VISUAL DATA AGENT: ACTIVE VISUALIZATION AND PERCEPTION OF DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-based agents are increasingly used to analyze and audit structured data. However, existing agents perceive data predominantly through textual representations, such as sampled records, serialized schemas, or summary statistics. Processing more records increases token cost and makes relevant information harder to locate in context, so agents typically inspect only a small portion of a large data source at a time. Furthermore, distributional data patterns that emerge across many records, such as a second peak in a series of values, are hard to capture. To bridge this gap, we present Visual Data Agent (VDA), an agent that actively plots data into chart-based views and captures patterns that sampled records and summary statistics can ignore. Starting with rule-based booting views, VDA actively explores data through calibrated charts paired with numerical summaries and task-specific agent notes, using coverage feedback to guide further inspection and visual memory to revisit earlier visual evidence. We evaluate VDA on seven public benchmarks, from ECG rhythms to satellite telemetry, and on DATAPATTERNBENCH, a benchmark we construct that asks an agent to find distributional structure in real multi-table databases. On DATAPATTERNBENCH, VDA improves detection F1 from 59.8% to 84.1% and discovery F1 from 43.1% to 78.0% over a strong coding agent (OpenHands) using the same base LLM, GPT-5.4. On six localization and classification benchmarks, VDA achieves an average absolute accuracy gain of 9.2% over OpenHands, averaging pooled accuracy gains across GPT-5 and GPT-5.4. In a controlled study with GPT-5.4, charts outperform numerical summaries in detection F1 (75.6% vs. 70.7%), highlighting the value of preserving distributional information beyond numerical summaries.

1 INTRODUCTION

Large language models (LLMs) have shown remarkable capabilities in data analysis tasks, and LLM-based agents are increasingly deployed to explore, query, and audit structured data (Sun et al., 2026; Lei et al., 2025; Tian & Zhang, 2026). However, these agents perceive data predominantly through textual representations, such as sampled records or summary statistics. Due to the context limit, the agents need to iteratively select fragments of a large data source to perceive at a time. The root limitation of such a design is the constrained bandwidth of data perception. An agent facing millions of records can afford only several samples or a summary in the context, which can hardly capture long-range distributional data patterns. Reading more records increases context length. Longer contexts can weaken instruction adherence (Tian & Zhang, 2025; Li et al., 2024a) and reduce task performance (Hong et al., 2025). An effective data agent therefore needs a compact representation of patterns across many records.

We are inspired by how human analysts interpret an unfamiliar data source by plotting charts that aggregate and present many records at once. Long-range distributional patterns, such as a second peak or a periodic rhythm, are directly visible in the chart, whereas a textual description can struggle to present them.

When a textual description or summary omits a pattern, a stronger model cannot reliably recover it from that input alone. Statisticians have recognized this phenomenon as fundamental for half a century. Anscombe's quartet (Anscombe, 1973) presents datasets with the same summary statistics, including means, variances, and correlations, that exhibit entirely different structures in their

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.

plots. Recent work reports similar limitations in LLM-based data analysis. Bendinelli et al. (2025) equipped LLM agents with an IPython shell for tabular data cleaning and found that they "struggle to detect more complex errors that require understanding data distribution across multiple rows", naming visualization as future work. A recent benchmark (Sun et al., 2026) observes that returning rendered figures to agents improves their performance, and a controlled study (Zhou & Yu, 2025) shows that multimodal LLMs detect time-series anomalies more reliably from plots than from serialized numbers. While reasoning over images has been studied in domains such as image understanding, mathematical diagrams, and chart question answering (Hu et al., 2024; Masry et al., 2022), these images are either given as input or drawn as intermediate sketches, rather than rendered by an agent to inspect a large data source. These findings motivate a data agent that selects, renders, and revisits charts while inspecting a large database.

To bridge this gap, we present the Visual Data Agent (VDA), a data agent that can actively plot the data into charts and inspect them. VDA starts with rule-based booting views that give the agent a compact visual overview of the data. Each view combines a calibrated chart that efficiently shows the distributional information, a numerical summary that indicates key exact values, and task-specific notes written by the agent. An observation window keeps recent visual views in context, like a queue. As older charts leave the window, their numerical summaries and agent notes remain in context, while external visual memory preserves the notes for later retrieval. Coverage feedback identifies views the agent has not received and can prompt further inspection before it completes the task. Along with general agent capabilities, VDA can effectively plot charts and reason about the data on demand.

We evaluate VDA on seven public benchmarks (Wu & Keogh, 2023; Lavin & Ahmad, 2015; van den Burg & Williams, 2020; Kotowski et al., 2026; Clifford et al., 2017; Wu et al., 2015; Sahu et al., 2025), ranging from localizing an anomaly or a change point in a long series, to classifying labeled examples such as ECG rhythms and wafer maps, to discovering insights in a business dataset. Across the six localization and classification benchmarks, VDA improves pooled accuracy over the strong coding agent baseline (OpenHands), increasing it from 32.9% to 49.8% with GPT-5 and from 51.6% to 53.2% with GPT-5.4.

To evaluate defect detection and pattern discovery in large multi-table databases under controlled conditions, we further construct DATAPATTERNBENCH, using real databases from Spider 2.0-lite (Lei et al., 2025). DATAPATTERNBENCH reflects a real-world scenario, where a production database is not a clean file (Li et al., 2021; Bendinelli et al., 2025) but grows for years through migrations, outages, and repeated loads, each leaving a mark such as a column whose unit changed halfway, a month with no records, or a batch loaded twice. Specifically, DATAPATTERNBENCH has two splits. The detection split plants eight classes of realistic defects with exact ground truth, and the discovery split plants analytical patterns, such as trend breaks and seasonal rhythms, that summary statistics can miss. Across all 66 detection and 51 discovery evaluation cases of DATAPATTERNBENCH, VDA with GPT-5.4 achieves 84.1% and 78.0% F1, compared with 59.8% and 43.1% for OpenHands using the same base LLM.

The main contributions of this paper are summarized as follows.

- We present VDA, a data agent that actively creates and interprets charts, with perception components transferable to other agent frameworks.
- We construct DATAPATTERNBENCH, a benchmark with verifiable ground truth for evaluating data agents on defect detection and pattern discovery in real multi-table databases.
- We conduct a comprehensive study across ten LLMs, three agent frameworks, seven public benchmarks, and DATAPATTERNBENCH, showing when visual perception improves data analysis and where its benefits are limited.

2 METHOD

VDA extends a general-purpose data agent with visual observations and a plotting tool. We use OpenHands (Wang et al., 2025) as the default framework and retain its existing tools. At turn t, the agent model π selects an action a_t using the context c_t and available tools $\mathcal{A}$. The environment $\mathcal{E}$

returns an observation o_t, and the action and observation are appended to the context,

$$a_t \sim \pi(\cdot \mid c_t), \qquad o_t = \mathcal{E}(a_t), \qquad c_{t+1} = c_t \oplus (a_t, o_t). \tag{1}$$

VDA organizes visual observations into *views*. For a plotting specification v, the corresponding view is $\text{View}_v = (I_v, S_v, r_v)$, comprising a chart I_v, a numerical summary S_v, and task-specific agent notes r_v. The chart and summary describe the same data, and the agent adds its notes after inspecting them. Exact duplicate counts are provided as text-only views. Figure 1 illustrates the workflow. Schema-based rules select booting views to provide initial observations (Ⓐ). Rendering calibration adjusts the charts to make data structure visible, with numerical summaries computed from the same aggregates (Ⓑ). The observation window maintains a queue of recent views for the agent to inspect and annotate (Ⓒ). As older charts leave the window, their summaries and agent notes remain in context as text-only notes (Ⓓ). External visual memory preserves the notes for later retrieval (Ⓔ). Coverage feedback identifies views that have not been delivered (Ⓕ), and the plotting tool allows the agent to create additional views (Ⓖ). Algorithm 1 in Appendix E formally defines this procedure.

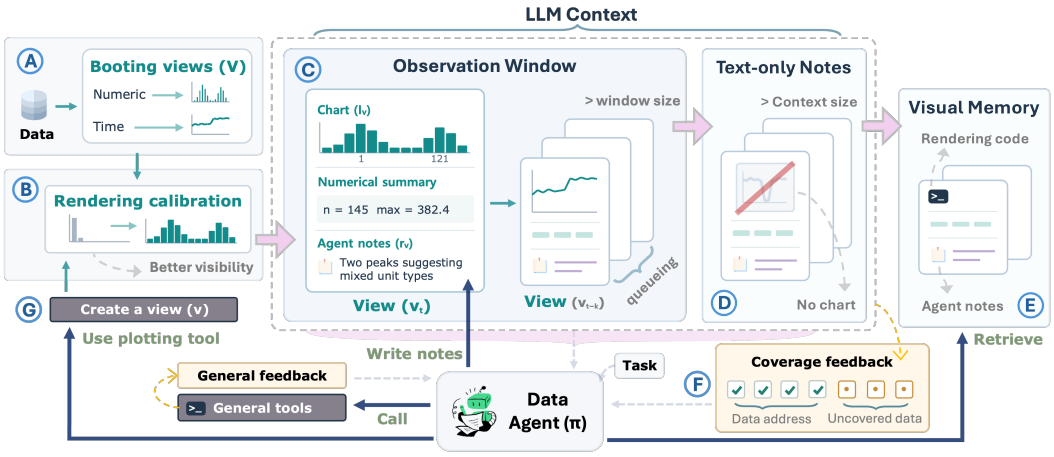


Figure 1: Overview of VDA. The observation window contains recent charts with numerical summaries and agent notes. The dashed enclosure marks the LLM context.

