DEEPCHART: How Far are LLMs from Faithful Data-Science Chart Generation?

Jiahui Tang[1], Kuicai Dong[2], Dexun Li[2], Hongchao Gu[1], Haocheng Yu[1]
Wei Han[2], Chen Zhang[2], Yong Liu[2], Hao Wang[1]*, Enhong Chen[1]
[1]University of Science and Technology of China
[2]Huawei Technologies Co., Ltd.

Abstract

Faithful chart generation in real-world data-science workflows requires grounding visualizations in scattered evidence, computing chart-ready quantities, and rendering them accurately. Modern LLMs can produce visually plausible, instruction-compliant charts, yet data-level hallucinations remain difficult to detect in long, noisy, and multimodal contexts. To measure this gap, we introduce DEEPCHART, an expert-annotated benchmark of 1,482 task-conditioned chart-generation instances drawn from real-world scientific papers, financial filings, and ecosystem reports. DEEPCHART formulates chart generation as an Extract–Reason–Visualize pipeline and evaluates source-data extraction, derived-data reasoning, and chart rendering stage by stage. Experiments with state-of-the-art models show that visually plausible charts often conceal data-level hallucinations, with extraction and reasoning errors common in realistic long and multimodal settings. These findings suggest that larger context windows alone are insufficient; faithful chart generation also requires reliable evidence extraction and quantitative reasoning before rendering. Our benchmark and associated resources are available at https://github.com/tangdouer1005/DeepChart.

1 Introduction

Charts are a primary medium for communicating data-science findings (Munzner, 2025). In real-world workflows, however, producing a faithful chart is rarely a matter of plotting a ready-made table. An analyst must sift through extensive, heterogeneous documents, extract scattered data points, apply non-trivial transformations, and only then render the result as a visualization. As LLMs are increasingly used as data-analysis agents (Singh et al., 2025; Jin et al., 2025), their outputs must move beyond textual answers to communicate derived findings through faithful visualizations (Huang et al., 2025; Dong et al., 2025). This raises a challenge that is easy to overlook: *the data faithfulness of a chart cannot be observed from the chart itself*. A figure rendered from mis-extracted or miscomputed numbers can still look visually plausible. The act of rendering conceals the errors that precede it. We call such surviving errors *hidden hallucinations*: subtle mistakes in extraction or reasoning that pass through rendering and evade conventional, appearance-based evaluation.

Current chart-generation benchmarks are ill-equipped to expose hidden hallucinations because they often map clean inputs directly to final charts and score only the endpoint. As summarized in Table 1, this gap appears in three aspects. **(1) *Pre-digested inputs.*** Existing benchmarks supply clean tables or short snippets rather than the large, heterogeneous, multimodal documents of real analysis, removing or simplifying the stages where hidden hallucinations originate. **(2) *Missing intermediate data references.*** Generation is treated as a one-shot prompt-to-chart mapping, with no reference source values or derived quantities for inspecting what happened before rendering. **(3) *Endpoint evaluation.*** Consequently, final-chart scoring can neither verify the underlying numbers nor localize failures to extraction, reasoning, or rendering.

To detect hidden hallucinations, we model chart generation as a **multi-stage analytical pipeline** that captures key data-science bottlenecks, as illustrated in Figure 1. Inspired by the classical ETL (Extract→Transform→Load) workflow (Khan et al., 2024), we decompose this pipeline into ***Extract*** (retrieve intent-relevant evidence), ***Reason*** (derive chart-ready quantities), and ***Visualize*** (render faithful charts). This decomposition enables failure isolation across extraction,

*Corresponding author.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 37005–37031
October 24-29, 2026 ©2026 Association for Computational Linguistics

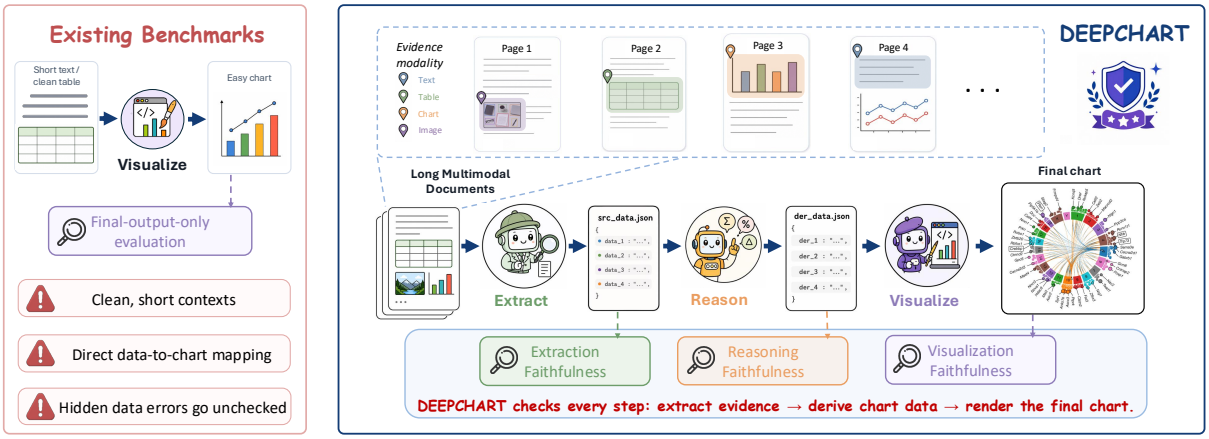


Figure 1: **DEEPCHART task overview.** Compared with visualize-only chart benchmarks, DEEPCHART evaluates chart generation as an Extract–Reason–Visualize pipeline over long multimodal documents.

reasoning, and visualization.

Building on this formulation, we introduce DEEPCHART, an expert-annotated benchmark for faithful data-science chart generation. DEEPCHART contains task-conditioned context-query instances drawn from real-world documents across text and multimodal protocols. Its text inputs average 220.6K tokens, its multimodal report inputs average 218 pages, and the average instance draws on 317.6 context-sourced data points. Beyond direct data-to-visual mapping, each instance requires the *ERV* pipeline: models must extract intent-relevant evidence, derive chart-ready data, and then render the chart. We instantiate this design across Academic-Normal/Long, Finance-Normal/Long/Ultra-Long, and Ecosystem report-level multimodal settings, covering both context-scale variation and native multimodal document inputs. To expose hidden hallucinations, DEEPCHART provides stage-by-stage evaluation aligned with the *ERV* pipeline: ***Source Data Fidelity*** for extraction, ***Derived Data Fidelity*** for reasoning, and ***Visual Accuracy Score*** **(VAS)** for rendered-chart faithfulness.

Zero-shot evaluation on DEEPCHART shows that (1) visually plausible charts often conceal data-level hallucinations, (2) models struggle with both Extract and Reason stages, and (3) full-document inputs further degrade end-to-end faithfulness, suggesting that larger context windows alone are insufficient. In summary, our contributions are:

- **Task formulation.** We formalize data-science chart generation as an *ERV* pipeline, exposing the coupled challenges of evidence retrieval, analytical reasoning, and faithful rendering.
- **DEEPCHART benchmark.** We release an expert-annotated benchmark over long-text and report-level multimodal contexts, with references and metrics for source values, derived quantities, and final charts.
- **Empirical findings.** We show that SOTA models suffer from hidden hallucinations, persistent extraction and reasoning challenges, and degraded faithfulness under full-document inputs.

2 Task Definition

Given a long, heterogeneous analytical context C and a natural-language chart intent Q, the task is to generate a faithful data-science chart. The context may contain textual descriptions, tables, figures, or document pages, and the chart intent specifies the analytical goal and desired visualization. Unlike direct prompt-to-chart generation, the pipeline is expected to expose the data path that supports the final visualization: the source values extracted from the context, the chart-ready quantities derived from those values, and the executable visualization that renders them.

We formulate this process as an Extract–Reason–Visualize (ERV) pipeline:

$$D_{\text{src}} \leftarrow \text{Extract}(C, Q), \quad (1)$$

$$D_{\text{der}} \leftarrow \text{Reason}(D_{\text{src}}, Q), \quad (2)$$

$$(P, G) \leftarrow \text{Visualize}(D_{\text{src}}, D_{\text{der}}, Q). \quad (3)$$

Here, D_{src} denotes extracted source data, D_{der} denotes derived chart-ready data, P denotes the executable chart-generation program, and G denotes the rendered chart.

We define the auditable intermediate state as $J = (D_{\text{src}}, D_{\text{der}})$. Exposing J makes a gener-

Method	Input Setting			Task Requirement		Evaluation Granularity	
	Context	Modality	Real-doc	Retrieval	Reasoning	Visual	Stage-by-stage
Text2Chart31 (Pesaran Zadeh et al., 2024)	<16K	Txt+Tab	✗	✗	✗	✓	✗
MatPlotBench (Yang et al., 2024)	<32K	Tab	✗	✗	✓	✓	✗
Text2Vis (Rahman et al., 2025)	<2K	Tab	✗	✓	✓	✓	✗
C^2/ChartUIE-8K (Koh et al., 2025)	<32K	Tab	✗	✗	✓	✓	✗
Doc2Chart (Jain et al., 2025)	<32K*	Txt+Tab	✓	✓	✓	✓	✗
Infogen (Ghosh et al., 2025)	<2K*	Txt	✗	✓	✗	✓	✗
PlotCraft (Zhang et al., 2026)	<128K*	Tab	✓	✗	✓	✓	✗
DEEPCHART	220K tokens / 218 pages .avg	Txt+Tab+Img	✓	✓	✓	✓	✓

Table 1: **Comparison with existing chart-generation benchmarks. Context** reports approximate input scale; starred values are estimated when complete statistics are unavailable. **Retrieval** means locating task-relevant data in complex contexts; **Reasoning** means deriving chart-ready quantities from extracted data. **Real-doc** indicates contexts are constructed from real-world documents, and **Stage-by-stage** indicates evaluation of intermediate data rather than only the final chart.

ated chart traceable: we can verify whether it is grounded in the correct source values and whether those values are transformed into the correct chart-ready quantities before rendering. This enables stage-by-stage evaluation across the Extract, Reason, and Visualize stages, rather than assessing only the final chart. With this design, DEEPCHART can distinguish extraction, reasoning, and visualization failures, thereby exposing hidden hallucinations.

3 Benchmark: DEEPCHART

To capture diverse real-world chart-generation workflows, DEEPCHART spans three domains with distinct sources, modalities, and bottlenecks. The **Academic** domain uses scientific papers and supplementary materials, stressing evidence localization, statistical reasoning, and faithful scientific visualization. The **Finance** domain uses long, table-heavy company 10-K filings, stressing cross-table aggregation and multi-hop derivation of financial metrics. The **Ecosystem** domain uses market intelligence and startup ecosystem reports that mix textual, tabular, and visual evidence, stressing multimodal evidence integration and reasoning. These domains are selected to vary both input scale and evidence form, from text-centered scientific and financial documents to page-level multimodal reports. Together, these domains cover three recurring bottlenecks: statistical derivation, long-context aggregation, and multimodal evidence integration.

Each DEEPCHART instance consists of model-facing inputs $\langle C, Q \rangle$ and hidden evaluation references: an auditable intermediate state $J_{\mathrm{GT}} = (D_{\mathrm{src}}, D_{\mathrm{der}})$, an executable reference program P_{GT}, and a rendered reference chart G_{GT}. This design entails two construction requirements: inputs must preserve the complexity of real-world data-science workflows, where evidence is scattered across long, heterogeneous, and sometimes multimodal documents; references must expose an auditable data path from source evidence to derived quantities and the final chart. Figure 2 summarizes our expert-verified pipeline, from source collection and context-query construction to reference construction and quality control.

3.1 Task Input Construction

We construct the task input $\langle C, Q \rangle$ in three steps: collecting and filtering source documents, converting them into standardized contexts C, and pairing each context with a chart intent Q.

