. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.112. URL https://aclanthology.org/2024.emnlp-main.112/.

Shaolei Zhang, Ju Fan, Meihao Fan, Guoliang Li, and Xiaoyong Du. DeepAnalyze: Agentic large language models for autonomous data science. *arXiv preprint arXiv:2510.16872*, 2025. URL https://arxiv.org/abs/2510.16872.

Wei Zhou, Mohsen Mesgar, Heike Adel, and Annemarie Friedrich. Texts or images? a fine-grained analysis on the effectiveness of input representations and models for table question answering. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 2307–2318. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.117. URL https://aclanthology.org/2025.findings-acl.117/.

A EXPERIENCE CONSTRUCTION COST

Tables 7 and 8 report API-recorded prompt and completion tokens from `llm_log`. All runs use `gpt-5.6-sol`. The 174 training tasks produce 870 trajectories: one unconstrained, one text-preferred, and three chart-preferred rollouts per task. Judge tokens come from the post-rollout evaluation turn; all remaining tokens belong to the rollout, builder, or gate agent.

Table 7: Token cost of conditioned rollout collection on the train split (174 tasks for each of the unconstrained and text-preferred policies; chart-preferred aggregates three runs per task).

Rollout policy	Traj.	Rollout agent		Trajectory judge		Total
		Input	Output	Input	Output	
Unconstrained	174	917,194	151,749	738,012	111,615	1,918,570
Text-preferred	174	938,777	167,501	747,263	112,236	1,965,777
Chart-preferred	522	3,489,026	623,247	2,920,261	357,221	7,389,755
All train rollouts	870	5,344,997	942,497	4,405,536	581,072	11,274,102

Table 8: Total token cost of experience construction. Distillation uses 30 iterations with 12 validation probes per iteration.

Stage	Input	Output	Total
Train rollout corpus (Table 7)	9,750,533	1,523,569	11,274,102
Builder (30 iterations)	9,237,636	174,256	9,411,892
Gate probe agent	4,235,416	387,437	4,622,853
Gate trajectory judge	2,160,964	226,357	2,387,321
Full construction pipeline	25,384,549	2,311,619	27,696,168

B DOWNSTREAM EVALUATION TOKEN OVERHEAD

Table 9 reports mean API-recorded prompt and completion tokens per *completed* downstream task for GPT-5.6 Luna under the configuration-matched baseline agent (no experience library) and CHARTMIND (read-only `kb_read` access to the frozen package). Counts are summed from each task's `usage` block in `trajectory.json` under `experiments/12_main_experiments/outputs`; benchmark judges and other offline evaluators are excluded.

Table 9: Mean agent token cost per completed task for GPT-5.6 Luna (baseline vs. CHARTMIND). Δ is the relative change in mean total tokens.

Benchmark	Baseline (no KB)			CHARTMIND (with KB)			Δ
	Input	Output	Total	Input	Output	Total	Total
DAComp-DA	73,339	9,312	82,651	122,451	9,733	132,184	+59.9%
DDR-Bench	89,797	10,923	100,720	111,517	8,483	120,000	+19.1%
DSBench	40,262	1,614	41,876	68,476	1,658	70,134	+67.5%

C AGENT AND LEARNING-SYSTEM IMPLEMENTATION

We report only behaviourally relevant prompt content and replace task-specific or repeated text with angle-bracket placeholders. Baseline and experience-augmented agents share the same ReAct loop, backbone, task input, interaction budget, and report-generation procedure.

C.1 DOWNSTREAM DATA AGENTS

Baseline data agent. The baseline receives the benchmark instruction as its user message. Its system prompt specifies the data source, inspection requirement, tools, and report contract:

Core system prompt: baseline data agent

```
You are a professional data analyst working on an open-ended business
question over a SQLite database.
Use execute_bash for Python/sqlite3 analysis and read_file, write_file,
and edit_file to maintain the report.
Do not clean the data or train machine-learning models.  Explore the
schema, samples, and question-relevant queries or aggregates.
Write the full report to <instance_id>.md.  Ground quantitative claims
in the executed analysis and finish with a concise summary.
<interaction limit and language requirement>
```

`execute_bash` runs Python or SQLite and returns text and chart outputs; three file tools maintain the Markdown report. Each task runs in a fresh directory against a read-only database. A final text-only turn reviews and returns the report.

Data agent with the experience library. The CHARTMIND agent keeps this execution path and adds read-only `kb_read`. The prompt exposes document metadata but loads document bodies only on demand:

System-prompt addition: experience access

```
The experience library does not replace exploration of this task's
data.
1.  Inspect the actual files, schema, samples, cohorts, aggregates,
and sanity checks with execute_bash.
2.  After understanding the data, optionally call kb_read when a
catalog entry fits the next analytical or representational decision.
3.  Apply the experience through fresh computation; never answer from
catalog text, memory, or a generic template.
<experience catalog metadata>
```

Thus the only functional changes are catalog metadata, read-only retrieval, and the explore-before-retrieval policy above.