2.1 BOOTING VIEWS

A plotting tool requires the agent to choose what to inspect before it has observed the data. VDA provides initial observations through *booting views*, selected by deterministic rules based on the schema and column types (Figure 1 Ⓐ). A plotting specification $v = (k, \tau, x, \theta)$ identifies the chart type k, table τ, column x, and optional parameters θ, such as a date column z, a grouping column, or an aggregation function. Four chart types define the initial candidates. PROFILE shows a column's value distribution, and TREND shows record counts over time. TRENDY and TRENDNULL show a numeric column's mean and missing-value rate over time, respectively. For each table τ, let $\text{cols}(\tau)$ denote columns eligible for profiling, $\text{dates}(\tau)$ date columns, and $\text{meas}(\tau)$ numeric measures. The candidate set is

$$\mathcal{V}_{\text{cand}}(\mathcal{E}) = \bigcup_{\tau \in \mathcal{E}} \Big(\{\textsc{Profile}(\tau, x) : x \in \text{cols}(\tau)\} \cup \{\textsc{Trend}(\tau, z) : z \in \text{dates}(\tau)\} \\ \cup \{\textsc{TrendY}(\tau, x, z),\ \textsc{TrendNull}(\tau, x, z) : x \in \text{meas}(\tau),\ z \in \text{dates}(\tau)\} \Big). \tag{2}$$

The *view inventory* $\mathcal{V}(\mathcal{E})$ is an ordered set of views derived from these candidates. The selection rules exclude small tables and identifier-like columns from numeric profiling. They add views for exact duplicate counts, identifier values by record position, and temporal trends grouped by category. Temporal and record-position views precede duplicate counts and column profiles, with larger tables given priority. The first N entries form the booting views $\mathcal{V}_{\text{boot}}$. The agent receives these views in small batches and writes brief task-specific notes before continuing its analysis. It can request additional supported charts through the plotting tool.

For time-series tasks, booting provides the full series as an overview chart or a set of tiles. For classification, it displays labeled examples alongside the query. Appendix B.2 provides the detailed selection rules and an example.

2.2 Rendering Calibration

Default plotting settings can compress informative regions or leave much of a chart empty. Rendering calibration adapts the plot to the data so that relevant structure remains visible in a compact chart (Figure 1 Ⓑ). For example, linear histogram bins can place most observations in a single bin when values span several orders of magnitude. For a database view, a query q_k computes the aggregate A_v from the relevant table rather than sampled records. The rendering function g produces the chart, and σ produces a numerical summary from the same aggregate,

$$A_v = q_k(\mathcal{E}; \tau, x, \theta), \qquad I_v = g(A_v), \qquad S_v = \sigma(A_v), \qquad \mathrm{Render}(v) = (I_v, S_v). \tag{3}$$

The chart presents distributional structure, while the summary provides important exact values such as bin edges, counts, and group sizes for quantitative checks. Calibration follows predefined rules for binning, axes, and missing time intervals. Numeric profiles use logarithmic bins when the median p_{50} is positive and $\max(x)/p_{50} > 200$. For temporal means with at least twelve populated buckets, the displayed axis is capped at $1.3p_{95}$ when the 95th percentile $p_{95} > 0$ and $\max(x)/p_{95} > 6$, with values above the cap explicitly annotated. Missing calendar intervals are shown as zeros for record counts and as gaps for measured values. Numeric columns use histograms and categorical columns use frequency bars. Linear histograms use bins at least one unit wide for integer-valued columns. The numerical summary reports aggregate values without adding interpretations. The same rules apply to booting views and subsequent plotting requests. For a public series of length n, the overview uses $\lceil n/(\kappa W) \rceil$ tiles, subject to a tile limit, to control visual density, with width $W = 1400$ pixels and target density $\kappa = 30$ records per pixel.

2.3 Observation Window

The observation window provides a visual interface within the LLM context by maintaining a queue of recent views (Figure 1 Ⓒ). The renderer supplies each chart I_v and numerical summary S_v, which the agent inspects before writing task-specific notes r_v. These notes record observed patterns, their relevance to the task, and possible follow-up checks. For example, two separated peaks may prompt the agent to check whether a column contains mixed units. The notes preserve the agent's interpretation and are not treated as verified findings.

An image budget B limits the number of chart images retained in context, independently of the number of booting views N. New views are appended to the queue, and the oldest images are removed when the budget is exceeded. Views delivered in the same batch are added and removed together. Our database experiments use $B = 40$. This bounded window keeps recent charts available for comparison across turns.

When a chart leaves the observation window, its numerical summary, view identifier, and agent notes remain in context as *text-only notes* (Figure 1 Ⓓ). The agent can therefore revisit the reported values and its earlier interpretation without retaining the image.

2.4 External Visual Memory

External visual memory can retain agent notes and rendering code, indexed by view identifier, when older views leave the LLM context (Figure 1 Ⓔ). During booting, the system saves the view identifiers and up to 2000 characters of agent notes per batch. The agent uses its general tools to retrieve or update these notes during subsequent analysis.

2.5 The Plotting Tool and Coverage Feedback

Plotting tool. After booting, the agent selects its own actions using its general tools and the plotting tool (Figure 1 Ⓖ). A plotting request specifies a supported chart type, table, and columns, such as a numeric measure grouped by category. The same rendering calibration is applied, and the resulting chart and numerical summary enter the observation window. The agent can then use

queries and computations to check a visual observation, retrieve earlier notes, or request another chart. These actions support iterative analysis in which the agent tests its hypotheses and revises its answer.

Coverage feedback. For database tasks, coverage feedback lists predefined views that have not been delivered (Figure 1 Ⓕ). The set Seen records views successfully delivered during booting or plotting-tool calls,

$$\rho(\mathrm{Seen}) = \mathcal{V}(\mathcal{E}) \setminus \mathrm{Seen}. \tag{4}$$

After the initial analysis, the agent receives one reminder if this set is nonempty. For series tasks, feedback checks the plotting requests made after the automatic overview. A reminder is sent unless a requested interval contains the predicted location and spans at most 10% of the series. It reports the narrowest requested interval containing the prediction and its records per pixel, or states that no requested interval contains it. The automatic overview is excluded from this check. Classification in the main evaluation has no separate coverage reminder. Coverage measures which data have been presented, not the correctness of the agent's interpretation or answer. Appendix B.2 specifies the task settings.

3 Experiments

Our experiments evaluate end-to-end performance across agent frameworks, the effects of model settings, and the contributions of individual components and evidence representations.

3.1 Experimental Setup

Base LLMs. Our studies cover ten strong LLMs in total. We select GPT-5.4 and GPT-5 as the base models for main results. We keep the reasoning setting fixed within each paired comparison. Task-specific settings and budgets are given in Appendix B.2.

Comparison baselines. We compare VDA with three strong agents, OpenHands (Wang et al., 2025), MiroFlow (Su et al., 2026), and smolagents (Roucher et al., 2025). We also compare with strong LLMs under the zero-shot setting, using a detailed prompt but without tools or an additional agent loop. More details are included in Appendices B.2 and B.3.

Benchmarks and metrics. We evaluate on seven public benchmarks and our created DataPatternBench. UCR Anomaly Archive (Wu & Keogh, 2023), NAB (Lavin & Ahmad, 2015), TCPD (van den Burg & Williams, 2020), and ESA-ADB (Kotowski et al., 2026) test anomaly or change-point localization. CinC 2017 (Clifford et al., 2017) and WM-811K (Wu et al., 2015) test few-shot classification of ECG rhythms and wafer maps. InsightBench (Sahu et al., 2025) evaluates insight discovery in business data. Localization accuracy uses the benchmark-specific acceptance windows, classification uses exact label accuracy, and for InsightBench we measure gold-insight recall with an LLM matcher. All systems within a reported comparison use the same cases. Appendix B.1 gives the evaluation-set sizes and selection rules. To examine whether visual perception also helps answer specific questions rather than discover patterns, we evaluate non-visualization questions from DataGovBench (Hasegawa et al., 2026) (§3.4).

DataPatternBench evaluates discovery of distributional structure in real multi-table databases from Spider 2.0-lite (Lei et al., 2025), with files up to $300\,\mathrm{MB}$ and tables containing up to 10^6 records. The detection split plants defects per case from eight classes, including unit mixtures, null bursts, and duplicated loads. The discovery split plants temporal structures, including trend breaks, rhythms, and composition shifts. Findings matching pre-existing structures are excluded from scoring. A shared deterministic verifier checks submitted findings before exact matching on structure type, table, and column. Tables 1 and 2 report mean per-case F1 on all 66 detection and 51 discovery cases in DataPatternBench, with results for the development and held-out partitions reported separately in Appendix B.1. Controlled studies compare evidence representations on the same cases within each experiment (Appendix A.4).

3.2 MAIN RESULTS

End-to-end performance. Table 1 compares all systems on the same evaluation sets. On DATAPATTERNBENCH with GPT-5.4, VDA scores 84.1% detection F1 and 78.0% discovery F1, compared with 59.8% and 43.1% for OpenHands. Across the 611 public localization and classification cases, the gain depends on the model. Accuracy rises from 32.9% to 49.8% with GPT-5, but from 51.6% to 53.2% with GPT-5.4. The latter aggregate includes improvements on CinC-2017 (41.0% → 55.0%) and WM-811K (53.3% → 59.2%), alongside a decrease on UCR (39.2% → 32.4%). Applying the benchmark's statistical checks directly to the same views gives 63.8% detection F1 and 77.6% discovery F1, showing that known rules already solve much of the discovery task (Appendix B.1). Zero-shot performs poorly on DATAPATTERNBENCH, where a single prompt containing a schema and a few records provides limited evidence of distributional patterns, highlighting the need to iteratively inspect data beyond the initial prompt.

Table 1: End-to-end performance. Scores are F1 (%) on DATAPATTERNBENCH, accuracy (%) on localization and classification, and gold-insight recall (%) on InsightBench. Best per row is bold.