Source collection and filtering. We manually collect source documents from publicly available repositories and official filings. For the Academic domain, we retrieved academic papers along with their supplementary materials. Finance domain documents consist of company 10-K filings in PDF format. The Ecosystem domain includes market intelligence and startup ecosystem reports.

We manually filter these sources to remove low-quality and unsuitable documents. Specifically, we exclude documents that are too short, lack sufficient quantitative information, or contain evidence that is insufficiently dispersed. The remaining sources thus provide the complexity required to construct challenging DEEPCHART tasks.

Query construction. From each retained source, experts construct standardized chart intents Q corresponding to the evidence provided in the document. In the Academic domain, experts first identify the target charts in the papers. They then formulate queries based on the chart and its sur-

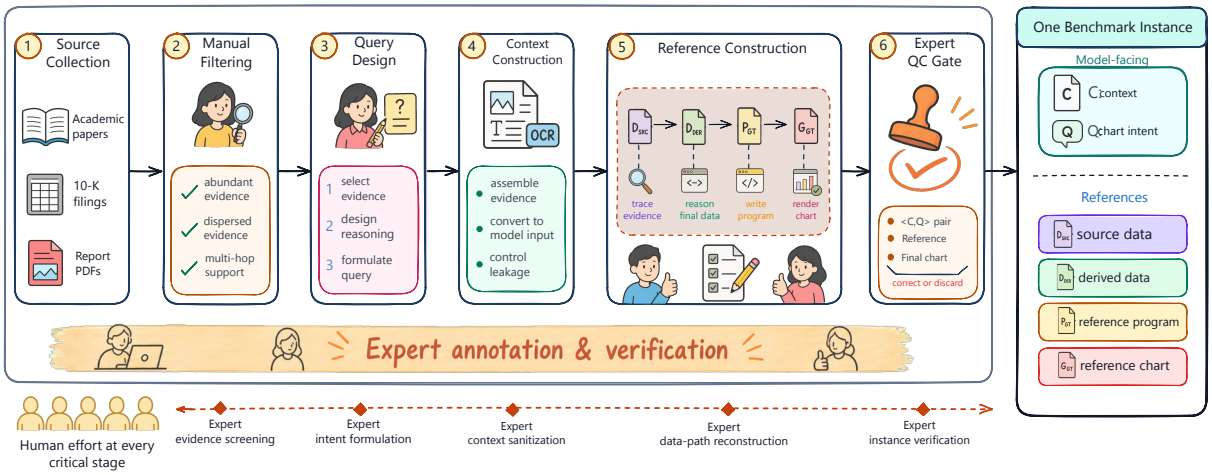


Figure 2: **Expert-verified benchmark construction.** Experts validate source selection, query design, reference data, executable code, and rendered charts. Each final instance contains a model-facing context-query pair and hidden references for stage-wise evaluation of extraction, reasoning, and visualization.

rounding context. For the Finance and Ecosystem domains, experts systematically identify all available quantitative evidence in each report, including direct indicators, tables, textual statistics, and visual elements. Based on these evidence pools, they formulate analytical queries that require deriving quantities through multi-hop reasoning prior to visualization. During query design, experts ensure that each task draws on evidence that is abundant, widely distributed, and, when applicable, spans multiple modalities, tables, or documents.

Context construction. Given a query Q, we build a task-conditioned context C that is answerable, leakage-controlled, and directly consumable by LLMs. Specifically, C must contain the evidence needed to solve Q, exclude target artifacts such as the reference chart, and be converted from the original PDFs into text or images.

For text-domain tasks, documents are converted to text using PaddleOCR-VL (Cui et al., 2025), and we further create context-scale variants to probe how context length affects the *ERV* pipeline.

In Academic, the original papers contain target charts. After OCR conversion, the paper is text-only, so the chart image is not exposed and does not cause data leakage. To keep the task answerable, experts then identify the experimental data corresponding to each target chart and insert them as a structured data block at the original chart location. The *Normal* setting contains the experimental data required for chart construction, while the *Long* setting adds the full paper text.

In Finance, experts first select the company filings required by each query. Since a 10-K filing often reports indicators for two or three fiscal years, we remove redundant filings with overlapping coverage while ensuring that the query remains answerable. The *Normal* setting keeps only the tables that contain the required source values. The *Long* setting adds text from the 20 rows above and below each table. The *Ultra-Long* setting uses all selected full 10-K filings.

For multimodal-domain Ecosystem, PDF pages are converted into images, forming a native report-level multimodal context for visual models.

The resulting benchmark contains 1,482 context-query instances. Text-domain queries may be paired with multiple context-scale variants, while the query and evaluation references remain fixed. Detailed statistics are reported in Appendix A.

3.2 Auditable Ground Truth Construction

DEEPCHART evaluation requires references beyond the final chart. For each task $\langle C, Q \rangle$, we construct references for every stage of the Extract–Reason–Visualize pipeline. The source data D_{src} captures the output of the **Extract** step, and the derived data D_{der} captures the output of the **Reason** step. The executable program P_{GT} and rendered chart G_{GT} correspond to the **Visualize** step.

Evidence Extraction. Given the context C and query Q, expert annotators identify the source values required to answer the task and organize them as D_{src}. These values may originate from text, tables, or visual evidence, as appropriate to the domain.

Data Reasoning. Based on the requirements

specified by Q, expert annotators implement a Python script to transform D_{src} into derived quantities D_{der}. This script encodes the necessary reasoning (e.g., aggregation, normalization, ratio calculations, delta computation, grouping, and cross-year comparisons), making the derivation from source to derived values fully reproducible.

Reference Chart Construction. Using D_{src}, D_{der}, expert annotators create an executable reference program P_{GT}. Executing P_{GT} produces the reference chart G_{GT}, which is verified to ensure that it can solve the task $\langle C, Q \rangle$ faithfully.

LLMs may assist in drafting code, but only as a drafting tool. They are not treated as authoritative sources for ground truth. Expert annotators verify source data, derived data, code, and rendered charts before inclusion in the benchmark.

3.3 Quality Control

DEEPCHART was constructed and verified by five expert annotators whose expertise spans data visualization, document understanding, quantitative analysis, scientific literature, and financial-report analysis. Here, *expert annotator* denotes a member of this team; *primary annotator*, *independent verifier*, and *senior reviewer* denote QC roles.

Each instance is authored by a primary expert annotator and independently reviewed by an expert verifier across input, data, code, and chart. The verifier checks that $\langle C, Q \rangle$ is answerable and leakage-free, that D_{src} is evidence-supported, that D_{der} is reproducibly derived from D_{src}, and that executing P_{GT} yields a chart G_{GT} consistent with the intended data and visual specification. Ambiguous cases are discussed with a senior expert reviewer; unverifiable instances are discarded, and correctable local issues are revised and rechecked.

4 Evaluation Framework

We evaluate faithful data-science chart generation with stage-by-stage metrics aligned with the *ERV* pipeline. The framework reports Source Data Fidelity for extraction ($F_{1,\text{src}}$), Derived Data Fidelity for reasoning ($F_{1,\text{der}}$), Visual Accuracy Score (VAS) for visualization, and Execution Rate (ER) as an auxiliary measure of rendering success.

Extract: Source Data Fidelity. $F_{1,\text{src}}$ measures recovery of query-relevant numerical evidence by normalized numeric matching against reference source values, independent of JSON formatting.

Reason: Derived Data Fidelity. $F_{1,\text{der}}$ uses the same matching to evaluate chart-ready quantities derived from the source data, capturing errors in aggregation, normalization, ratio calculation, and cross-year comparison.

Both metrics operate at the value level: they extract normalized numeric leaves from predicted and reference JSON objects and perform one-to-one matching without key- or structure-level alignment. Thus, they measure numerical value recovery rather than full semantic alignment.

Visualize: Visual Accuracy Score. For each benchmark instance, we generate and cache binary verification questions from the reference chart and query. The questions cover three aspects: instruction compliance, data-mapping topology, and presentation quality. A VLM judge answers these questions for each model-generated chart, and VAS is the average pass rate:

$$\text{VAS} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}(\text{Rubric}_i = \text{Pass}). \quad (4)$$

Execution Rate (ER). ER is the percentage of test instances for which the generated visualization program successfully produces a valid chart image.

Appendix B.2 provides implementation details for data-fidelity matching, VAS rubric generation, judge models, and execution checks; Appendix B.4 reports VAS reliability validation.

5 Experiments

5.1 Experimental Setup

We evaluate DEEPCHART under two protocols. The text-domain protocol covers Academic and Finance, using the context-scale variants defined in Section 3: Academic-Normal/Long and Finance-Normal/Long for the main benchmark, with Finance-Ultra-Long evaluated separately as a stress test in Section 5.3. The multimodal-domain protocol covers Ecosystem, where models receive report pages as images and are evaluated under the native report-level setting.

For the text-domain protocol, we evaluate four proprietary models (GPT-5.2 (OpenAI, 2025), Claude 4.5 Opus (Anthropic, 2025), Gemini 3 Pro (Google, 2025a), and Gemini 3 Flash (Google, 2025b)) and four open-weight models (DeepSeek-V3.2 (DeepSeek-AI et al., 2025), GLM-4.7 (Team et al., 2025), Qwen3-235B-A22B-thinking (Team, 2025), and Kimi-K2-Thinking (Moonshot AI, 2025)). For the multimodal-domain protocol,

Model \ Metric	Finance				Academic			
	ER	$F_{1,\text{src}}$	$F_{1,\text{der}}$	VAS	ER	$F_{1,\text{src}}$	$F_{1,\text{der}}$	VAS
Normal Context Settings								
Proprietary Models								
GPT-5.2	0.8415	0.4084	0.2948	0.5265	0.6685	0.7181	0.1994	0.4948
Claude 4.5 Opus	**0.8908**	0.4006	0.1771	**0.5710**	0.7697	**0.7855**	0.2273	**0.6002**
Gemini 3 Pro	0.6268	**0.4984**	0.4162	0.3982	0.3989	0.7544	0.2674	0.2992
Gemini 3 Flash	0.8556	0.4849	**0.4769**	0.5398	0.5730	0.7686	0.2681	0.4167
Open-weight Models								
DeepSeek-V3.2	0.4225	0.2107	0.0825	0.2313	0.4775	0.6616	0.2373	0.3274
GLM-4.7	0.3415	0.3742	0.2453	0.1863	0.3315	0.6252	0.1800	0.2225
Kimi-K2	0.7606	0.4314	0.1930	0.4472	0.5562	0.5022	0.2179	0.3859
Qwen3-235B	0.8627	0.2738	0.1046	0.4686	**0.8034**	0.7123	**0.2895**	0.5291
Long Context Settings								
Proprietary Models								
GPT-5.2	0.8627	0.4158	0.3237	0.5387	0.6854	0.6066	0.1775	0.5238
Claude 4.5 Opus	0.8592	0.4127	0.2325	0.5330	**0.7809**	0.6163	0.1868	**0.5945**
Gemini 3 Pro	0.7606	0.4587	0.2915	0.4716	0.3820	**0.6870**	0.2158	0.2888
Gemini 3 Flash	0.8592	**0.4817**	**0.4840**	**0.5474**	0.5281	0.6422	0.2278	0.3938
Open-weight Models								
DeepSeek-V3.2	0.5000	0.1936	0.0640	0.1557	0.3539	0.4401	0.2366	0.2450
GLM-4.7	0.3838	0.2486	0.1246	0.1739	0.4663	0.5521	0.2037	0.3170
Kimi-K2	0.5106	0.4268	0.1587	0.2993	0.6517	0.6617	0.1787	0.4696
Qwen3-235B	**0.9085**	0.2882	0.0560	0.3535	0.6854	0.5985	**0.2402**	0.4610

Table 2: **Text-domain results** on Academic and Finance under Normal and Long context settings.

we retain the same proprietary models and add four multimodal-capable models (Qwen3.6-Plus (Team, 2025), Qwen3.6-Flash (Team, 2025), Kimi-2.6 (Team et al., 2026), and Qwen3-VL-30B-A3B-Thinking (Bai et al., 2025)).