C.2 CONDITIONED ROLLOUT AGENT

Each construction task has five matched rollouts: one unconstrained, one text-preferred, and three chart-preferred. All receive identical inputs, backbone, and `execute_bash` capability; only the requested representation policy differs.

The unconstrained prompt requires an evidence-grounded answer using only the given dataset. The text-preferred condition adds:

System-prompt addition: text-preferred rollout

```
You MUST prioritize TEXT as the primary analysis representation:
tables, printed statistics, and stdout summaries should carry most of
the reasoning.  Charts are optional supporting evidence only.
```

Chart-preferred rollouts instead require charts as the primary intermediate representation. Later chart runs exclude families used earlier. If $T(\tau_i^{C_j})$ denotes the families used in chart rollout j, the forbidden set for run k is

$$\mathcal{F}_{i,k} = \bigcup_{j<k} T(\tau_i^{C_j}). \tag{3}$$

Thus C_1 has no exclusion, C_2 excludes the chart families used by C_1, and C_3 excludes the chart families used by both earlier runs.

C.3 TRAJECTORY JUDGE

The judge runs only after answer generation, so reference information cannot affect the rollout. It receives the task, reference materials, final answer, and a compact summary of tool calls, charts, and Bash scripts:

Core prompt: trajectory judge

```
Act as a strict evaluator, not as the analyst. Compare semantic
meaning rather than wording overlap. Score the final answer from 0
to 4 against the reference, then attribute each main analytical step.
For every step, separately summarize the analysis operation and its
representation, assign a verdict, and explain how it affected the
final score. Return only the requested structured report.
<question, reference materials, answer, and trajectory summary>
<output schema>
```

Scores of 4, 3, 2, 1, and 0 denote fully correct, correct with minor omissions, partially correct, weakly related, and incorrect or absent answers, respectively. The normalised score is $f_{\text{score}}(\tau_i^\rho) = s_i^\rho/4$.

Each step receives an analysis summary, a representation summary, a comment, and one of three verdicts: `Helpful`, `Ineffective`, or `Harmful`. This compact report lets the builder compare trajectories without loading full histories.

C.4 BUILDER AGENT AND VALIDATION-GATED REVISION

The builder processes mini-batches of complete matched bundles through compact judge summaries. Learned experiences may not contain ground-truth answers or task-specific identifiers.

During *planning*, tools are disabled. The builder compares matched runs and separates analytical lessons from representation lessons:

Core prompt: builder agent

```
Distill cross-task portable lessons; never memorize task identifiers,
entities, or numeric answers. Compare text and chart runs on the
same task. Maintain two separate tracks: analysis experience for
operations and validation, and representation experience for modality
and visual choices. Convert failures into preventive rules and
successful steps into reusable procedures.
```

During *application*, it reads overlapping entries and then refines, adds, splits, deletes, or leaves them unchanged. Its tools are restricted to the experience directory, and each document must contain one atomic lesson with valid metadata. The candidate is then accepted or rejected by the validation rule in Equation 2.

D LEARNED EXPERIENCE CASES

Analysis experience case 1

```
name: Define the metric contract before calculation
description: When a request uses an ambiguous metric term, define
its entity, unit, population, time basis, and aggregation before
computing it.
category: analysis
```

Before calculating an ambiguous measure, state the metric contract: entity and observational unit; eligible population and exclusions; numerator and denominator; time anchor and comparison window; grouping levels, aggregation, and weighting. Inspect actual field values before mapping semantic terms to labels, and do not replace the requested outcome with a convenient proxy. Before finalizing, reconcile every computed field and reported conclusion against this contract.

Analysis experience case 2

```
name: Decompose the drivers of an elevated average
description: When explaining why a group average is high, separate
category frequency, category intensity, and contribution to the
total.
category: analysis
```

For each component category, report its record count, mean or median value, and share of the group's total. Compare the focal group's component mix with peers, then compare like-for-like component values across groups. When useful, recompute the focal average after excluding or standardizing the suspected high-value components. This separates composition effects from within-category intensity and prevents confusing the largest category by count with the main per-record driver.

Representation experience case 1

```
name: Prefer text or tables for compact results
description: When the result is a simple scalar, ranking, or small
grouped summary, use concise text or a labeled table unless a chart
adds material value.
category: representation
```

Match presentation complexity to the analytical structure. A direct sentence or compact table is often clearer for a few exact values, a short ranking, or a binary comparison. Add a chart only when it improves perception of pattern, spread, trend, or scale; plotting is not a quality requirement by itself.

Representation experience case 2

```
name: Verify every chart against computed results
description: After generating a chart, reconcile its data, ordering,
units, labels, and claims with the supporting table before presenting
it.
category: representation
```

Perform a chart-then-verify pass: compare plotted values, population, filters, derived metric, and category order with the source table; check axis units, scales, baselines, legends, and time order; confirm titles, reference lines, and annotations; and ensure that the narrative states the same population, metric, direction, and comparison. If validation changes the data or ordering, recompute the source table and regenerate the chart rather than interpreting a stale figure.