Benchmark	Zero-Shot	OpenHands	MiroFlow	Smolagents	VDA
GPT-5					
DATAPATTERNBENCH (detection)	1.4	35.5	38.3	25.6	**58.4**
DATAPATTERNBENCH (discovery)	0.0	18.7	19.5	13.8	**42.9**
UCR-Anomaly	15.2	19.6	16.8	21.2	**29.6**
NAB	59.6	55.8	53.8	61.5	**82.7**
TCPD	72.4	62.1	65.5	58.6	**89.7**
ESA-ADB	80.0	40.0	40.0	38.3	**86.7**
CinC-2017	31.0	39.0	31.0	**45.0**	40.0
WM-811K	35.8	35.0	17.5	23.3	**57.5**
InsightBench	39.3	49.1	39.7	31.5	**52.1**
GPT-5.4					
DATAPATTERNBENCH (detection)	0.8	59.8	63.8	54.3	**84.1**
DATAPATTERNBENCH (discovery)	0.0	43.1	38.9	30.0	**78.0**
UCR-Anomaly	22.0	**39.2**	35.6	21.6	32.4
NAB	53.8	76.9	**80.8**	69.2	78.8
TCPD	**89.7**	79.3	82.8	69.0	86.2
ESA-ADB	78.3	81.7	85.0	35.0	**86.7**
CinC-2017	36.0	41.0	31.0	35.0	**55.0**
WM-811K	54.2	53.3	43.3	38.3	**59.2**
InsightBench	35.3	**51.9**	50.9	47.7	**51.9**

Transfer across agent frameworks. Table 2 compares OpenHands and smolagents with and without visual augmentation. Shared entries reuse the runs in Table 1, with configurations detailed in Appendix B.2. On DATAPATTERNBENCH, adding VDA to smolagents changes detection F1 from 54.3% to 67.6% and discovery F1 from 30.0% to 55.8%. Pooled accuracy across the six public localization and classification benchmarks rises from 34.7% to 51.9%. The gains with both OpenHands and smolagents show that VDA transfers across agent frameworks.

3.3 ABLATION STUDY

Key component contributions. We compare full VDA with its base agent, OpenHands, and evaluate seven key components by removing each from the full system. Table 3 includes 46 detection and 34 discovery development cases and all predefined UCR, NAB, TCPD, ESA, and WM cases. Within each benchmark, conditions share the model, task inputs, and budgets, with only the specified component removed. Database ablations use the main configuration. Public ablations enable all applicable components, including the persistent memory omitted from the public main configuration. We include more details in Appendix B.3.

On DATAPATTERNBENCH, full VDA reaches a mean F1 of 81.3%, compared with 53.2% for OpenHands, averaging the two splits equally. Every removal lowers this performance. Removing booting

Table 2: Visual augmentation of two agent frameworks with GPT-5.4. Detection and Discovery report F1 (%), and the other columns report accuracy (%). Parentheses show absolute changes in percentage points from each framework's baseline before rounding, with colors indicating direction and magnitude.

	DATAPATTERNBENCH		Localization				Classification	
System	Det	Disc	UCR	NAB	TCPD	ESA	CinC	WM
OpenHands	59.8	43.1	39.2	76.9	79.3	81.7	41.0	53.3
OpenHands + VDA	84.1 (↑24.3)	78.0 (↑34.9)	32.4 (↓6.8)	78.8 (↑1.9)	86.2 (↑6.9)	86.7 (↑5.0)	55.0 (↑14.0)	59.2 (↑5.8)
Smolagents	54.3	30.0	21.6	69.2	69.0	35.0	35.0	38.3
Smolagents + VDA	67.6 (↑13.2)	55.8 (↑25.8)	33.6 (↑12.0)	84.6 (↑15.4)	86.2 (↑17.2)	86.7 (↑51.7)	40.0 (↑5.0)	60.0 (↑21.7)

Table 3: Ablation study with OpenHands and GPT-5.4. Detection and Discovery report development F1 (%) on DATAPATTERNBENCH, and the public columns report accuracy (%). "w/o" removes the named component. Bold scores mark column maxima.

	Detection	Discovery	UCR	NAB	TCPD	ESA	WM
OpenHands (w/o all components)	62.4	44.0	40.0	69.2	75.9	85.0	41.7
VDA w/o booting views	68.4	53.2	37.2	84.6	93.1	88.3	61.7
VDA w/o plotting tool	80.7	64.1	33.6	84.6	93.1	**90.0**	66.7
VDA w/o numerical summary	78.6	71.8	**42.4**	82.7	96.6	88.3	63.3
VDA w/o agent notes	76.3	55.2	42.0	78.8	96.6	88.3	**67.5**
VDA w/o coverage feedback	77.1	73.7	38.4	84.6	96.6	**90.0**	63.3
VDA w/o visual memory	82.2	72.5	37.6	**88.5**	89.7	**90.0**	66.7
VDA w/o rendering calibration	75.0	76.2	37.2	N/A	N/A	N/A	N/A
VDA	**85.2**	**77.4**	40.0	84.6	**100.0**	**90.0**	65.0

views or agent notes lowers mean F1 to 60.8% or 65.8%, followed by 72.4% without the plotting tool. Removing numerical summaries, coverage feedback, visual memory, or rendering calibration yields mean F1 between 75.2% and 77.3%. The results show that these effects depend on the task. Full VDA achieves the highest score on TCPD (100.0%) and ties for the highest on ESA (90.0%), but removing numerical summaries or agent notes raises UCR accuracy from 40.0% to 42.4% or 42.0%, respectively. On NAB, full VDA scores 84.6%, compared with 69.2% for OpenHands. Removing visual memory raises accuracy to 88.5%, while removing agent notes lowers it to 78.8%. On WM, full VDA scores 65.0%, compared with 41.7% for OpenHands. Removing agent notes raises accuracy to 67.5%, whereas removing booting views lowers it to 61.7%.

Effect of visual perception across base LLMs. Figure 2, left, compares the same agent with and without visual perception under a separate 20-turn protocol. Across ten LLMs, detection F1 ranges from 1.4% to 80.9% without visual perception and from 54.1% to 89.5% with it. Every model improves. Detailed scores appear in Appendix C.1. Gains are generally larger for weaker baselines. On the public tasks, charts with numerical summaries outperform both sampled records and numerical summaries alone at every tested generation in the GPT-mini and GPT families (Figure 2, right). At GPT-5.4, charts with summaries achieve 52.4% accuracy, compared with 42.7% for sampled records and 47.1% for summaries alone. At GPT-5.4-mini, the corresponding scores are 47.5%, 30.6%, and 37.6%. Stronger models therefore still benefit from charts across the tested generations, with varying gains.

On DATAPATTERNBENCH, a fixed-view comparison at GPT-5.4 isolates the contributions of sampled records, numerical summaries, charts, and charts with summaries (Table 10, Appendix C.2). Detection F1 is 60.1%, 70.7%, 75.6%, and 76.2%, respectively. Charts recover most of the combined condition's gain, with numerical summaries raising F1 from 75.6% to 76.2%. The public-benchmark breakdown shows that the combination is not always best. Numerical summaries lead on ESA-ADB and charts alone on CinC-2017. The benefit depends on how much information the text contains. In an additional control, we replace charts in full VDA with all plotted values as text,

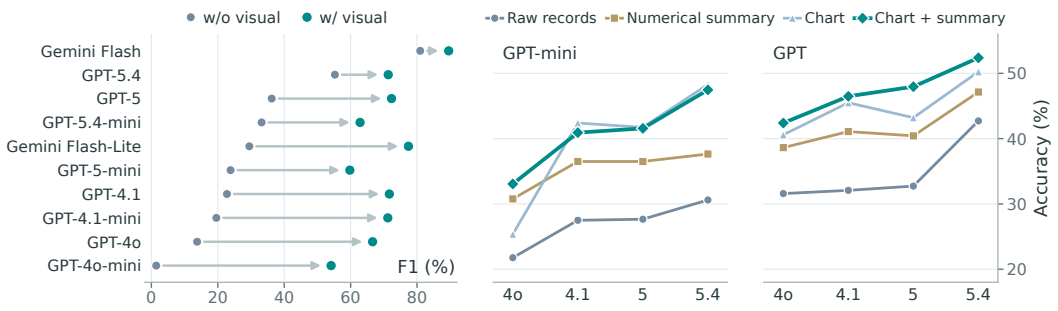


Figure 2: Controlled comparisons of evidence representations. **Left.** Detection F1 (%) with and without visual perception across ten LLMs on DATAPATTERNBENCH. **Right.** public-task accuracy across eight models, by model size.

preserving the remaining components and maximum budgets. The text condition has 11 failed runs. In a post hoc comparison of cases completed by both conditions, detection F1 is 83.7% with charts and 81.3% with complete text, while discovery F1 is 75.8% and 76.6%. This comparison does not establish an advantage over equally complete text, although charts use fewer recorded tokens (Appendix D.1).

Effect of reasoning effort. To understand how reasoning effort affects the benefit of visual perception, we vary GPT-5.4's reasoning effort while holding the agent and evaluation cases fixed (Table 4). Visual perception improves F1 at all four settings, from 55.3% to 71.3% with reasoning off and from 73.3% to 77.2% at high effort, with a smaller gap at each successive effort level. High effort without charts reaches 73.3%, exceeding the 71.3% achieved with charts and reasoning off. Thus, additional reasoning can partly compensate for the absence of visual evidence in this comparison. The results do not establish equal cost or equivalent performance.

Table 4: Detection F1 (%) on DATAPATTERNBENCH with different levels of reasoning effort using GPT-5.4. Each row compares the same agent with and without visual perception.

Reasoning Effort	w/o Visual Perception	w/ Visual Perception
Off	55.3	**71.3**
Low	61.2	**74.9**
Medium	68.4	**76.6**
High	73.3	**77.2**

3.4 ADDITIONAL EXPERIMENTS AND DISCUSSION

Token consumption. Charts can summarize more records without a proportional increase in context tokens. At a fixed rendering resolution, image token cost depends primarily on pixel dimensions rather than record count. This limits growth in the cost of each observation, but does not ensure lower total cost. In the fixed-view detection study, charts with numerical summaries use 37–40k tokens per case across GPT-5.4, GPT-5, and GPT-4o, compared with 22–23k for summaries alone (Appendix C.2). The full system also uses more requests and cumulative tokens than OpenHands under the same maximum budgets (Appendix B.2).