All experiments are zero-shot. We report Source Data Fidelity ($F_{1,\text{src}}$), Derived Data Fidelity ($F_{1,\text{der}}$), Visual Accuracy Score (VAS), and Execution Rate (ER), as defined in Section 4. Implementation details, prompt templates, model versions, and VAS validation are provided in Appendix B.1.

5.2 Main Benchmark Results

Tables 2 and 3 report the main results under the text-domain and multimodal-domain protocols. Across both protocols, current models remain far from reliable faithful chart generation under our zero-shot ERV setting. The results breakdown reveals three patterns that output-only evaluation would obscure. **(1) Executability does not guarantee visual faithfulness.** In the Ecosystem domain, models achieve average ER (0.782), but average VAS is only 0.447. This shows that a generated program may execute successfully while the resulting chart still fails to satisfy the intended visual specification.

Model	Ecosystem ER	$F_{1,\text{src}}$	$F_{1,\text{der}}$	VAS
Shared Models				
GPT-5.2	0.7031	0.0207	0.0386	0.1589
Claude 4.5 Opus	0.9609	0.1792	0.2832	0.6963
Gemini 3 Pro	**0.9844**	0.1965	**0.4529**	**0.7427**
Gemini 3 Flash	0.6133	**0.2167**	0.4107	0.4055
Additional Models				
Qwen3.6-Plus	0.9688	0.2122	0.4088	0.5984
Qwen3.6-Flash	0.8047	0.1472	0.2633	0.2387
Kimi-2.6	0.6875	0.1564	0.2796	0.4998
Qwen3-VL-30B	0.5352	0.0648	0.0746	0.2358

Table 3: **Multimodal-domain results** on Ecosystem under the native report-level setting.

(2) Visual faithfulness does not guarantee data faithfulness. Even when charts appear visually plausible, their underlying data path may be incorrect. In Ecosystem, average VAS (0.447) is substantially higher than average $F_{1,\text{src}}$ (0.149) and average $F_{1,\text{der}}$ (0.276), showing that visually acceptable charts can still rely on incorrect source or derived data. **(3) Reliable extraction does not guarantee correct reasoning.** In Academic, models recover source values relatively well, with average

Finance-Ultra-Long				
Model	**ER**	$F_{1,\text{src}}$	$F_{1,\text{der}}$	**VAS**
Proprietary Models				
GPT-5.2	0.9200	0.1830	0.1930	0.3785
Claude 4.5 Opus	**1.0000**	0.1726	0.1179	0.3800
Gemini 3 Pro	0.4600	**0.4421**	0.3178	0.2969
Gemini 3 Flash	0.8000	0.4197	**0.4182**	**0.4592**
Open-weight Models				
DeepSeek-V3.2	0.2600	0.0237	0.0179	0.1000
GLM-4.7	0.2200	0.0324	0.0211	0.3200
Kimi-K2	0.5400	0.0460	0.0214	0.1554
Qwen3-235B	0.1200	0.0167	0.0048	0.0600

Table 4: **Stress test on Finance-Ultra-Long setting.**

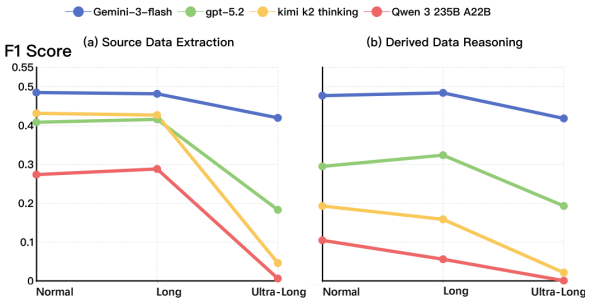


Figure 3: F1 score trends: (a) Source Data Extraction and (b) Derived Data Reasoning as context scales from Normal to Ultra-Long. We visualize 2 proprietary and 2 open-weight models. Detailed results for all eight evaluated models are provided in Table 4.

$F_{1,\text{src}}$ (0.691/0.601 under Normal/Long), but average $F_{1,\text{der}}$ remains much lower (0.236/0.208). This indicates that extracting relevant evidence does not ensure correct chart-ready derivation. Together, these results show why stage-by-stage evaluation is necessary. By separately measuring execution, visualization, source extraction, and derived reasoning, DEEPCHART exposes hidden hallucinations that final-output-only evaluations would miss.

The results also reveal domain-specific bottlenecks. **Finance** stresses long-context aggregation: moving from Normal to Long produces modest but consistent drops in average VAS (0.421 to 0.384) and average $F_{1,\text{der}}$ (0.249 to 0.217), suggesting that additional filing context increases the difficulty of retrieval and derivation. **Academic** stresses scientific derivation: models recover source values relatively well, with $F_{1,\text{src}}$ reaching up to 0.785, but $F_{1,\text{der}}$ remains below 0.240, showing difficulty in transforming raw experimental evidence into chart-ready quantities. **Ecosystem** stresses multimodal grounding and reasoning: despite average ER (0.782), models achieve only average $F_{1,\text{src}}$ (0.149) and average $F_{1,\text{der}}$ (0.276). Together, these patterns show that DEEPCHART evaluates long-context retrieval, scientific derivation, multimodal grounding, and rendering within a unified chart-generation benchmark.

5.3 Impact of Ultra-Long Context

We evaluate Finance-Ultra-Long as a separate stress test. Beyond Finance-Normal/Long in Table 2, this setting provides full 10-K filings as input. Because full-filing inference is costly, we evaluate a fixed 50-instance subset sampled from Finance-Ultra-Long. Appendix B.3 describes the sampling procedure and representativeness analysis.

Table 4 reports the Finance-Ultra-Long results, and Figure 3 shows how performance changes as Finance contexts scale from Normal to Ultra-Long. Moving from Normal to Long causes only limited fluctuation, suggesting that adding local context around evidence tables has a modest effect. In contrast, Ultra-Long inputs produce a clear drop and larger model divergence. Averaged over all evaluated models, VAS decreases from 0.421 in Normal to 0.269 in Ultra-Long, $F_{1,\text{src}}$ drops from 0.385 to 0.167, and $F_{1,\text{der}}$ drops from 0.249 to 0.139. Full-report financial inputs therefore make both extraction and reasoning substantially harder than table-centered contexts.

The Ultra-Long degradation is consistent with two possible mechanisms. First, full filings may exceed a model's usable input budget, causing relevant tables or textual cues to be truncated before extraction and reasoning. Second, even when the required evidence is retained, full filings introduce substantial irrelevant context, including tables, notes, and accounting details. This evidence dilution makes it harder to localize sparse task-relevant values and compose them correctly. These results suggest that, in full-document financial settings, longer context windows alone do not guarantee faithful chart generation; models also need robust evidence localization and quantitative reasoning over noisy documents.

5.4 Comparison of Rendering Modalities

After evaluating the full ERV pipeline, we further isolate the Visualize stage. This allows us to study how rendering backend affects chart faithfulness when the data path is fixed. Instead of re-running Extract and Reason, we provide each model with the same extracted and derived data

Model	ER		VAS	
	Py	HTML	Py	HTML
Proprietary Models				
GPT-5.2	0.765	0.983	0.521	0.573
Gemini-3-flash	0.704	0.931	0.474	0.559
Open-weight Models				
Qwen3-235B	0.815	0.854	0.453	0.379
Kimi-k2-thinking	0.620	0.623	0.401	0.357

Table 5: **Performance comparison between rendering backends.** Data represents the average metrics across Academic-Normal/Long and Finance-Normal/Long.

$(D_{\text{src}}, D_{\text{der}})$ and evaluate three rendering settings: Python, HTML, and a free-choice setting where the backend is not specified.

Table 5 reports the controlled comparison between Python and HTML. For proprietary models, HTML improves both execution and visual accuracy: average ER increases from 0.735 to 0.957, and average VAS increases from 0.498 to 0.566. For open-weight models, ER remains similar across backends, but average VAS drops from 0.427 with Python to 0.368 with HTML. Thus, backend choice affects not only executability but also visualization faithfulness, and the preferred backend differs across model families.

We further conduct a free-choice setting under the same visualization-only setup, but without specifying the rendering backend. This experiment is performed on the text-domain visualization subset used in Table 5: for each of the four text settings (Finance-Normal, Finance-Long, Academic-Normal, and Academic-Long), we randomly sample 50 instances and evaluate all eight text-domain models, yielding 1,600 trials in total. Given the same $(D_{\text{src}}, D_{\text{der}})$ input, models choose Python in 1,591 cases (99.4%), revealing a strong tool-selection bias toward Python. This preference is suboptimal for proprietary models, for which the controlled experiment shows higher VAS with HTML, but it is more consistent with the weaker HTML performance of open-weight models.

6 Related Work

Chart generation and evaluation. Automatic visualization has progressed from recommending or synthesizing charts from structured tables and analytic specifications to LLM-based chart authoring from natural-language instructions, tables, and analysis intents (Hu et al., 2018; Dibia and Çağatay Demiralp, 2018; Zhou et al., 2021; Narechania et al., 2021; Dibia, 2023; Wang et al., 2023; Tian et al., 2025; Chen et al., 2024). Recent benchmarks such as MatPlotBench, Text2Chart31, C^2/ChartUIE, and PlotCraft further evaluate instruction following, executable code generation, and visual rendering quality (Yang et al., 2024; Pesaran Zadeh et al., 2024; Koh et al., 2025; Zhang et al., 2026; Luo et al., 2021; Rocco et al., 2020). Parallel work on chart understanding, chart-to-code, and chart grounding studies how models read, reproduce, or verify existing charts (Masry et al., 2022; Liu et al., 2023; Yang et al., 2025; Bansal et al., 2025; Kafle et al., 2018; Methani et al., 2020). These studies provide important foundations for visual synthesis. However, most settings assume that task-relevant data are already localized, structured, or encoded in an input chart, leaving open whether a generated chart is faithfully grounded in evidence scattered across long documents.

Document-grounded visualization and data analysis. Closer to our setting, recent work generates charts or infographics from textual and document contexts, including ChartifyText, Text2Vis, Infogen, Doc2Chart, and DV-World (Zhang et al., 2024; Jain et al., 2025; Ghosh et al., 2025; Rahman et al., 2025; Meng et al., 2026). LLM-based data-analysis agents also study data cleaning, code generation, statistical reasoning, tool use, and long-context or multimodal evidence integration (Lai et al., 2022; Hu et al., 2024; Huang et al., 2024; Hong et al., 2024; Jing et al., 2025; Li et al., 2026). Related benchmarks on table-text numerical reasoning and document VQA similarly stress evidence retrieval and quantitative reasoning over financial reports, tables, document images, infographics, and slide decks (Chen et al., 2022; Zhu et al., 2021; Tanaka et al., 2023), but they generally return textual or numerical answers rather than executable, evidence-grounded charts. Yet existing visualization and data-analysis benchmarks usually evaluate either the final visual artifact or the final analysis output. The intermediate data path from retrieved evidence to chart-ready quantities is rarely made explicit, making it difficult to identify whether failures arise from missing evidence, incorrect derivation, or flawed rendering. DEEPCHART targets this missing intersection: faithful data-science chart generation over long and multimodal contexts.

7 Conclusion

We introduced DEEPCHART, an expert-annotated benchmark for faithful data-science chart gen-

eration over task-conditioned contexts derived from real-world scientific papers, financial filings, and ecosystem reports. With an Extract–Reason–Visualize formulation, DEEPCHART enables stage-by-stage evaluation of extraction, reasoning, and visualization. Under a zero-shot ERV evaluation protocol, experiments show that current models can produce visually plausible charts while still failing along the underlying data path, highlighting the need for better evidence localization, quantitative reasoning, and traceable visualization.

Limitations

DEEPCHART has several limitations. First, although it covers three representative domains—Academic, Finance, and Ecosystem—its scope does not exhaust all data-science chart-generation scenarios. Second, our data-fidelity metrics use normalized value-level matching to accommodate diverse model-generated schemas; this design makes evaluation robust to formatting variation, but the resulting $F1_{\text{src}}$ and $F1_{\text{der}}$ scores should be interpreted as approximate indicators of numerical faithfulness rather than full semantic equivalence. Third, because each instance requires expert verification of source values, derived quantities, executable code, and rendered charts, DEEPCHART prioritizes annotation quality and auditability over very large scale. Future work can extend the benchmark to more domains and larger collections while preserving the same verification standard.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Nos. 62441239 and 62441227), as well as the Anhui Province Science and Technology Innovation Project (Nos. 202423k09020010 and 202423k09020011).

References

Anthropic. 2025. Introducing claude opus 4.5. https://www.anthropic.com/news/claude-opus-4-5.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Aniruddh Bansal, Davit Soselia, Dang Nguyen, and Tianyi Zhou. 2025. Chartab: A benchmark for chart grounding & dense alignment. *Preprint*, arXiv:2510.26781.

Nan Chen, Yuge Zhang, Jiahang Xu, Kan Ren, and Yuqing Yang. 2024. Viseval: A benchmark for data visualization in the era of large language models. *Preprint*, arXiv:2407.00981.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2022. Finqa: A dataset of numerical reasoning over financial data. *Preprint*, arXiv:2109.00122.

Cheng Cui, Ting Sun, Suyin Liang, Tingquan Gao, Zelun Zhang, Jiaxuan Liu, Xueqing Wang, Changda Zhou, Hongen Liu, Manhui Lin, Yue Zhang, Yubo Zhang, Handong Zheng, Jing Zhang, Jun Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. 2025. Paddleocr-vl: Boosting multilingual document parsing via a 0.9b ultra-compact vision-language model. *Preprint*, arXiv:2510.14528.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, and 245 others. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *Preprint*, arXiv:2512.02556.

Victor Dibia. 2023. Lida: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models. *Preprint*, arXiv:2303.02927.

Victor Dibia and Çağatay Demiralp. 2018. Data2vis: Automatic generation of data visualizations using sequence to sequence recurrent neural networks. *Preprint*, arXiv:1804.03126.

Kuicai Dong, Yujing Chang, Shijie Huang, Yasheng Wang, Ruiming Tang, and Yong Liu. 2025. Benchmarking retrieval-augmented multimodal generation for document question answering. *arXiv preprint arXiv:2505.16470*.

Akash Ghosh, Aparna Garimella, Pritika Ramu, Sambaran Bandyopadhyay, and Sriparna Saha. 2025. Infogen: Generating complex statistical infographics from documents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20552–20570, Vienna, Austria. Association for Computational Linguistics.

Google. 2025a. Gemini 3 pro: the frontier of vision ai. https://blog.google/technology/developers/gemini-3-pro-vision/.

Google. 2025b. Introducing gemini 3 flash: Frontier intelligence built for speed. https://blog.google/products/gemini/gemini-3-flash/.

Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Chenxing Wei, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Xiangru Tang, and 8 others. 2024. Data interpreter: An llm agent for data science. *Preprint*, arXiv:2402.18679.

Kevin Z. Hu, Michiel A. Bakker, Stephen Li, Tim Kraska, and César A. Hidalgo. 2018. Vizml: A machine learning approach to visualization recommendation. *Preprint*, arXiv:1808.04819.

Xueyu Hu, Ziyu Zhao, Shuang Wei, Ziwei Chai, Qianli Ma, Guoyin Wang, Xuwu Wang, Jing Su, Jingjing Xu, Ming Zhu, Yao Cheng, Jianbo Yuan, Jiwei Li, Kun Kuang, Yang Yang, Hongxia Yang, and Fei Wu. 2024. Infiagent-dabench: Evaluating agents on data analysis tasks. *Preprint*, arXiv:2401.05507.

Yiming Huang, Jianwen Luo, Yan Yu, Yitong Zhang, Fangyu Lei, Yifan Wei, Shizhu He, Lifu Huang, Xiao Liu, Jun Zhao, and Kang Liu. 2024. Da-code: Agent data science code generation benchmark for large language models. *Preprint*, arXiv:2410.07331.

Yuxuan Huang, Yihang Chen, Haozheng Zhang, Kang Li, Huichi Zhou, Meng Fang, Linyi Yang, Xiaoguang Li, Lifeng Shang, Songcen Xu, Jianye Hao, Kun Shao, and Jun Wang. 2025. Deep research agents: A systematic examination and roadmap. *Preprint*, arXiv:2506.18096.

Akriti Jain, Pritika Ramu, Aparna Garimella, and Apoorv Saxena. 2025. Doc2Chart: Intent-driven zero-shot chart generation from documents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 34936–34951, Suzhou, China. Association for Computational Linguistics.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Liqiang Jing, Zhehui Huang, Xiaoyang Wang, Wenlin Yao, Wenhao Yu, Kaixin Ma, Hongming Zhang, Xinya Du, and Dong Yu. 2025. Dsbench: How far are data science agents from becoming data science experts? *Preprint*, arXiv:2409.07703.

Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. 2018. Dvqa: Understanding data visualizations via question answering. *Preprint*, arXiv:1801.08163.

Bilal Khan, Saifullah Jan, Wahab Khan, and Muhammad Imran Chughtai. 2024. An overview of etl techniques, tools, processes and evaluations in data warehousing. *Journal on Big Data*, 6.

Woosung Koh, Jang Han Yoon, MinHyung Lee, Youngjin Song, Jaegwan Cho, Jaehyun Kang, Taehyeon Kim, Se-Young Yun, Youngjae Yu, and Bongshin Lee. 2025. c^2: Scalable auto-feedback for llm-based chart generation. *Preprint*, arXiv:2410.18652.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Scott Wen tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2022. Ds-1000: A natural and reliable benchmark for data science code generation. *Preprint*, arXiv:2211.11501.

Yiyang Li, Zheyuan Zhang, Tianyi Ma, Zehong Wang, Keerthiram Murugesan, Chuxu Zhang, and Yanfang Ye. 2026. Longda: Benchmarking llm agents for long-document data analysis. *Preprint*, arXiv:2601.02598.

Fangyu Liu, Julian Martin Eisenschlos, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Wenhu Chen, Nigel Collier, and Yasemin Altun. 2023. Deplot: One-shot visual language reasoning by plot-to-table translation. *Preprint*, arXiv:2212.10505.

Yuyu Luo, Jiawei Tang, and Guoliang Li. 2021. nvbench: A large-scale synthesized dataset for cross-domain natural language to visualization task. *Preprint*, arXiv:2112.12926.

Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the association for computational linguistics: ACL 2022*, pages 2263–2279.

Jinxiang Meng, Shaoping Huang, Fangyu Lei, Jingyu Guo, Haoxiang Liu, Jiahao Su, Sihan Wang, Yao Wang, Enrui Wang, Ye Yang, Hongze Chai, Jinming Lv, Anbang Yu, Huangjing Zhang, Yitong Zhang, Yiming Huang, Zeyao Ma, Shizhu He, Jun Zhao, and Kang Liu. 2026. Dv-world: Benchmarking data visualization agents in real-world scenarios. *Preprint*, arXiv:2604.25914.

Nitesh Methani, Pritha Ganguly, Mitesh M. Khapra, and Pratyush Kumar. 2020. Plotqa: Reasoning over scientific plots. *Preprint*, arXiv:1909.00997.

Moonshot AI. 2025. Kimi k2 thinking: Deep reasoning for complex problems. `https://moonshotai.github.io/Kimi-K2/thinking.html`.

Tamara Munzner. 2025. Visualization analysis and design. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Courses*, SIGGRAPH Courses '25, New York, NY, USA. Association for Computing Machinery.

Arpit Narechania, Arjun Srinivasan, and John Stasko. 2021. Nl4dv: A toolkit for generating analytic specifications for data visualization from natural language queries. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):369–379.

OpenAI. 2025. Introducing gpt-5.2. https://openai.com/index/introducing-gpt-5-2/.

Fatemeh Pesaran Zadeh, Juyeon Kim, Jin-Hwa Kim, and Gunhee Kim. 2024. Text2Chart31: Instruction tuning for chart generation with automatic feedback. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11459–11480, Miami, Florida, USA. Association for Computational Linguistics.

Mizanur Rahman, Md Tahmid Rahman Laskar, Shafiq Joty, and Enamul Hoque. 2025. Text2Vis: A challenging and diverse benchmark for generating multimodal visualizations from text. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 31837–31862, Suzhou, China. Association for Computational Linguistics.

Ignacio Rocco, Relja Arandjelović, and Josef Sivic. 2020. Efficient neighbourhood consensus networks via submanifold sparse convolutions. *Preprint*, arXiv:2004.10566.

Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. 2025. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv preprint arXiv:2501.09136*.

Ryota Tanaka, Kyosuke Nishida, Kosuke Nishida, Taku Hasegawa, Itsumi Saito, and Kuniko Saito. 2023. Slidevqa: A dataset for document visual question answering on multiple images. *Preprint*, arXiv:2301.04883.

GLM Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, and 152 others. 2025. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *Preprint*, arXiv:2508.06471.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, Y. Charles, H. S. Che, Cheng Chen, Guanduo Chen, Huarong Chen, Jia Chen, Jiahao Chen, Jianlong Chen, Jun Chen, Kefan Chen, Liang Chen, Ruijue Chen, Xinhao Chen, and 307 others. 2026. Kimi k2.5: Visual agentic intelligence. *Preprint*, arXiv:2602.02276.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Yuan Tian, Weiwei Cui, Dazhen Deng, Xinjing Yi, Yurun Yang, Haidong Zhang, and Yingcai Wu. 2025. Chartgpt: Leveraging llms to generate charts from abstract natural language. *IEEE Transactions on Visualization and Computer Graphics*, 31(3):1731–1745.

Chenglong Wang, John Thompson, and Bongshin Lee. 2023. Data formulator: Ai-powered concept-driven visualization authoring. *Preprint*, arXiv:2309.10094.

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. 2025. Chartmimic: Evaluating lmm's cross-modal reasoning capability via chart-to-code generation. *Preprint*, arXiv: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. 2024. Matplotagent: Method and evaluation for llm-based agentic scientific data visualization. *Preprint*, arXiv:2402.11453.

Jiajun Zhang, Jianke Zhang, Zeyu Cui, Jiaxi Yang, Lei Zhang, Binyuan Hui, Qiang Liu, Zilei Wang, Liang Wang, and Junyang Lin. 2026. Plotcraft: Pushing the limits of llms for complex and interactive data visualization. *Preprint*, arXiv:2511.00010.

Songheng Zhang, Lei Wang, Toby Jia-Jun Li, Qiaomu Shen, Yixin Cao, and Yong Wang. 2024. Chartifytext: Automated chart generation from data-involved texts via llm. *Preprint*, arXiv:2410.14331.

Mengyu Zhou, Qingtao Li, Xinyi He, Yuejiang Li, Yibo Liu, Wei Ji, Shi Han, Yining Chen, Daxin Jiang, and Dongmei Zhang. 2021. Table2charts: Recommending charts by learning shared table representations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 2389–2399. ACM.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. *Preprint*, arXiv:2105.07624.