When does visual perception not help? Open-ended auditing requires deciding what to investigate and recognizing patterns across records, whereas a specific question may already define the required computation. On non-visualization questions in DataGovBench with GPT-5.4-mini, adding a plotting tool lowers mean within-file accuracy from 47.4% to 40.1% and increases the average number of actions from 6.7 to 8.5 on pairs with complete action logs (Appendix C.3). This result limits the benefit of plotting in the tested question-answering setting. It does not establish a general disadvantage for semantic parsing or text-to-SQL. These findings suggest that plotting should be chosen according to the information needed for the task. A chart can help an agent identify a

candidate subgroup or temporal interval before it knows which computation to perform. Numerical tools can then test that observation and quantify its extent. When the question already specifies the relevant rows and operation, direct computation may be sufficient. This distinction motivates future work on selecting when to plot, which view to request, and when to return to numerical verification under a fixed analysis budget.

4 Related Work

Visual reasoning and tool use. Visual Sketchpad (Hu et al., 2024) uses sketches as intermediate representations for multimodal reasoning, while related work studies images as a workspace for reasoning (Su et al., 2025) and active visual perception (Li et al., 2025). VisProg, ViperGPT, and PyVision (Gupta & Kembhavi, 2023; Surís et al., 2023; Zhao et al., 2025) generate code that composes or creates visual operations to inspect visual inputs, and VipAct (Zhang et al., 2026) coordinates specialized VLM agents with vision expert tools. VDA applies a related principle to structured data. The agent constructs charts from database aggregates and uses them to guide further analysis.

Visualization for model criticism. Automated modeling systems iteratively propose, evaluate, and refine statistical models (Li et al., 2024b), with related work automating model criticism (Li et al., 2024c) and benchmarking model discovery (Gandhi et al., 2025). VESTA (Rudman et al., 2026) creates and selects diagnostic tools, including visualizations and statistical tests, to refine statistical models. Our focus is exploratory analysis of the data itself, before a particular predictive or generative model has been specified.

Visualization for data analysis. Profiler (Kandel et al., 2012) combines statistical checks with suggested views to help analysts investigate data-quality issues. LIDA (Dibia, 2023) generates visualization goals and charts, while MatPlotAgent (Yang et al., 2024) uses visual feedback to correct generated plots. VDA uses charts as observations that guide an agent's search for defects and distributional patterns. Its contribution is to connect calibrated booting views and subsequent plotting through an observation window, agent notes, visual memory, and coverage feedback, supporting repeated inspection and verification across analysis turns.

Data agents and evaluation. A controlled study compares plots and serialized numbers for time-series anomaly detection (Zhou & Yu, 2025), and ChartQA evaluates reasoning over charts (Masry et al., 2022). Data-science benchmarks evaluate agents that receive rendered figures (Sun et al., 2026) or discover insights (Sahu et al., 2025). Cleaning benchmarks assess data errors and downstream performance (Li et al., 2021; Bendinelli et al., 2025). Our controlled comparisons examine how visual presentation and information completeness affect pattern detection. We also introduce DataPatternBench, which evaluates defect detection and pattern discovery with planted structures and deterministic scoring.

5 Conclusion

We present VDA, a data agent that analyzes data by actively plotting charts. VDA is built on top of a general-purpose agent and is augmented with booting views, rendering calibration, a chart-based observation window, visual memory, and coverage feedback. In addition to existing public benchmarks, we introduce DataPatternBench, a verified benchmark for defect detection and pattern discovery, to better evaluate VDA in modern agent settings. Our experiments show gains across base LLMs, agent frameworks, and reasoning efforts, with effects that vary by task. Charts help agents detect patterns that sampled records and numerical summaries miss, while comparisons with complete chart data as text show that information completeness explains part of these gains. These benefits persist across the tested model generations, showing that stronger base models can still benefit from visual perception. However, adding a plotting tool reduces accuracy in our question-answering evaluation, where discovering distributional patterns is not the main challenge.

AI USE STATEMENT

We used generative AI tools to assist with manuscript and figure editing, experimental planning and implementation, construction of additional evaluation cases, and result analysis. The core research ideas originated with the authors. The authors take full responsibility for the accuracy and integrity of the paper, including all AI-assisted material.

REPRODUCIBILITY STATEMENT

The anonymous code supplement contains the VDA implementation, benchmark construction and scoring scripts, case manifests, and saved results. It includes instructions for obtaining the source databases and reproducing individual cases. Appendices A and B describe benchmark validation, case selection, model settings, and experimental budgets. Appendices C and D specify the controlled comparisons, and Appendix E provides the algorithm.

REFERENCES

Francis J. Anscombe. Graphs in statistical analysis. *The American Statistician*, 27(1):17–21, 1973.

Tommaso Bendinelli, Artur Dox, and Christian Holz. Exploring LLM agents for cleaning tabular machine learning datasets. *arXiv preprint arXiv:2503.06664*, 2025.

Gari D. Clifford, Chengyu Liu, Benjamin Moody, Li-wei H. Lehman, Ikaro Silva, Qiao Li, A. E. Johnson, and Roger G. Mark. AF classification from a short single lead ECG recording: The PhysioNet/Computing in Cardiology Challenge 2017. In *Computing in Cardiology (CinC)*, volume 44, 2017.

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, 2023. doi: 10.18653/v1/2023.acl-demo.11. URL https://aclanthology.org/2023.acl-demo.11/.

Kanishk Gandhi, Michael Y. Li, Lyle Goodyear, Agam Bhatia, Louise Li, Aditi Bhaskar, Mohammed Zaman, and Noah D. Goodman. BoxingGym: Benchmarking progress in automated experimental design and model discovery, 2025.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

So Hasegawa, Shailaja Keyur Sampat, Lei Liu, and Wei-Peng Chen. Data analysis in the wild: Benchmarking large language models against real-world data complexities. *arXiv preprint arXiv:2607.06482*, 2026.

Kelly Hong, Anton Troynikov, and Jeff Huber. Context rot: How increasing input tokens impacts LLM performance. Technical report, Chroma, July 2025. URL https://www.trychroma.com/research/context-rot.

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Ranjay Krishna. Visual Sketchpad: Sketching as a visual chain of thought for multimodal language models. In *Advances in Neural Information Processing Systems*, 2024. arXiv:2406.09403.

Sean Kandel, Ravi Parikh, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Profiler: Integrated statistical analysis and visualization for data quality assessment. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, 2012. doi: 10.1145/2254556.2254659. URL https://idl.uw.edu/papers/profiler.

Krzysztof Kotowski, Christoph Haskamp, Jacek Andrzejewski, Bogdan Ruszczak, Jakub Nalepa, Daniel Lakey, Peter Collins, Jose Martínez-Heras, and Gabriele De Canio. European Space Agency dataset and benchmark for real-world anomaly detection in spacecraft time series. *Journal of Data-centric Machine Learning Research*, 2026. URL https://openreview.net/forum?id=bbpRMoatVO.

Alexander Lavin and Subutai Ahmad. Evaluating real-time anomaly detection algorithms – the Numenta Anomaly Benchmark. In *IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pp. 38–44, 2015.

Fangyu Lei et al. Spider 2.0: Evaluating language models on real-world enterprise text-to-SQL workflows. In *International Conference on Learning Representations*, 2025. arXiv:2411.07763.

Kenneth Li, Tianle Liu, Naomi Bashkansky, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Measuring and controlling instruction (in)stability in language model dialogs. In *Conference on Language Modeling (COLM)*, 2024a.

Michael Y. Li, Emily B. Fox, and Noah D. Goodman. Automated statistical model discovery with language models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, 2024b.

Michael Y. Li, Vivek Vajipey, Noah D. Goodman, and Emily B. Fox. CriticAL: Critic automation with language models, 2024c.

Peng Li, Xi Rao, Jennifer Blase, Yue Zhang, Xu Chu, and Ce Zhang. CleanML: A study for evaluating the impact of data cleaning on ML classification tasks. In *IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 13–24, 2021.

Yian Li, Xiaoyu Guo, Hao Zhang, Shuiwang Li, and Xiaowei Dai. Active visual perception: Opportunities and challenges, 2025. URL https://arxiv.org/abs/2512.03687.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, 2022.

Justin Matejka and George Fitzmaurice. Same stats, different graphs: Generating datasets with varied appearance and identical statistics through simulated annealing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 1290–1294, 2017.

Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. smolagents: a smol library to build great agentic systems. https://github.com/huggingface/smolagents, 2025.

William Rudman, Abhishek Divekar, Kanishk Jain, Sebastian Joseph, Stella S. R. Offner, Matthew Lease, Kyle Mahowald, Greg Durrett, and Junyi Jessy Li. VESTA: Visual exploration with statistical tool agents, 2026. URL https://arxiv.org/abs/2606.00384.

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 (ICLR)*, 2025.

Shiqian Su, Sen Xing, Xuan Dong, Muyan Zhong, Bin Wang, Xizhou Zhu, Yuntao Chen, Wenhai Wang, Yue Deng, Pengxiang Zhu, Ziyuan Liu, Tiantong Li, Jiaheng Yu, Zhe Chen, Lidong Bing, and Jifeng Dai. MiroFlow: Towards high-performance and robust open-source agent framework for general deep research tasks, 2026. URL https://arxiv.org/abs/2602.22808.

Zhaochen Su et al. Thinking with images for multimodal reasoning: Foundations, methods, and future frontiers. *arXiv preprint arXiv:2506.23918*, 2025.

Maojun Sun, Yifei Xie, Yue Wu, et al. DSAEval: Evaluating data science agents on a wide range of real-world data science problems. *arXiv preprint arXiv:2601.13591*, 2026.

Dídac Surís, Sachit Menon, and Carl Vondrick. ViperGPT: Visual inference via Python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Yuan Tian and Tianyi Zhang. Selective prompt anchoring for code generation. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Yuan Tian and Tianyi Zhang. PV-SQL: Synergizing database probing and rule-based verification for text-to-SQL agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 25827–25845, 2026.

Gerrit J. J. van den Burg and Christopher K. I. Williams. An evaluation of change point detection algorithms. *arXiv preprint arXiv:2003.06222*, 2020.

Xingyao Wang et al. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations (ICLR)*, 2025.