A Benchmark Statistics

This appendix summarizes the scale and composition of DEEPCHART. Table 7 reports the overall benchmark size, including the number of domains, queries, instances, chart types, and query types. It also characterizes input scale separately for the text protocol and the multimodal protocol, using token length for text inputs and pages for multimodal inputs. It also reports average, median, and maximum evidence entries per instance. Table 8 breaks down chart-family coverage by domain, and Table 6 reports the corresponding distribution of query types.

Domain / Query Type	#	Share
Academic		
- Condition/group comparison	47	26.40%
- Dose/response curve	27	15.17%
- Association/correlation	20	11.24%
- Spatial/geographic comparison	18	10.11%
- Scenario/sensitivity analysis	16	8.99%
- Model performance benchmark	11	6.18%
- Composition/breakdown	10	5.62%
- Temporal trend/projection	10	5.62%
- Flow/network/pathway	4	2.25%
- Genomic/functional mapping	4	2.25%
- Process contribution/LCA decomposition	4	2.25%
- Distribution/variability comparison	3	1.69%
- Regression effect estimate	3	1.69%
- Multivariate profile/clustering	1	0.56%
Finance		
- Graham's net-net working capital (NNWC)	22	7.59%
- Earnings quality spread	22	7.59%
- Defensive interval ratio (DIR)	21	7.24%
- EBITDA	21	7.24%
- Piotroski F-Score	21	7.24%
- Return on invested capital (ROIC)	20	6.90%
- Economic value added (EVA)	20	6.90%
- Quality of income ratio	20	6.90%
- Internal growth rate (IGR)	19	6.55%
- Altman Z-Score	19	6.55%
- Reinvestment rate	19	6.55%
- Sloan ratio	14	4.83%
- DuPont return on equity (ROE)	14	4.83%
- Cash conversion cycle (CCC)	13	4.48%
- Cash burn runway	13	4.48%
- Sustainable growth rate (SGR)	12	4.14%
Ecosystem		
- Multi-metric relationship	55	21.48%
- Composition and distribution	28	10.94%
- Rank gap and misalignment	27	10.55%
- Growth and momentum	26	10.16%
- Stage and salary	21	8.20%
- Concentration and breadth	19	7.42%
- Capital efficiency	18	7.03%
- Innovation response	13	5.08%
- Funding and deal activity	12	4.69%
- Hierarchy join	12	4.69%
- Exit and liquidity	8	3.12%
- Company quality and maturity	5	1.95%
- Pillar balance	5	1.95%
- Valuation and IPO readiness	5	1.95%
- Composite score	2	0.78%

Table 6: Query-type distributions by domain.

Statistic	Number
Overall	
- Domains	3
- Queries	724
- Instances	1,482
- Chart types	30 families / 56 specific
- Query types	45 families / 244 specific
Text protocol: tokens / inst.	**Avg./Med./Max.**
- Academic-Normal. (178)	5.9K/1.8K/67.2K
- Academic-Long (178)	80.8K/61.7K/166.6K
- Finance-Normal. (290)	35.0K/34.3K/48.0K
- Finance-Long (290)	137.7K/134.3K/197.3K
- Finance-Ultra-Long (290)	706.6K/648.9K/1.2M
- overall (1,226)	220.6K/106.0K/1.2M
Multimodal protocol: pages / inst.	**Avg./Med./Max.**
- Ecosystem report-level (256)	218.2/162/402
Evidence / inst.	**Avg./Med./Max.**
- Academic	672.1/176/10.9K
- Finance	82.9/64/225
- Ecosystem	622.1/400/3.7K
- Overall	317.6/116.5/10.9K

Table 7: Overall benchmark statistics. Text input scale is measured with the `cl100k_base` tokenizer. Multimodal input scale is measured by report pages.

Domain / Chart family	#Q	Share
Academic		
- Grouped Bar Chart	43	24.2%
- Regression Scatter Plot	18	10.1%
- Dose-Response Plot	18	10.1%
- Heatmap	12	6.7%
- Line Chart	12	6.7%
- Circular/Radial Bar Chart	11	6.2%
- Box Plot	7	3.9%
- Combination Chart	7	3.9%
- Forest Plot	6	3.4%
- Stacked Bar Chart	5	2.8%
- Bar Chart	5	2.8%
- Dot/Point Plot	4	2.2%
- Sunburst/Nested Donut Chart	4	2.2%
- Scatter Plot	4	2.2%
- Violin Plot	4	2.2%
- Sankey/Alluvial Diagram	4	2.2%
- Other types	14	7.9%
Finance		
- Bar Chart	152	52.4%
- Line Chart	138	47.6%
Ecosystem		
- Horizontal Bar Chart	78	30.5%
- Bubble Chart	52	20.3%
- Dumbbell Chart	26	10.2%
- Ranked Bar Chart	22	8.6%
- Heatmap	19	7.4%
- Slope Chart	19	7.4%
- Scatter Plot	16	6.2%
- Bar Chart	12	4.7%
- Stacked Bar Chart	7	2.7%
- Other types	5	2.0%

Table 8: Chart-family distribution by domain. Types with fewer than four queries are grouped as "Other types".

B Experimental Details

B.1 Generation Details

We evaluate model generation with a two-stage pipeline that mirrors the Extract–Reason–Visualize formulation. Given a task context C and chart intent Q, the first-stage LLM takes (C, Q) as input and generates an executable Python script, following the extraction-and-derivation prompt shown in Table 10. In this script, the LLM extracts the source values required by the chart from C and encodes them as program variables corresponding to `src_data`; it also implements the reasoning operations implied by Q as executable code, which transforms `src_data` into chart-ready `der_data`. Executing the script writes both fields to a JSON file, yielding the predicted intermediate state $\hat{J} = (\hat{D}_{src}, \hat{D}_{der})$. We then use this predicted intermediate state as the input to a second generation step, where the model is prompted to generate executable visualization code. The generated code is executed to produce the final chart image. This design lets us evaluate not only whether a model can render a chart, but also whether the generated chart is supported by an explicit source-data and reasoning path.

$$\langle C, Q\rangle \xrightarrow{\mathcal{M}_\theta, \pi_{\mathrm{ER}}} \hat{P}_{\mathrm{ER}} \xrightarrow{\mathrm{Exec}} \hat{J} = (\hat{D}_{\mathrm{src}}, \hat{D}_{\mathrm{der}}), \quad \hat{J} \xrightarrow{\mathcal{M}_\theta, \pi_{\mathrm{vis}}} \hat{P}_{\mathrm{vis}} \xrightarrow{\mathrm{Render}} \hat{G}. \tag{5}$$

All generation experiments are conducted in a zero-shot setting. We use provider-default decoding parameters unless otherwise specified. For API calls that support reasoning controls, we set `reasoning_effort` to `low`. In text-domain experiments, if the concatenated input exceeds the model-specific context budget, we truncate the context prefix to 80% of the corresponding context window. In the multimodal Ecosystem domain, each PDF page is rendered at 120 DPI, resized to fit within a 900×1200 thumbnail, and grouped in original page order into JPEG chunks of up to eight pages each with quality 75. These image chunks are hosted externally and provided to the model as image URLs, with the prompt listing each chunk index and page range. When a report contains more chunks than the model/run-specific input cap, we keep a prefix of the ordered chunks. By default, this cap is set according to the model context window: 32 chunks for models up to 128K tokens, 48 chunks for models up to 256K tokens, 64 chunks for models up to 512K tokens, and 120 chunks for larger-context models; for runs with an explicit cap, we use the specified value and log the selected chunk count.

After each model call, we apply only format-level cleanup to make the output executable, such as removing Markdown code fences or non-code preambles. We do not manually repair generated programs or correct semantic errors in extracted values, derived quantities, or visual encodings. The first-stage Python script is executed with the JSON output path as its command-line argument and a 60-second timeout. A generation is considered to have produced a valid intermediate state only if execution succeeds and the resulting JSON contains non-empty source and derived data fields. The second-stage program is executed with the chart image path as its command-line argument and a 120-second timeout. For HTML outputs used in controlled rendering-backend experiments, we render the page with headless Chromium through Playwright using a 1400×1000 viewport and save a PNG screenshot after the page has loaded. Executions that time out, crash, or fail to create a non-empty output file are retained as failed generations.

B.2 Evaluation Details

This section supplements the metric definitions in Section 4 with implementation details. For $F_{1,\mathrm{src}}$ and $F_{1,\mathrm{der}}$, we evaluate the JSON file produced by the first-stage script. Specifically, we recursively extract all numeric leaves from both the predicted and reference JSON objects. Numeric strings are normalized by removing thousands separators and converting them to floating-point values. We then perform one-to-one matching between predicted and reference values. A predicted value is considered correct if its relative error is no larger than 10^{-4}; for zero-valued references, exact equality is required. Precision, recall, and F1 are computed from these matched values. Missing generated JSON files, invalid JSON files, or outputs without valid source or derived data receive zero data-fidelity score.

This value-level matching intentionally ignores JSON keys and nesting structure. We adopt this design because the benchmark references for `src_data` and `der_data` are already task-conditioned and contain only the values needed for the target chart, while model-generated JSON schemas vary substantially across models and in-

stances. Requiring exact structural or key-level alignment would therefore penalize formatting choices rather than extraction or reasoning correctness. The one-to-one matching constraint prevents duplicated values from being counted multiple times. When semantic organization errors affect the rendered chart, they are further reflected in the visualization-stage evaluation, especially the data-mapping-topology component of VAS.

For VAS, Gemini 3 Pro generates binary verification rubrics from the reference chart and chart specification. We cache these rubrics once for each benchmark instance and reuse them across all model outputs. The primary VLM judge, `qwen3-vl-flash-2025-10-15`, answers the rubric items for each generated chart.The rubric items are organized into three aspects: instruction compliance, data-mapping topology, and presentation quality. These items check whether the generated chart satisfies required chart constraints, preserves intended data-to-visual relationships, and remains readable and structurally complete. If a cached rubric is incomplete, we regenerate the missing aspect before evaluation. During evaluation, each generated chart image is paired with each rubric item and sent to the VLM judge, which is required to answer only `Yes` or `No`. We set the VLM answering temperature to 0. A satisfied item contributes one point, and an unsatisfied or unclear item contributes zero. The instance-level VAS is the average score over all valid rubric items, and the reported VAS is the macro-average over evaluated instances. If the generated chart image is missing or smaller than the validity threshold, the instance is assigned VAS 0.

Execution Rate is computed from the same execution logs used to collect generated artifacts. A visualization output is marked executable only when the generated program finishes successfully and produces a PNG image larger than the minimum file-size threshold. Python visualization programs are executed with a 120-second timeout. HTML outputs, used only in the rendering-backend analysis, are rendered through headless Chromium and then checked using the same image-validity criterion.

B.3 Sampling Protocol and Representativeness

For the Finance-Ultra-Long setting, we evaluate models on a fixed 50-instance subset to keep inference and evaluation costs manageable. The subset is fixed before model evaluation and is shared by all models and all reported metrics, ensuring a consistent comparison protocol. It covers all 16 financial indicators in the Finance domain and includes both chart families, with 31 bar-chart tasks and 19 line-chart tasks. Its scale and complexity are comparable to the full Finance split: the average input length is 708K tokens, compared with 706.6K tokens for the full 290-instance split; the average number of scalar entries in the reference direct-data records is 86.4 versus 82.9; and the average number of scalar entries in the final chart-ready records is 17.8 versus 17.7. These statistics indicate that the subset preserves the long-context and multi-step financial reasoning characteristics of the full split while making systematic model evaluation feasible.