Ming-Ju Wu, Jyh-Shing R. Jang, and Jui-Long Chen. Wafer map failure pattern recognition and similarity ranking for large-scale data sets. *IEEE Transactions on Semiconductor Manufacturing*, 28(1):1–12, 2015.

Renjie Wu and Eamonn J. Keogh. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *IEEE Transactions on Knowledge and Data Engineering*, 35 (3):2421–2429, 2023. doi: 10.1109/TKDE.2021.3112126.

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, 2024. doi: 10.18653/v1/2024.findings-acl.701. URL https://aclanthology.org/2024.findings-acl.701/.

Zhehao Zhang, Ryan A. Rossi, Tong Yu, Franck Dernoncourt, Ruiyi Zhang, Jiuxiang Gu, Sungchul Kim, Xiang Chen, Zichao Wang, and Nedim Lipka. VipAct: Visual-perception enhancement via specialized VLM agent collaboration and tool-use. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

Shitian Zhao, Haoquan Zhang, Shaoheng Lin, Ming Li, Qilong Wu, Kaipeng Zhang, and Chen Wei. PyVision: Agentic vision with dynamic tooling, 2025.

Zihao Zhou and Rose Yu. Can LLMs understand time series anomalies? In *International Conference on Learning Representations*, 2025. arXiv:2410.05440.

APPENDIX CONTENTS

A **The DATAPATTERNBENCH Benchmark** . . . 14
A.1 Detection Split and Defect Classes . . . 14
A.2 Discovery Split . . . 14
A.3 Ground-Truth Validation and Scoring . . . 14
A.4 Controlled Perception Protocols . . . 15

B **Evaluation Details** . . . 15
B.1 Evaluation Sets and Metrics . . . 15
B.2 Agent Configurations . . . 17
B.3 Component Ablation Protocol . . . 18

C **Additional Experimental Results** . . . 19
C.1 Visual Perception across Base LLMs . . . 19
C.2 Comparing Evidence Representations . . . 20
C.3 Task Boundaries . . . 20

D **Analysis of Visual Evidence** . . . 21
D.1 Charts versus Textual Aggregates . . . 21
D.2 Rendering and Chart Batching . . . 22
D.3 Direct Image Delivery . . . 22

E **Formal Definition of the VDA Algorithm** . . . 23

F **Case Study** . . . 24

A THE DATAPATTERNBENCH BENCHMARK

A.1 DETECTION SPLIT AND DEFECT CLASSES

Each case starts from a real multi-table SQLite database from the Spider 2.0-lite local corpus (Lei et al., 2025). Files reach 300 MB, with up to 10^6 records per table. Using fixed random seeds, we plant two or three defects from the eight classes in Table 5 and record their types, tables, columns, and affected records. The detection split contains 46 development cases with 132 defects and 20 held-out cases generated with fresh seeds on the same 14 source databases. Development databases contain 1.6×10^5 to 2.4×10^6 records across 4–26 tables (Figure 3), with 1396 predefined views in total. Appendix B.1 reports the evaluation sets and results for each partition.

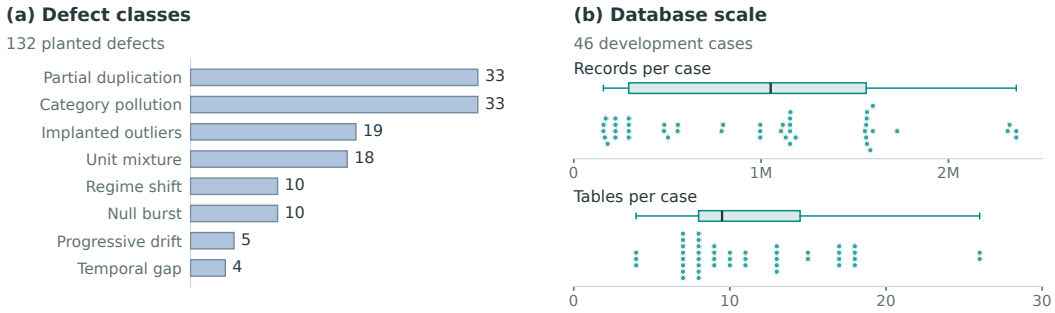


Figure 3: Composition and scale of the detection development split of DATAPATTERNBENCH. (a) Defects by type. (b) Records and tables per case. Dots show individual cases, offset vertically to avoid overlap. Boxes mark the middle 50%, center lines mark medians, and whiskers span the full range.

Table 5: The eight detection classes and their corresponding views.

Defect Type	Planted Content	Display
unit mixture	Rescale a subset of numeric values	Column distribution
implanted outliers	Insert a few extreme numeric values	Column distribution
category pollution	Insert variant category spellings	Top category counts
partial duplication	Duplicate a block of records	Duplicate counts
temporal gap	Delete records in a time window	Count by time bucket
null burst	Set values to null in a time window	Missing rate by time bucket
regime shift	Shift the level after a given date	Mean by time bucket
progressive drift	Add a gradual drift over time	Mean by time bucket

A.2 DISCOVERY SPLIT

The discovery split tests whether an agent can find temporal patterns that support analytical findings. We plant six types of patterns, including level shifts (1.6–2×), trend breaks, transient spikes (3–4×), volume spikes, seasonal rhythms, and categorical composition shifts. The split contains 34 development cases with 94 planted structures and 17 held-out cases with 47 structures, drawn from the same 9 source databases. An injection is accepted only if its verifier is false before the change and true afterward. We also apply the verifiers to the original database to identify pre-existing patterns. Findings that match these patterns are excluded from both precision and recall, rather than counted as false positives. Multiple structures may be reported for the same view.

A.3 GROUND-TRUTH VALIDATION AND SCORING

Injection checks. We fix the set of views before planting structures and restrict injection to the tables and columns they cover. Detection checks verify view coverage, reject identifier-like columns, and check properties relevant to each defect type, including the size of a shifted group, the number

of extreme values, the duplicate fraction, and the time resolution of a gap. All 132 development defects pass these checks. Every retained database passes SQLite integrity checks, and every annotated target satisfies its scoring predicate. These checks constrain the planted structures but do not guarantee that every defect is equally easy to recognize.

Scoring. A deterministic verifier checks submitted findings against predefined statistical conditions and filters those that do not satisfy them before scoring. Accepted findings are matched by structure type, table, and column without an LLM judge or partial credit. The verifier checks the reported structure and location, rather than every numerical claim or explanation in the response. All evidence conditions use the same verifier. Reported F1 therefore measures verified findings rather than unfiltered agent outputs. Appendix B.1 also reports scores before filtering candidate findings.

A.4 Controlled Perception Protocols

The agentic protocol provides SQL access, a schema, sampled records, and a 20-turn budget. The condition with visual perception adds booting views and the plotting tool, and the agent chooses what to inspect next. The fixed-view protocol instead makes one model call per view, holding the set of views, model, and verifier constant across evidence representations. The raw-record condition supplies 60 sampled values, while numerical summaries supply percentiles, frequent values, null counts, and other aggregates. Chart conditions receive images, with or without the accompanying summaries as specified in Table 10. Appendix D.1 separately tests a textual condition containing the full chart aggregates.

Both protocols provide the definitions of the target pattern types and use the same verifier before scoring. They test whether agents can identify these specified patterns, rather than find arbitrary data-quality problems. Models use their lowest supported reasoning setting unless stated otherwise, which does not equalize computation across model families. The end-to-end framework evaluation has separate budgets listed in Appendix B.2.

B Evaluation Details

B.1 Evaluation Sets and Metrics

DataPatternBench. Table 1 evaluates five systems with GPT-5.4 and GPT-5 on all 66 detection and 51 discovery cases. Detection contains 46 development and 20 held-out cases, and discovery contains 34 development and 17 held-out cases. The VDA configuration was selected on development data, so the combined scores are not independent held-out estimates. Table 6 separates the two partitions. Scores are mean per-case F1 after applying the shared verifier and matching findings by structure type, table, and column. All systems receive the same cases, definitions of target patterns, and scoring rule. Failed runs remain in the denominator and receive zero F1.

Before aggregating results, we require each database to pass SQLite integrity checks and every annotated target to satisfy its scoring predicate. Of the original 77 detection candidates, two contain corrupted tables and nine further cases have an annotation that fails its predicate. These 11 cases are excluded for every system and controlled comparison. All 51 discovery cases pass. The exclusions use only database and annotation checks, not model outputs. The supplementary manifest preserves the original cases and the reason for each exclusion. On the original detection set, GPT-5.4 scores 56.4% for OpenHands and 80.7% for VDA, compared with 59.8% and 84.1% on the retained set.

Scoring sensitivity. We also score the saved candidate findings before verifier filtering, retaining the same matching rule, failed-run treatment, and discovery exclusions. With GPT-5.4, OpenHands scores 56.7% detection F1 and 39.2% discovery F1, while VDA scores 80.2% and 70.8%. With GPT-5, the corresponding scores are 31.1% and 13.9% for OpenHands, and 48.0% and 35.9% for VDA. The advantage therefore remains when unsupported candidates count against precision. Restricting detection to the 41 cases whose target predicates are false before injection and true afterward gives 89.3% F1 for VDA and 64.6% for OpenHands with GPT-5.4. For the verified GPT-5.4 scores, an exact paired sign-flip test on the mean F1 of each source database gives $p < 0.001$ for detection and $p = 0.0039$ for discovery, using 14 and 9 database-level pairs. This groups cases from the same source rather than treating their injection seeds as independent databases.

Table 6: Partition-specific F1 (%) for the complete evaluations in Tables 1 and 2. Development and held-out scores are reported separately.

Model	System	Det. dev	Det. held	Disc. dev	Disc. held
GPT-5.4	Zero-shot	1.1	0.0	0.0	0.0
GPT-5.4	OpenHands	62.4	53.8	44.0	41.2
GPT-5.4	MiroFlow	63.3	65.0	40.5	35.8
GPT-5.4	smolagents	55.5	51.7	25.0	40.0
GPT-5.4	VDA	85.2	81.6	77.4	79.2
GPT-5.4	smolagents + VDA	68.3	65.9	53.2	61.0
GPT-5	Zero-shot	2.0	0.0	0.0	0.0
GPT-5	OpenHands	34.2	38.4	15.2	25.9
GPT-5	MiroFlow	38.7	37.4	16.0	26.4
GPT-5	smolagents	24.9	27.1	11.7	18.0
GPT-5	VDA	60.3	54.1	40.3	48.0

Deterministic checks. We enumerate candidate pattern types and locations within the same predefined views and apply the benchmark's statistical checks without an LLM. The detector does not read case annotations, but it uses the acceptance predicates themselves, so it is specific to these known pattern classes. On the 66 detection cases it achieves 63.8% F1, 50.5% precision, and 100% recall. On the 51 discovery cases it achieves 77.6% F1, 65.6% precision, and 100% recall, with the same pre-existing-pattern exclusions. Perfect recall is expected because retained targets must satisfy these checks. The comparison measures how many additional locations the rules flag. VDA is more selective on detection, but its 78.0% discovery F1 is close to the rule detector. This limits what discovery scores alone establish about the need for an agent.

Previously unused databases. We freeze the agents and attempt two injection seeds on each of seven source databases not used in development or held-out evaluation. Five cases from three databases pass the predeclared checks for database integrity, newly introduced defects, and changed views. No replacement seeds or model outputs enter this selection. On these five cases, GPT-5.4 achieves 86.0% F1 with VDA and 82.7% with OpenHands. VDA wins on two cases, loses on one, and ties on two. This small check retains the same defect classes and construction rules and does not establish broad generalization to new databases or naturally occurring defects.

Public benchmarks. The seven public benchmarks comprise four localization tasks, two classification tasks, and one insight-discovery task. Their evaluation sets contain 250 UCR, 52 NAB, 29 TCPD, 60 ESA-ADB, 100 CinC-2017, 120 WM-811K, and 100 InsightBench cases. The two models and five systems in Table 1 are evaluated on all 711 cases, yielding 7110 outcomes. Pooled localization and classification accuracy covers 611 cases and excludes InsightBench, which uses a different metric. These sets do not cover every item in the original source datasets. Localization uses benchmark-specific acceptance windows, classification uses exact label accuracy, and for InsightBench a gold insight counts as recalled when a GPT-4.1-mini matcher finds the same finding among the agent's insights, in place of the official LLaMA-3-70B G-Eval score. The matcher checks the variables and direction of the finding, allows numerical differences of approximately 20%, and considers at most ten generated insights per case. Failed outcomes remain in the denominator. Infrastructure failures are retried under the original settings, and successful runs are not repeated based on their scores.

Case selection. UCR includes all 250 source series. NAB includes all 52 series with usable anomaly windows, excluding six series without such windows. TCPD includes all 29 locally available annotated univariate series with at least one change point. ESA-ADB uses the ten target channels with the most annotations in Mission 1 and up to six valid event windows per channel, processed in chronological order. CinC-2017 uses the first three records per class in the reference file as labeled examples and the next 25 as evaluation queries. WM-811K uses eight defect classes and maps whose height and width are each between 20 and 60 cells. In source order, the first three eligible maps per class provide labeled examples and the next 15 provide evaluation queries. InsightBench includes all 100 tasks in the local benchmark collection.

B.2 AGENT CONFIGURATIONS

Models and budgets. The main comparisons use reasoning effort `none` for GPT-5.4 and `minimal` for GPT-5, except that public OpenHands and smolagents runs retain GPT-5's provider default in both conditions. In the DATAPATTERNBENCH framework evaluation, each case has an 80-request and 1500-second cap. OpenHands permits 40 iterations per invocation, smolagents 40 steps per invocation, and MiroFlow 40 turns. Zero-shot makes one LLM API request. MiroFlow uses its native agent loop with a local shell tool and no subagents for these local-data tasks. Tool instructions follow the API's native tool-call format, and the loop checks that the requested output file has been produced before finishing, within the same turn and request limits. The evaluated MiroFlow tool interface returns text, so chart images are not delivered directly to its main agent. We do not report a visually augmented MiroFlow condition because it would require modifying this interface. Public-benchmark runs use the following iteration limits. OpenHands permits 40 iterations per invocation for localization and 30 for classification and InsightBench. Smolagents permits 20, 15, and 25 steps, respectively. MiroFlow permits 20 turns with a 2400-second subprocess limit.

Recorded resource use. Across the 117 database cases with GPT-5.4, OpenHands averages 11.2 model requests and 200k cumulative API tokens per case, compared with 27.3 requests and 1342k tokens for VDA. These token totals include repeated context and cached input, so they are not billing estimates. The systems share maximum budgets but use different amounts of computation. The reported gains do not establish improved cost efficiency.

VDA on DATAPATTERNBENCH. The main results on DATAPATTERNBENCH use all components tested in the ablation study. Booting supplies up to 64 views in batches of 3, and the observation window retains up to 40 chart images. Coverage tracks successfully delivered booting views and views requested through the plotting tool. The evaluated memory configuration stores agent notes extracted from assistant messages and indexed by view identifiers.

Booting rules. Booting views are selected from the database before the agent starts. Tables with at least 5000 records are considered in descending order of size. Columns whose names mark identifiers or codes, such as names ending in `id` or containing `zip`, `phone`, or `hour`, are excluded. A text column counts as a date column when its values parse as dates and span more than 90 days, and each table uses its first date column. A numeric measure is a numeric column that is not identifier-like and whose non-missing values are mostly positive, and each table uses at most two measures. A table with a date column receives a record-count trend, a mean trend for each measure, and a missing-rate trend for the first measure. When such a table also has a text column with 2 to 30 distinct values, each measure additionally receives a mean trend split by that column. Every eligible table receives a view of identifier values against record position and a count of exact duplicate records. Every remaining numeric or text column receives a distribution profile. Views over time and record position come first, followed by duplicate counts and profiles, and at most 36 views are selected. The discovery split additionally includes category-composition views, bringing some inventories above 36 views. The booting budget of 64 accommodates these inventories, but views that fail to render are skipped. Coverage records successful delivery rather than scheduled views. If views remain undelivered after the initial analysis, the agent receives one coverage reminder before completing the task.

Booting example. The Delivery Center database of the case study in Appendix F has seven tables. Only `payments`, `deliveries`, and `orders` have at least 5000 records, so the other four tables are skipped. None of their text columns parses as a date, so no time-series views are selected. Identifier-like columns such as `payment_id` and `order_created_hour` are excluded. The resulting set contains 31 views, one identifier-position view and one duplicate count per table, followed by 25 column profiles such as `payment_amount`, `delivery_distance_meters`, and `order_status`. All 31 views are delivered during booting in batches of three, and the `payment_amount` profile is the chart in which the agent notices the second payment cluster.

VDA on public benchmarks. On the six localization and classification tasks, the agent receives overview charts and does not use persistent visual memory. Series tasks also provide a plotting tool and coverage feedback. InsightBench is evaluated as a database-analysis task.

Table 7 compares the overview configuration with a variant that enables booting views and memory at GPT-5.4. Their pooled scores are close, but the effects differ by task. The variant improves TCPD and ESA while reducing CinC accuracy. Because it changes booting and memory together, this comparison does not isolate either component or establish equivalence between the configurations. Table 1 uses the overview configuration for both models. Both configurations use the same 611 cases and are evaluated separately from the component ablation.

Table 7: Public OpenHands configurations at GPT-5.4, accuracy (%).

Configuration	UCR	NAB	TCPD	ESA	CinC	WM	Pooled
Overview (main tables)	32.4	78.8	86.2	86.7	55.0	59.2	53.2
Booting views + memory	33.6	86.5	96.6	90.0	36.0	63.3	52.9

Cross-framework comparison. Table 2 uses the same 66 detection, 51 discovery, and 611 public cases as Table 1. The OpenHands baseline, OpenHands with VDA, and smolagents baseline reuse the corresponding Table 1 runs. For smolagents with VDA, we evaluate all 117 database cases using the same renderer, view inventories, booting and image budgets, and coverage rules as OpenHands with VDA. Charts are delivered through the native smolagents image interface, and the agent's notes are saved in external text memory. Both smolagents conditions retain its CodeAgent tools and the same findings-submission tool. The model, task prompts, request and time limits, verifier, and scoring rule are shared across frameworks. The two failed smolagents runs with VDA receive zero F1. The public results reuse the completed runs on the same fixed case sets, including failed outcomes in the denominator.

B.3 Component Ablation Protocol

Shared settings. Table 3 compares full VDA, seven component removals, and OpenHands on 46 detection and 34 discovery development cases. GPT-5.4 uses reasoning effort `none`, temperature 0, and a 4096-token output limit, with 40 iterations per invocation and 80 requests or 1500 seconds per case. Booting and image budgets follow Appendix B.2. Component removals share prompts, views, data, and scoring except for the specified change. The full configuration was selected on development data before ablation. Two of 720 runs fail and receive zero F1. OpenHands retains its original tools, including the ability to compute and plot.

Component removals. Removing *booting views* omits initial charts and their note requests but retains the plotting tool. Removing the *plotting tool* omits the tool and its prompt instructions but retains booting views and general tools. Removing the *numerical summary* omits numerical text and summary-only views but retains chart labels and numerical computation.

Removing *agent notes* disables note requests and automatic saving, leaving chart batches and delivery timing unchanged. The memory file starts empty and remains available for independently written notes. Removing *coverage feedback* disables coverage reminders. Removing *visual memory* disables external note storage but retains note requests and notes in context. Removing *rendering calibration* disables logarithmic binning, integer-bin alignment, calendar completion, and axis adjustment for extreme temporal peaks in all charts. Chart templates and other aggregation rules remain unchanged.

Public benchmarks. The public ablations use all 250 UCR, 52 NAB, 29 TCPD, 60 ESA-ADB, and 120 WM-811K cases. Unlike the overview configuration used in Table 1, these experiments enable booting views, agent notes, and persistent text memory, with each component controlled independently. The public scores in Table 3 therefore compare component removals within this configuration. Settings and case limits follow the database ablation above. All public conditions, including OpenHands, are run separately under these settings rather than reusing the main-table runs. All conditions retain the underlying data and general tools. Three failures among the 4338 outcomes are scored as incorrect.