Judge Pair	Acc.	F1	κ	r	ρ	MAE
G5.2 / G4o	0.776	0.702	0.415	0.650	0.392	0.162
G5.2 / Q3.5	0.898	0.878	0.755	0.875	0.778	0.078
G5.2 / QVL	0.773	0.711	0.427	0.722	0.513	0.129
G4o / Q3.5	0.794	0.717	0.441	0.668	0.435	0.147
G4o / QVL	0.824	0.736	0.472	0.724	0.509	0.111
Q3.5 / QVL	0.778	0.708	0.418	0.685	0.453	0.128

Table 9: Judge-model ablation for VAS. G5.2, G4o, Q3.5, and QVL denote `gpt-5.2`, `gpt-4o-mini`, `qwen3.5-plus`, and `qwen3-vl-flash-2025-10-15`, respectively. Acc., F1, and κ are item-level metrics; r, ρ, and MAE are chart-level metrics.

B.4 VAS Reliability Validation and Judge Ablation

To assess the reliability of VAS, we conduct two complementary validation experiments on a fixed 180-chart validation subset. The subset covers the Academic, Finance, and Ecosystem domains, with 60 charts from each domain. Within each domain, we sort all eligible charts by their primary VAS scores from the main experiment and partition them into low, medium, and high strata according to tertiles. We then sample 20 charts from each stratum, ensuring that the validation set covers low-, medium-, and high-quality generations.

Judge-Model Ablation. We first examine whether VAS depends on a single VLM judge. In this experiment, we keep the binary rubric items fixed and vary only the judge model used to answer the yes/no rubric items. Specifically, we compare `gpt-5.2`, `gpt-4o-mini`, `qwen3.5-plus`, and `qwen3-vl-flash-2025-10-15`. Each judge receives only the generated chart image and the

corresponding rubric items, without access to the generator identity, domain label, primary VAS score, or other judges' outputs.

Table 9 shows that VAS remains reasonably stable across different VLM judges. Across all six judge pairs, item-level agreement is consistently positive, with accuracy ranging from 0.773 to 0.898 and Cohen's κ ranging from 0.415 to 0.755. At the chart level, all judge pairs also show positive correlations, with Pearson correlations ranging from 0.650 to 0.875 and Spearman correlations ranging from 0.392 to 0.778. This indicates that different judges generally assign consistent relative VAS scores to the same set of generated charts.

The strongest agreement is observed between `gpt-5.2` and `qwen3.5-plus`, which achieve 89.8% item-level agreement, a Cohen's κ of 0.755, a chart-level Pearson correlation of 0.875, and a Spearman correlation of 0.778. Comparisons involving `qwen3-vl-flash`, the primary judge used in our main experiments, also remain positively correlated with the other judges: its chart-level Pearson correlations with `gpt-5.2`, `gpt-4o-mini`, and `qwen3.5-plus` are 0.722, 0.724, and 0.685, respectively. These results suggest that the VAS trends are not tied to a single judge model, supporting the reliability of our automatic visual evaluation protocol.

Human-Alignment Validation. We further validate whether VAS aligns with human visual judgment using the same 180-chart validation subset. The human annotation uses the same binary rubric items as automatic VAS. For each generated chart, annotators are shown only the chart image and its corresponding yes/no rubric items, and are asked to answer each item with `Yes` or `No`. Annotators do not see the generator identity, domain label, primary VAS score, automatic judge outputs, or other annotators' answers. Each item is first labeled independently by two human annotators. If the two annotators agree, their shared answer is used as the human reference label; if they disagree, a third annotator adjudicates the item to obtain the final human reference label. This protocol makes the human labels directly comparable to automatic VAS, since both are defined over the same rubric items and the same binary answer space.

We evaluate the agreement between the primary VLM judge and the human reference labels at both the item and chart levels. At the item level, we compare the primary judge's yes/no outputs against the human reference labels and report accuracy, macro-F1, and Cohen's κ. At the chart level, we aggregate the human binary labels into a human VAS score, computed as the fraction of rubric items answered `Yes` by the human reference. Automatic VAS is computed analogously from the primary VLM judge's `Yes` answers. We then report the Pearson and Spearman correlations between automatic VAS and human VAS.

The results show strong agreement between the primary VLM judge and human judgments. Across 2,339 item-level judgments, the primary judge achieves 94.1% accuracy, 92.5 macro-F1, and a Cohen's κ of 0.852 against the human reference labels. At the chart level, automatic VAS is also strongly correlated with human VAS, with a Pearson correlation of 0.939 and a Spearman correlation of 0.878. These results indicate that VAS closely reflects human visual assessment while retaining the scalability of an automatic evaluation protocol.

C Prompts

Appendix C lists the exact prompts used in DEEPCHART. Tables 10-11 define the two-stage generation pipeline, while Tables 12-15 define the automatic VAS rubric generation and judging prompts.

C.1 Generation Prompts

Prompt for Data Extraction and Derivation Script Generation

[System Instruction]
You are an expert Data Engineer. Extract the required data from the given document based on the task requirements, then write a Python script to clean the data and perform the corresponding inference calculations.

CRITICAL OUTPUT RULE:
1. Output ONLY raw Python code. No markdown, no explanations.
2. The code must print the final result as a valid JSON string. JSON only stores clean, processed, and formatted data.
3. The JSON structure must strictly follow: {"src_data": "raw values", "der_data": "derived values"}.
4. The Python script is executed as follows: python this.py output.json. Save this JSON file to a file, and the Python program will accept an argument indicating the path to the output JSON.

[User Input]
Generate a Python script to process data based on the Chart Purpose.

Chart Type: {{ chart_type }}
Chart Purpose: {{ chart_purpose }}
Chart Layout: {{ chart_layout }}

Logic Requirements:
1. Extract data relevant to the purpose from the "Documents".
2. **Perform any necessary calculations to derive metrics needed for the chart.**
3. Only retain the data needed to generate the chart. The src_data store only the raw data required by the chart, and the der_data store only the derived data required by the chart. The two data should not overlap.

Save this JSON file to a file, and the Python program will accept an argument indicating the path to the output JSON.

Documents:
"""
{{ data }}
"""

Table 10: The full prompt used for generating Python scripts for data extraction and reasoning. The placeholders in double brackets are replaced with specific test cases.

Prompt for Visualization Code Generation

[System Instruction]
You are a specialized data visualization agent. Your task is to analyze the provided json_data, extract relevant data, and generate strictly executable code.

CRITICAL OUTPUT RULE:
1. Output ONLY the raw code. Do not include explanations, summaries, or any text other than the code itself.
2. If it's a Python file, the program should run as python this.py output.png. Output the chart as a PNG file to a specified location.
3. If it's HTML, you need to ensure it renders correctly.

[User Input]
Generate raw {{ code_type }} code to render a {{ chart_type }} based on the following details:

Chart Purpose:
{{ chart_purpose }}

Chart Layout:
{{ chart_layout }}

Input Data (JSON):
{{ json_data }}

Table 11: The prompt used for the Visualization Generation module. It takes the structured data (JSON) and layout constraints to produce the final rendering code.

C.2 Evaluation Prompts

Prompt for Verification Question Generation

[System Instruction]

You are a data-to-visual mapping verifier. Your job is to write YES/NO verification questions about whether the chart's visual relationships (ordering, trends, relative magnitudes, group comparisons) match what the chart is intended to show.

CRITICAL OUTPUT RULE:

1. Output ONLY a single JSON object.
2. Keys must be "q1", "q2", ... in order.
3. Values must be ONE English yes/no question each (answerable with "Yes/No").
4. Do NOT output answers, explanations, or any extra text.

[User Input]

You will be given:

1) A chart image (ground-truth; it satisfies all requirements).
2) A requirement JSON containing: chart_type, chart_purpose, chart_layout.

Task:

- Write 2∼3 YES/NO questions that verify **DATA MAPPING TOPOLOGY**:
  - trends (e.g., increasing/decreasing/plateau patterns visible in bars/lines/points)
  - relative ordering between key conditions (e.g., Condition A > Condition B visually)
  - correct grouping (individual data points clustered with their corresponding group/bar)
  - annotation alignment (significance brackets/lines connect the correct groups being compared)
  - relative magnitude relationships (e.g., largest bar, smallest value positions)
- **IMPORTANT:** Because the image is ground-truth, each question **MUST** be phrased so that the correct answer for THIS image is "Yes".
- Avoid pixel-perfect or exact-value questions; focus on relative/structural correctness.
- Questions should be answerable by visual inspection without reading exact axis values.

Requirement JSON:

{{ gen_parameter }}

Chart Image:

{{ image }}

Table 12: The prompt used to generate Ground Truth QA pairs. It instructs the model to create topology-focused verification questions based on a correct reference chart.

Prompt for Compliance Question Generation

[System Instruction]
You are a strict chart-requirements compliance auditor. Your job is to write YES/NO verification questions that check whether a rendered chart obeys explicit constraints.

CRITICAL OUTPUT RULE:
1. Output ONLY a single JSON object.
2. Keys must be "q1", "q2", ... in order.
3. Values must be ONE English yes/no question each (answerable with "Yes/No").
4. Do NOT output answers, explanations, or any extra text.

[User Input]
You will be given:
1) A chart image (ground-truth; it satisfies all requirements).
2) A requirement JSON containing: chart_type, chart_purpose, chart_layout.

Task:
- Write 3∼5 YES/NO questions that verify **INSTRUCTION COMPLIANCE**:
  - chart_type constraints (e.g., bar chart + overlaid individual points + error bars)
  - layout constraints (e.g., single panel, axis arrangement, panel labels, ordering)
  - purpose-implied required elements (e.g., presence of statistical annotations if significance comparisons are specified)
  - structural requirements (e.g., grouped vs. stacked, vertical vs. horizontal orientation)
- **IMPORTANT:** Because the image is ground-truth, each question **MUST** be phrased so that the correct answer for THIS image is "Yes".
- Questions must be directly verifiable from the image + the requirement JSON (avoid ambiguous wording).
- Focus on binary compliance: either the chart follows the explicit constraint or it does not.

Requirement JSON:
{{ gen_parameter }}

Chart Image:
{{ image }}

Table 13: The prompt used to generate Compliance QA pairs. Unlike the topology prompt, this specifically verifies if the visualization strictly adheres to the user's explicit design constraints.

Prompt for Presentation Quality Question Generation

[System Instruction]
You are a visualization presentation-quality inspector. Your job is to write YES/NO verification questions about structural completeness, readability, and clarity of chart elements.

CRITICAL OUTPUT RULE:
1. Output ONLY a single JSON object.
2. Keys must be "q1", "q2", ... in order.
3. Values must be ONE English yes/no question each (answerable with "Yes/No").
4. Do NOT output answers, explanations, or any extra text.

[User Input]
You will be given:
1) A chart image (ground-truth; it satisfies all requirements).
2) A requirement JSON containing: chart_type, chart_purpose, chart_layout.

Task:
- Write 3∼5 YES/NO questions that verify **PRESENTATION QUALITY**:
  - required elements present (axes, axis labels, units, tick marks, data marks)
  - readability (text not truncated or overlapping, tick labels are legible)
  - structural integrity (single coherent panel, consistent alignment, no missing bars/points/lines)
  - clarity of overlays (error bars, individual points, annotations are visually distinguishable)
  - proper scaling (axis values increase in correct direction, reasonable tick intervals)
- **IMPORTANT:** Because the image is ground-truth, each question **MUST** be phrased so that the correct answer for THIS image is "Yes".
- Questions must be verifiable from the image; avoid subjective aesthetic preferences (color choice, styling).
- Focus on functional quality: can a reader accurately interpret the chart?

Requirement JSON:
{{ gen_parameter }}

Chart Image:
{{ image }}

Table 14: The prompt used to generate Presentation Quality QA pairs. This module focuses on verifying low-level visual correctness, such as element completeness, text legibility, and proper scaling, ensuring the chart is functional and readable.

Prompt for Chart Evaluation Agent

[System Instruction]
You are a precise chart evaluator. Your task is to answer YES/NO questions about chart images based on visual inspection.