For localization, booting supplies a full-series chart, the plotting tool accepts further intervals, and numerical summaries describe the viewed interval. Rendering calibration divides intervals longer

than 42000 points into multiple charts, affecting 100 UCR series. UCR scores include all 250 cases. NAB, TCPD, and ESA contain no series above this threshold, so their calibration removals are marked N/A. For WM, booting supplies charts of examples and the query for each candidate class in batches of three. Wafer maps have no separate calibration transformation, so that removal is also N/A. Coverage feedback reports viewed intervals or candidate classes and allows further investigation without requiring exhaustive viewing. Visual memory stores task-specific notes, while its removal retains notes already in context.

C Additional Experimental Results

C.1 Visual Perception across Base LLMs

DataPatternBench. Table 8 provides the scores behind Figure 2, left. Each pair uses the same agent and detection development cases, with the model at its lowest supported reasoning setting. All ten models use the same 46 paired cases. The separate reasoning-effort comparison appears in Table 4.

Table 8: Detection F1 (%) with and without visual perception across ten LLMs on DataPatternBench.

Model	w/o Visual Perception	w/ Visual Perception
GPT-4o-mini	1.4	**54.1**
GPT-4o	13.8	**66.6**
GPT-4.1-mini	19.6	**71.2**
GPT-5-mini	23.8	**59.8**
GPT-4.1	22.8	**71.7**
Gemini 3.1 Flash-Lite	29.5	**77.4**
GPT-5.4-mini	33.2	**62.8**
GPT-5	36.2	**72.3**
GPT-5.4	55.3	**71.3**
Gemini 3.7 Flash	80.9	**89.5**

Adding numerical summaries to the agent without visual perception gives F1 of 54.8%, 45.9%, and 28.3% at GPT-5.4, GPT-5, and GPT-4o, respectively. These scores remain below the corresponding scores with visual perception in Table 8.

Public benchmarks. Table 9 separates the localization and classification results pooled in Figure 2, right. The comparison covers eight GPT models under the observation protocols in Appendix C.2. Every model and condition uses the same 391 localization and 220 classification cases. Failed runs are incorrect. Charts with numerical summaries achieve or tie the highest accuracy in 11 of the 16 model-task groups. Charts alone lead in the remaining five.

Table 9: Public-benchmark accuracy (%) across eight base models under the controlled observation protocol. Raw denotes sampled records, num. denotes numerical summaries, and both combines charts and summaries. Best within each model-task group is bold.

	localization (4 benchmarks)				classification (2 benchmarks)			
Base model	raw	num.	chart	both	raw	num.	chart	both
GPT-4o-mini	19.2	29.2	19.2	**30.7**	26.4	33.6	36.4	**37.3**
GPT-4o	27.6	35.8	34.5	**39.1**	38.6	43.6	**51.4**	48.2
GPT-4.1-mini	26.6	34.0	**39.9**	38.1	29.1	40.9	**46.8**	45.9
GPT-4.1	31.2	38.6	**41.7**	**41.7**	33.6	45.5	52.3	**55.0**
GPT-5-mini	27.1	33.8	**38.4**	37.6	28.6	41.4	47.7	**48.6**
GPT-5	32.2	38.6	42.2	**46.3**	33.6	43.6	45.0	**50.9**
GPT-5.4-mini	33.2	38.1	**48.8**	47.6	25.9	36.8	**47.3**	**47.3**
GPT-5.4	40.9	45.3	48.3	**49.9**	45.9	50.5	53.6	**56.8**

C.2 Comparing Evidence Representations

This study compares sampled records, numerical summaries, charts, and charts with summaries. Table 10 reports GPT-5.4 results by benchmark. The protocols control the observations available to the model and differ from the framework evaluation in Table 1.

DataPatternBench. We use the fixed-view protocol in Appendix A.4, with one model call per view and no agent selection of views. The comparison uses the same 46 detection and 34 discovery development cases as the component ablation. Charts and numerical summaries are computed from the same data aggregates.

Public localization. The model first observes the full series and has up to six turns to inspect an interval or predict a location. Records provide up to 800 uniformly sampled index and value pairs. Summaries provide bucketed means, standard deviations, minima, and maxima. Charts provide a line plot, and interval requests return the same representation for that interval. A prediction is correct within ± 100 of the labeled interval on UCR and ESA-ADB, inside a labeled window on NAB, or within ± 5 of an annotated change point on TCPD.

Public classification. Each item uses one model call with three labeled examples per class. Charts show the examples and query together, while textual conditions provide the corresponding records or numerical summaries. The combined condition adds numerical summaries to the chart. All reported scores use the first run of each model and condition, including failed runs. Additional repeated runs are preserved in the supplement and are not substituted for first-run outcomes.

Table 10: Chart and numerical summary decomposition at GPT-5.4. DataPatternBench uses F1 (%) and public tasks use accuracy (%). Best per row is bold.

Benchmark	Raw	Summary	Chart	Summary + Chart
DataPatternBench (detection)	60.1	70.7	75.6	**76.2**
DataPatternBench (discovery)	64.7	63.3	**83.1**	83.1
UCR-Anomaly	22.0	23.2	26.8	**28.4**
NAB	57.7	76.9	**80.8**	**80.8**
TCPD	86.2	86.2	96.6	**100.0**
ESA-ADB	83.3	**90.0**	86.7	88.3
CinC-2017	36.0	38.0	**41.0**	40.0
WM-811K	54.2	60.8	64.2	**70.8**

Charts with summaries achieve the highest score on DataPatternBench detection and achieve or tie the highest score on four of the six public benchmarks. On discovery, charts alone and charts with summaries both round to 83.1%. Numerical summaries lead on ESA-ADB and charts alone on CinC-2017. The fixed-view detection comparison also favors charts with summaries at GPT-5 and GPT-4o, with F1 of 73.9% and 69.0%, compared with 63.8% and 62.8% for summaries alone. This F1 gain does not imply lower token use. Across GPT-5.4, GPT-5, and GPT-4o, charts with summaries use 37–40k tokens per detection case, compared with 22–23k for summaries alone. Providing the complete chart data as text is evaluated separately in Appendix D.1.

C.3 Task Boundaries

DataGovBench (Hasegawa et al., 2026) poses questions over government open data, with textual or visualization answers. We compare OpenHands with and without a plotting tool at GPT-5.4-mini on all 195 task files containing non-visualization questions, totaling 315 questions. Both conditions use the same questions, tolerant answer matcher, and maximum budgets of 100 requests and 1500 seconds per file, with 25 iterations per invocation. Both receive the original CSV files and the same SQLite tables. The shared CSV parser preserves numeric types, removes byte-order marks, and fills missing trailing fields with null values. We use the corrected parser for both conditions on all 42 affected task files. The supplement records the correction and superseded outputs.

Mean within-file accuracy is 47.4% without the plotting tool and 40.1% with it. Pooled question accuracy is 45.4% and 40.3%, respectively. The baseline wins on 37 files, the augmented agent on

20, and the remaining 138 are ties. One baseline run reaches the time limit and receives zero accuracy. Its action log is unavailable, so action counts use the 194 pairs with complete logs, averaging 6.7 actions without plotting and 8.5 with plotting. Tool observations are not counted as additional actions. Excluding the six task files whose rows were omitted by the original parser gives mean accuracy of 47.6% and 40.7%. Adding a plotting tool therefore increases the recorded work and reduces accuracy in this question-answering comparison.

D Analysis of Visual Evidence

D.1 Charts versus Textual Aggregates

Complete text in the full agent. We repeat full VDA with GPT-5.4 on the same databases, replacing each chart observation with all values used to draw it. Booting views, rendering calibration, numerical summaries, agent notes, memory, coverage feedback, and maximum budgets remain unchanged. The text condition uses new runs, while the chart condition reuses the main evaluation runs. General tools remain available in both conditions. Complete text preserves the plotted values exactly and can provide more numerical precision than the image.

Table 11: F1 (%) with charts or complete chart data as text in the full agent. Failed runs receive zero F1 in the full evaluation set.

	Full evaluation set		Both runs completed	
Evidence	Detection	Discovery	Detection	Discovery
Charts	84.1	78.0	83.7	75.8
Complete chart data as text	77.6	64.6	81.3	76.6

The full evaluation includes 66 detection and 51 discovery cases (Table 11). The text condition has 11 failed runs, including two confirmed context-limit errors and nine HTTP errors whose causes were not recorded. The chart condition has no failed runs. We therefore also report the 63 detection and 43 discovery cases completed by both conditions. This additional comparison was introduced after failures were observed and conditions on successful completion. A paired sign-flip test swaps the two conditions jointly within each source database while retaining per-case weights. It does not establish a difference in either split, for either set of cases. Across all cases, charts use 1.34 million recorded tokens per case and complete text uses 1.92 million. On completed pairs, the corresponding means are 1.30 and 1.78 million. These totals include repeated context tokens and do not measure monetary cost. Failed responses may omit token usage. The results support a more compact representation with charts, but do not establish better pattern detection than equally complete text.

Complete text with fixed views. This comparison tests whether charts help because they preserve more information than summaries or because they present that information visually. Summary statistics provide numerical information such as counts, means, percentiles, value ranges, correlations, and group means, depending on the view. A separate condition supplies all values used to draw each chart as text, including histogram counts, group means, and heatmap cell values. The study uses 65 databases with 106 planted structures across eight classes designed to be missed by particular summary statistics (Anscombe, 1973; Matejka & Fitzmaurice, 2017). All conditions receive the same fixed set of views (Table 12).

At GPT-5.4, complete chart data as text raises F1 from 21.1% with summary statistics to 55.3%, compared with 55.2% for charts with summaries. Charts also outperform complete chart data as text at GPT-4o (54.3% versus 50.4% F1) and GPT-5 (55.8% versus 52.0%). These results show that omitted information explains much of the gap against summary statistics. Presenting these values as charts provides a further benefit for GPT-5 and GPT-4o.

The benefit also varies by structure type. For Simpson reversals, where the overall trend reverses within individual groups, GPT-5.4 identifies $4/12$ cases from charts and $12/12$ from complete chart data as text. This result shows that charts do not consistently outperform sufficiently detailed text.

Table 12: F1 and recall (%) on 65 databases with 106 planted structures, using the same views across conditions. Best per model and metric is bold.

	GPT-5.4		GPT-5		GPT-4o	
Evidence	F1	Recall	F1	Recall	F1	Recall
sampled records	31.6	28.2	39.3	54.2	43.5	68.5
summary statistics	21.1	19.0	36.3	36.2	45.4	59.7
complete chart data as text	**55.3**	**73.8**	52.0	88.5	50.4	86.9
chart with numerical summary	55.2	68.8	**55.8**	**89.2**	**54.3**	**90.8**

D.2 Rendering and Chart Batching

Rendering calibration. Two targeted chart-reading checks examine calibration on the 46 detection development cases. At GPT-5.4, removing logarithmic binning reduces unit-mixture recall from $12/18$ to $4/18$. Removing calendar completion reduces temporal-gap recall from $3/4$ to $1/4$. Both changes affect whether the relevant pattern is visible in the chart.

Views per call. Figure 4 varies the number of views in a model call while holding their content and scoring rule fixed. At GPT-4o, recall falls from 94.7% with one view per call to 83.3% with two and 81.1% with eight. GPT-5.4 is less sensitive, with recall between 85.6% and 92.4% across the tested settings. At GPT-4o, the rate of reporting structures on views without a planted structure also falls from 60.4% with one view to 29.5% with all views in one call. Larger batches therefore reduce reporting on both kinds of views in this setting. The results favor smaller batches for some models, without identifying one best batch size for every model.

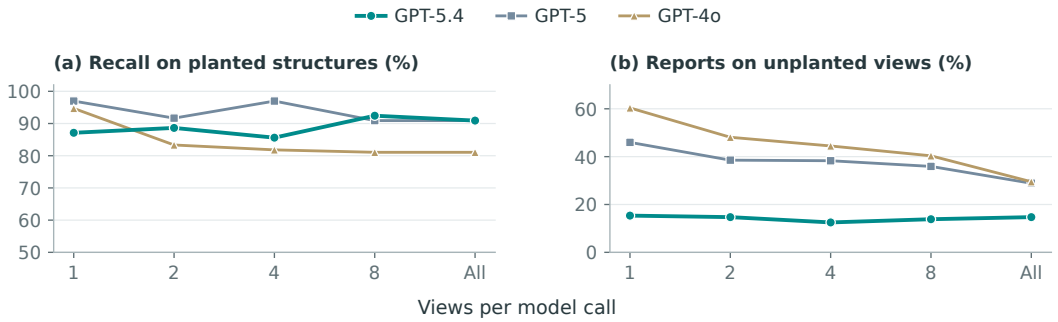


Figure 4: Effect of the number of views per model call, with view content and the scoring rule held fixed.

D.3 Direct Image Delivery

On the 20 detection held-out cases, we compare an OpenHands agent with an on-demand plotting tool against a version that receives generated descriptions of its charts. All three conditions, including the baseline without plotting, use separate runs with a 60-iteration limit per invocation, rather than the 40-iteration limit in the main framework evaluation. This study does not use booting views, agent notes, visual memory, or coverage feedback. The same model generates each description in a separate call with an 800-token output limit, and both conditions retain the numerical summary. At GPT-5.4, F1 is 63.8% with direct images and 42.3% with descriptions, compared with 64.5% without the plotting tool. At GPT-5.4-mini, the image and description conditions score 27.5% and 27.8%, respectively. Generated descriptions can therefore lose useful information, but this comparison does not show a consistent benefit from the plotting tool alone. Appendix D.1 separately tests text containing all plotted values.

E FORMAL DEFINITION OF THE VDA ALGORITHM

Algorithm 1 defines the database workflow of VDA. Its annotations refer to the circled panels in Figure 1. The procedure Update maintains the observation window, links notes to their views, and applies the image budget B. Chart text remains in context after its image is removed. General tool outputs follow the host agent's context management. The evaluated configuration retrieves text notes through the original tools. It sends coverage feedback at most once, after the initial analysis.

Algorithm 1 The VDA algorithm. Booting views are shown individually for clarity. The implementation supports batches.

Require: environment $\mathcal{E}$; agent π with tools $\mathcal{A}$; booting budget N; image budget B
Ensure: the agent's answer y, or failure if no answer is submitted

$\mathcal{V} \leftarrow$ rule-based view inventory; $\mathcal{V}_{\mathrm{boot}} \leftarrow \mathrm{First}_N(\mathcal{V})$ ▷ Ⓐ
$c \leftarrow$ task prompt; $R \leftarrow \emptyset$; $\mathrm{Seen} \leftarrow \emptyset$; $f \leftarrow \mathrm{false}$
for each $v \in \mathcal{V}_{\mathrm{boot}}$ **do**
    $(I_v, S_v) \leftarrow \mathrm{Render}(v)$ ▷ Ⓑ skip unsuccessful renders
    $c \leftarrow \mathrm{Update}(c, (I_v, S_v), R; B)$; $\mathrm{Seen} \leftarrow \mathrm{Seen} \cup \{v\}$ ▷ Ⓒ–Ⓔ
    $r_v \leftarrow \pi(c)$ ▷ task-specific notes on the delivered view
    $c \leftarrow c \oplus r_v$; persist (v, r_v) in R
end for
while the task permits further actions **do**
    $a \leftarrow \pi(c)$
    if $a = \textsc{View}(v)$ **then**
        $o \leftarrow \mathrm{Render}(v)$ ▷ Ⓖ plotting tool applies Ⓑ
        add successfully delivered v to Seen
    else
        $o \leftarrow \mathcal{E}(a)$ ▷ general tools, including memory access
    end if
    $c \leftarrow \mathrm{Update}(c, (a, o), R; B)$ ▷ Ⓒ–Ⓔ associate notes with their views
    if a submits an answer y **then**
        if $\neg f$ and $\mathcal{V} \setminus \mathrm{Seen} \neq \emptyset$ **then**
            $c \leftarrow c \oplus \rho(\mathrm{Seen})$; $f \leftarrow \mathrm{true}$ ▷ Ⓕ one coverage reminder
        else
            return y
        end if
    end if
end while
return the latest submitted answer, or failure if none is available

F CASE STUDY

A delivery platform needs to check its payment, delivery, and order records before using them for analysis. The agents are asked to find injected defects from eight predefined classes, without being told which columns contain them. We follow paired GPT-5.4 runs of OpenHands and VDA on the same Delivery Center case of DATAPATTERNBENCH, with 400,834 payment records and 378,843 delivery records. The case uses the controlled defects described in Appendix A. The account below summarizes saved agent notes and final responses, with charts reconstructed from the saved numerical summaries.

Using its original tools, OpenHands reports that the delivery status contains two frequent labels, `DELIVERED` and `DELIVERED_`, which differ only by a trailing underscore. This finding is correct. It also reports a drop in the delivery fee from 6.90 to 4.90, but the benchmark verifier cannot confirm the proposed temporal shift. Its response leaves the payment amounts and delivery distances unflagged, missing two of the three defects in the case.

With VDA, the booting views bring these distributions into the agent's observation window. The payment chart shows a large concentration of small payments and a separate cluster in the thousands (Figure 5a). Rendering calibration uses logarithmic bins so that both groups are visible despite the wide range of values. The agent's saved notes identify the second cluster as a possible unit mixture. Its final response gives concrete evidence for that finding, identifying 44,819 payments from 2000 to below 20,000, or 11.18% of payment records. The separated cluster provides a reason to investigate mixed units that a large maximum alone would not establish.

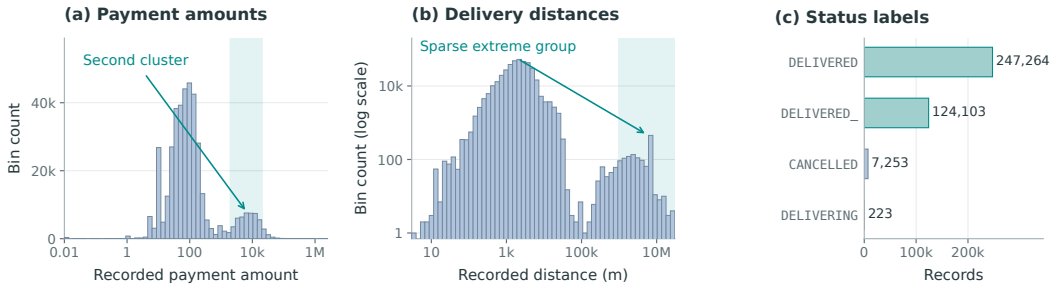


Figure 5: Charts from the Delivery Center case. (a) A second payment cluster. (b) A sparse group of extreme delivery distances, with counts on a logarithmic axis. (c) Frequent status labels differing only by an underscore. Histograms show positive values.

The delivery-distance chart presents a different pattern (Figure 5b). Most distances cluster in the thousands of meters, while a small group lies far away in the millions. The agent notes this separation and reports 1194 distances above one million meters, accounting for just 0.315% of delivery records. It identifies these as extreme outliers, distinguishing the sparse group from the substantial second cluster in payments. The two charts thus support different diagnoses even though both columns contain unusually large values.

The status chart makes the remaining problem explicit (Figure 5c). The labels `DELIVERED` and `DELIVERED_` account for 247,264 and 124,103 records, respectively. VDA records them as a category split caused by a trailing underscore, agreeing with the valid finding in the OpenHands response.

Before completing the task, VDA retrieves its saved notes. Its final response reports the payment unit mixture, the extreme delivery distances, and the inconsistent status labels, all three accepted by the verifier and matched to the case annotations. OpenHands finds the label inconsistency but misses both numerical defects. In this case, the charts help the agent identify which numerical patterns warrant investigation and distinguish their types, while the accompanying values support its final findings.