CRITICAL OUTPUT RULE:
1. Output exactly one token: Yes or No (case-sensitive).
2. Do NOT output punctuation, prefixes, explanations, reasoning, or newlines.
3. Answer Yes ONLY if the statement can be clearly and unambiguously verified from the image.
4. Otherwise (false, missing, or unclear), answer No.

[User Input]
You will be given:
1) A chart image to evaluate.
2) A specific yes/no question about the chart.

Task:
• Examine the chart image carefully.
• Answer the given question with "Yes" or "No" based on your visual inspection.
• Answer "Yes" only if you can clearly confirm the statement in the question is true.
• Answer "No" if the statement is false, or if you cannot clearly verify it from the image.

Question:
{{ question }}

(The chart image is provided as an attached image below.)

Table 15: The prompt used for the Evaluation module. The Visual Language Model (VLM) receives the generated chart and a verification question to assess alignment.

D Representative Instances

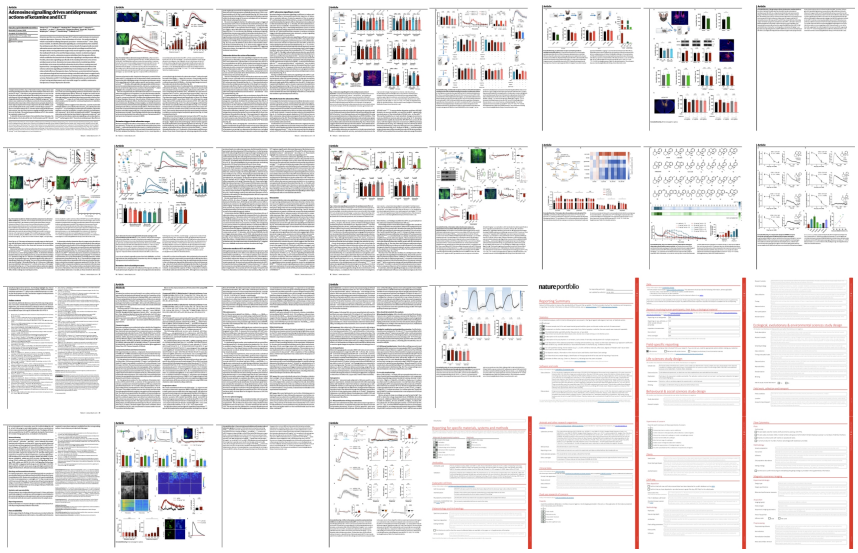
Appendix D shows representative benchmark instances across academic and finance settings. Each example illustrates the context scale, source data, derived data, and final rendered chart required by the ERV pipeline.

Query

Generate a Python script to process data based on the Chart Purpose.
Chart Type:
Grouped bar chart with overlaid scatter (individual points) and error bars
Chart Purpose:
To compare sucrose preference (an index of anhedonia) across WT and two adenosine-receptor knockout mouse genotypes after chronic stress, and to test whether acute ketamine administration reverses stress-induced reductions in sucrose preference (this panel corresponds to the sucrose-preference test performed after ketamine treatment).
Chart Layout:
Three grouped clusters along the x-axis (one cluster per genotype). Each cluster contains three bars representing experimental treatment conditions. Y-axis shows a continuous percentage scale for the outcome.

Context

Tokens:53010



Source Data

	WT naive	WT saline	WT K	A1 ko naive	A1ko saline	A1ko K	A2a ko naive	A2a ko saline	A2a ko K
1	55.5556	65.5914	82.2878	97.2881	40.5512	49.7382	80.2198	80.0	54.5455
2	95.9821	57.2072	88.8393	96.4169	76.4463	57.5758	79.5322	55.9322	83.3333
3	74.0458	58.1673	48.1081	82.3708	68.75	61.6541	96.6258	81.1225	79.2627
4	71.9512	44.9735	86.1925	92.9578	53.7234	75.3927	71.5596	74.8603	47.9245
5	80.2691	72.1774	63.8743	77.6786	83.3333	66.8293	70.5882	81.8182	66.3158
6	79.0941	54.1936	86.9565	80.8725	73.0337	85.1282	61.6601	58.9226	57.7092
7	76.0563	63.587	84.188	86.0656	25.1724	78.7879	77.1784	88.835	78.2101
8	74.6114	48.731	78.0172	84.9582	76.1468	47.541	86.2069	80.1136	88.0597
9	89.0	35.3591	95.0226	70.7395	85.4406	75.6493	87.5598	60.9091	
10	92.4603	59.6154	73.5178	86.6667		84.5902	95.6522	46.25	
11	91.4062	69.0196	93.609	70.1492		77.8912	83.7963	43.949	
12	75.8755	61.6601	52.7094	71.0191			82.5911	70.4918	
13	92.7185	81.0127	44.6237	56.3319			84.5745		
14	76.7933	54.0909	88.421	74.6988			75.5245		
15	69.9531	60.4938	79.9257	65.942			56.5217		
16	71.5026	62.3037		70.3971			96.8641		
17	88.6598	53.8117		83.5766					
18	62.234								
19									

Derivative Data

	Condition	Mean	SEM	col1_idx	col2_idx	col1	col2	p_val
1	WT naive	78.78716111111111	2.6143027747342003	0.0	1.0	WT naive	WT saline	5.4039524538823590e-06
2	WT saline	58.9409058235294	2.560113761524352	1.0	2.0	WT saline	WT K	0.0010777515811896
3	WT K	76.41952666666666	4.251886532861638	3.0	4.0	A1 ko naive	A1ko saline	0.0269524921165809
4	A1 ko naive	79.30172941176471	2.7116091436766725	4.0	5.0	A1ko saline	A1ko K	0.5606359313990721
5	A1ko saline	64.73307777777778	6.870126363846304	6.0	7.0	A2a ko naive	A2a ko saline	0.0262769488573722
6	A1ko K	69.16162727272727	4.026632015787137	7.0	8.0	A2a ko saline	A2a ko K	0.9055807377611114
7	A2a ko naive	80.41595000000001	2.8985841843044 68					
8	A2a ko saline	68.60035833333333	4.322905659189345					
9	A2a ko K	69.42009999999999	5.2529738158834505					
10								

The Generated Chart

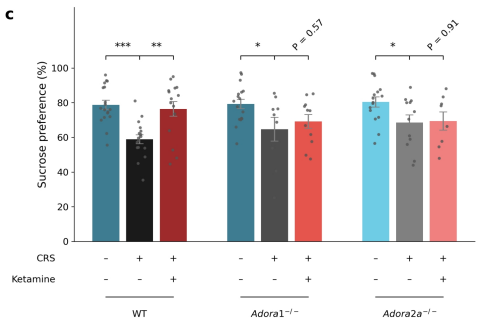
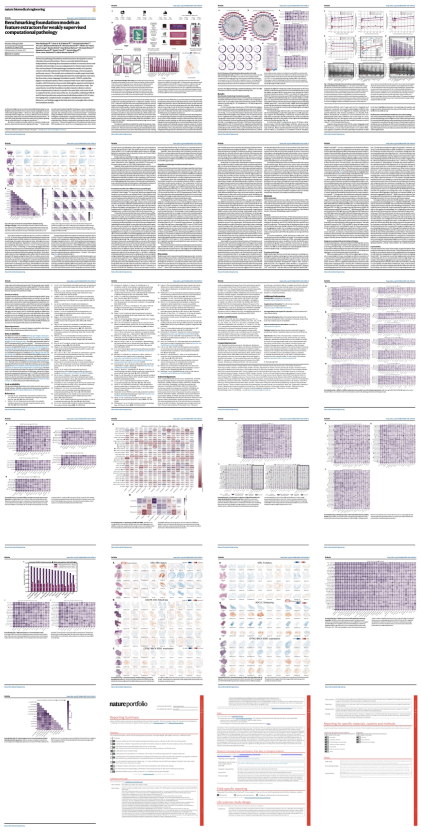


Figure 4: This academic example requires statistical reasoning. The model interprets user intent, extracts raw data points, and infers the underlying statistics, specifically deriving the Mean, Standard Error of the Mean (SEM), and the P-value between conditions.

Query

Generate a Python script to process data based on the Chart Purpose.
Chart Type:
Multiseries scatter plot
Chart Purpose:
To examine the relationship between representation of a cancer type in the pretraining dataset (percentage of slides) and downstream model performance (AUROC), showing individual model results by shape and cancer-type trends by color, and reporting correlations per cancer type.
Chart Layout:
Multiseries scatter plot (single Cartesian panel) showing individual model results as points with fitted linear trend lines per cancer type. Panel labeled 'c' in the corner.

Context Tokens:63593



Source Data

	Cancer Type	Model	Percentage	Performance
1	LUNG	ctranspath	8.4	0.7101566922966756
2	BRCA	ctranspath	7.5	0.6901946923437127
3	STAD	ctranspath	3.4	0.6727500426566878
4	CRC	ctranspath	11.7	0.6402351975254987
5	LUNG	phikon	9.7	0.7086469364758765
6	BRCA	phikon	8.8	0.6333019990370683
7	STAD	phikon	4.0	0.6658650873334706
8	CRC	phikon	5.7	0.6335703165007269
9	LUNG	uni	9.8	0.7582049929810751
10	BRCA	uni	3.3	0.6884572772690987
11	STAD	uni	6.7	0.6769115022211536
12	CRC	uni	8.3	0.6585192894528652
13	LUNG	kaiko	9.7	0.7311053739265448
14	BRCA	kaiko	8.8	0.6855187624084426
15	STAD	kaiko	4.0	0.6679383667204072
16	CRC	kaiko	5.7	0.6895842219569904
17	LUNG	prov-gigapath	45.0	0.7578128341867243
18	BRCA	prov-gigapath	2.7	0.667037192212l3
19	STAD	prov-gigapath	0.7	0.6867942247149665
20	CRC	prov-gigapath	30.0	0.6796362735454282
21	LUNG	virchow-class	6.1	0.7174955993098335
22	BRCA	virchow-class	25.0	0.6589119444462541
23	STAD	virchow-class	3.5	0.6845577051089727
24	CRC	virchow-class	3.2	0.6476261453717196
25	LUNG	virchow2-class	4.0	0.7453463548176368
26	BRCA	virchow2-class	19.0	0.7043400914608242
27	STAD	virchow2-class	3.0	0.7171710064688561
28	CRC	virchow2-class	6.0	0.6911429496854462
29	LUNG	panakeia	0.0	0.7136461434822977
30	BRCA	panakeia	82.0	0.7080816433172487
31	STAD	panakeia	0.0	0.6830453013043695
32	CRC	panakeia	10.0	0.6616418630330357
33				

Derivative Data

	cancer_type	r	p
1	LUNG	0.548585053629297	0.15913917874978992
2	STAD	-0.278840740522424 3	0.5036420596256193
3	BRCA	0.43959345798517785	0.27579154872184664
4	CRC	0.4472210806869125	0.2665607212308931
5			

The Generated Chart

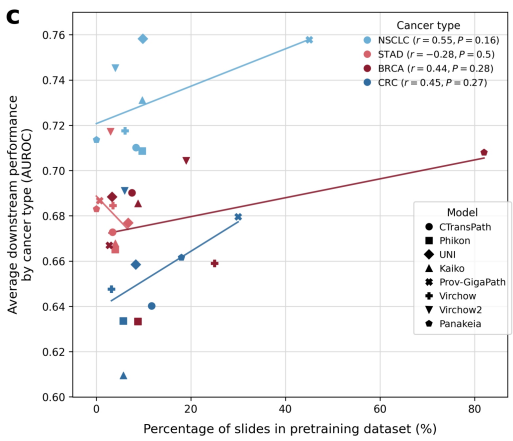


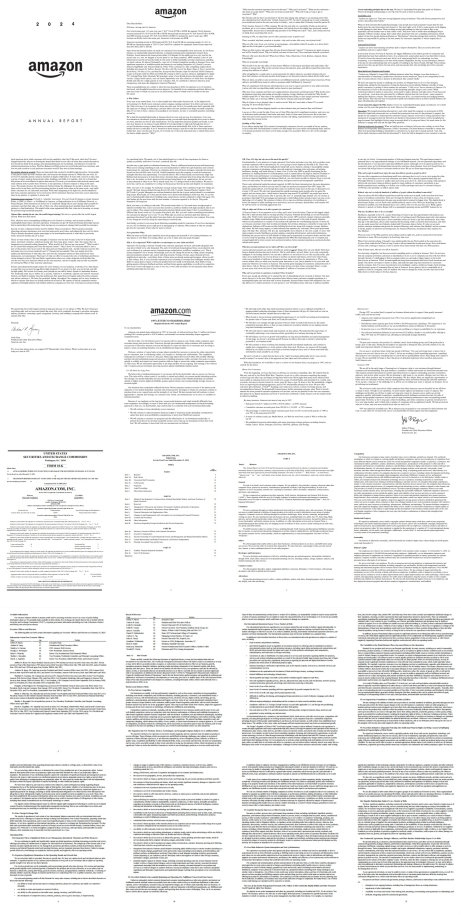
Figure 5: This academic example involves analytical reasoning. The model extracts raw data points to deduce the parameters of each conditional regression line.

Query

Generate a Python script to process data based on the Chart Purpose.
Chart Type:
Dual-Axis Combination Chart
Chart Purpose:
showing the change of amzn's (Cash Conversion Cycle, CCC) from 2016 to 2024. Ensure the data is accurate and reliable the chart design is clean. The unit uses millions of dollars.",
Chart Layout:
This layout utilizes a dual-axis design to superimpose grouped bars representing financial volumes (left axis) against lines with error bars for efficiency ratios (right axis), allowing for the simultaneous visualization of distinct metrics across a shared timeline.

Context

Tokens:458245



(Financial reports for the five years 2025, 2023, 2021, 2019, and 2017 are required in total.)

Source Data

	Fiscal Year	Avg Receivables	Revenue	Avg Inventory	Cost of Revenue	Avg Payables
1	2016	6996500000.0	135987000000	10852000000.0	121969000000	22853000000.0
2	2017	10751500000.0	177866000000	13754000000.0	111934000000	29962500000.0
3	2018	14920500000.0	232887000000	16610500000.0	139156000000	36404000000.0
4	2019	18746500000.0	280522000000	18835500000.0	165536000000	42687500000.0
5	2020	22679000000.0	386064000000	22146000000.0	233307000000	59861000000.0
6	2021	28716500000.0	469822000000	28217500000.0	272344000000	75601500000.0
7	2022	37625500000.0	513983000000	33522500000.0	288831000000	79132000000.0
8	2023	47306500000.0	574785000000	33861500000.0	304739000000	82290500000.0
9	2024	53852000000.0	637959000000	33766000000.0	326288000000	89672000000.0
10						

Derivative data

	Fiscal Year	DSO	DIO	DPO	CCC
1	2016	18.77916639090501	32.47530110109946	68.38905787536177	-17.13
2	2017	22.06322456231096	44.849732878303286	97.70322243464898	-30.79
3	2018	23.38465650723312	43.568602862973925	95.48607318405244	-28.53
4	2019	24.39192826231098	41.53149465977189	94.12416332398992	-28.2
5	2020	21.44161330763811	34.646581542774115	93.650276245462	-37.56
6	2021	22.30956085496209	37.81756712099404	101.32239924507242	-41.19
7	2022	26.719380796641133	42.362878292150086	100.00027697857917	-30.91
8	2023	30.040576041476378	40.55748525787641	98.56313927656126	-27.96
9	2024	30.810726081143144	37.772121561320056	100.31101358309223	-31.72
10					

The Generated Chart

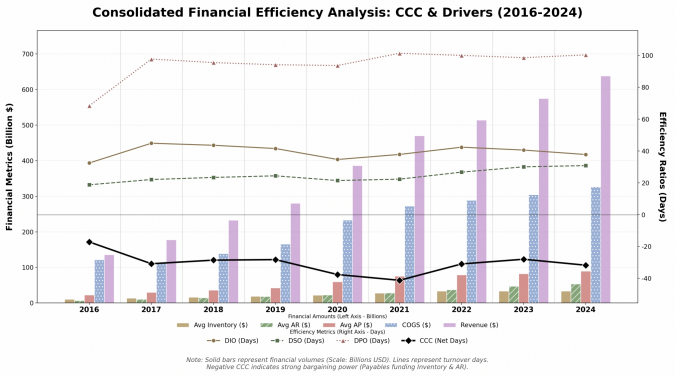


Figure 6: This Finance example demonstrates complex sequential reasoning based on year-end reports (2017 2019 2021 2023 2025). The model extracts base data and executes a multi-hop inference chain: first deriving annual averages, then inferring intermediate metrics (DIO, DSO, DPO), and finally deducing the Cash Conversion Cycle (CCC).

E Failure Case Studies

Appendix E provides qualitative failure cases corresponding to the three main failure modes observed in our experiments: incomplete reasoning, temporal coverage mismatch, and incorrect entity selection.

Bad Case 1: Correct Extraction, Missing Statistical Reasoning

Domain: Academic Model: gemini-3-pro-preview

Input

Query: Generate a Python-rendered Grouped bar chart with overlaid individual datapoints and error bars (mean +/- s.e.m.) for visualizing To compare peak extracellular adenosine responses in the mPFC across multiple ketamine dose conditions, showing group means with variability and individual data points, and to indicate which dose comparisons reach statistical significance.

Context: paper

Data Comparison

GT

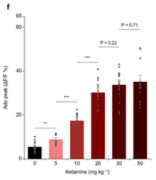
Source Data (raw points)

saline	K5	K10	K20	K30	K50
2.62	10.91	17.02	27.17	29.40	24.75
2.36	8.18	10.80	24.29	25.59	32.77
7.15	7.48	19.76	40.87	43.13	30.30
5.96	6.79	11.49	33.67	38.41	35.18
8.72	6.06	20.58	28.65	34.13	45.91
10.07	10.87	22.39	32.80	38.48	50.50
4.66	11.25	17.57	36.98	41.94	50.17
5.17	6.61	19.08	35.96	39.02	33.91
4.41	11.61	20.29	30.91	37.77	23.29
3.47		20.68	27.41	35.18	25.89
3.32		13.97	21.50	20.76	33.98
7.68		14.85	22.42	21.37	

Derived / Final Data

Derived item	GT value
saline mean/SEM/n	5.47/0.72/12
K5 mean/SEM/n	8.86/0.76/9
K10 mean/SEM/n	17.37/1.10/12
K20 mean/SEM/n	30.22/1.75/12
K30 mean/SEM/n	33.76/2.22/12
K50 mean/SEM/n	35.15/2.93/11
saline vs K5 p	p=0.0046
K5 vs K10 p	p=0.0000
K10 vs K20 p	p=0.0000
K20 vs K30 p	p=0.2230
K30 vs K50 p	p=0.7075

Chart Preview



Generated

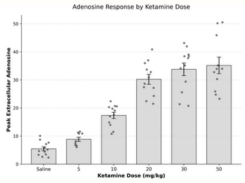
Source Data (raw points)

saline	K5	K10	K20	K30	K50
2.62	10.91	17.02	27.17	29.40	24.75
2.36	8.18	10.80	24.29	25.59	32.77
7.15	7.48	19.76	40.87	43.13	30.30
5.96	6.79	11.49	33.67	38.41	35.18
8.72	6.06	20.58	28.65	34.13	45.91
10.07	10.87	22.39	32.80	38.48	50.50
4.66	11.25	17.57	36.98	41.94	50.17
5.17	6.61	19.08	35.96	39.02	33.91
4.41	11.61	20.29	30.91	37.77	23.29
3.47		20.68	27.41	35.18	25.89
3.32		13.97	21.50	20.76	33.98
7.68		14.85	22.42	21.37	

Derived / Final Data

Derived item	Gen value
saline mean/SEM/n	5.47/0.72/12
K5 mean/SEM/n	8.86/0.76/9
K10 mean/SEM/n	17.37/1.10/12
K20 mean/SEM/n	30.22/1.75/12
K30 mean/SEM/n	33.76/2.22/12
K50 mean/SEM/n	35.15/2.93/11
saline vs K5 p	missing
K5 vs K10 p	missing
K10 vs K20 p	missing
K20 vs K30 p	missing
K30 vs K50 p	missing

Chart Preview



Analysis

1. Source extraction is correct

The model recovers all individual measurements across the six ketamine dose groups, so source-data F1 reaches 1.

2. Reasoning stops at summaries

The query asks which dose comparisons are statistically significant. GT includes pairwise p-values, but the generated data only keeps mean/SEM/n.

3. Final chart lacks significance

The bars, points, and error bars are plausible, but the significance brackets and p-value annotations required by the query are missing.

Scores

Source Data F1	Derivative Data F1	VLM Accuracy
1.000	0.429	0.385
P=1.000, R=1.000	P=0.667, R=0.316	IC=0.40, DT=0.67, PQ=0.20

Figure 7: Academic bad case: correct extraction but incomplete statistical reasoning.

Bad Case 2: Report-Year Sampling Is Mistaken as Data-Year Coverage

Domain: Finance Model: qwen3-235b-a22b

Input

Query: Generate a Python-rendered line chart showing NVIDIA's Graham's Net-Net Working Capital (NNWC) from 2016 to 2024.

Context: 

10k report will include indicators from previous years, so we input at intervals

Data Comparison

GT

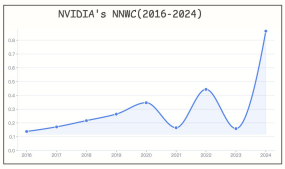
Source Data (all fiscal years)

Year	Cur. assets	Liabilities	Shares
2016	6053	2901	22760
2017	8536	4079	25960
2018	9255	3770	25280
2019	10557	3950	25000
2020	13690	5111	24720
2021	16055	11898	25120
2022	28829	17575	25350
2023	23073	19081	25070
2024	44345	22750	24940

Derived / Final Data (NNWC/share)

Year	GT NNWC / share
2016	0.1385
2017	0.1717
2018	0.2170
2019	0.2643
2020	0.3470
2021	0.1655
2022	0.4439
2023	0.1592
2024	0.8659

Chart Preview



Generated

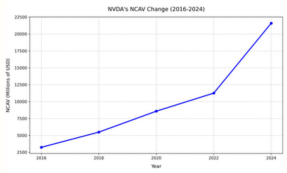
Source Data (generated)

Year	Cur. assets	Liabilities	Shares
2016	6053	2814	missing
2017	missing	missing	missing
2018	9255	3770	missing
2019	missing	missing	missing
2020	13690	5111	missing
2021	missing	missing	missing
2022	28829	17575	missing
2023	missing	missing	missing
2024	44345	22750	missing

Derived / Final Data (raw NCAV)

Year	Generated NCAV
2016	3239
2017	missing
2018	5485
2019	missing
2020	8579
2021	missing
2022	11254
2023	missing
2024	21595

Chart Preview



Analysis

1. Report years are treated as data coverage

Each 10-K contains two or three comparable fiscal years. The model only outputs 2016/2018/2020/2022/2024 and drops the in-between years.

2. NNWC is confused with raw NCAV

GT computes NNWC per share = (Current Assets − Total Liabilities) / Shares. The generated result omits Shares and plots raw NCAV dollars.

3. The line chart hides real changes

With only five sampled points and the wrong scale, the generated chart smooths over the 2021 and 2023 reversals.

Scores

Source Data F1	Derivative Data F1	VLM Accuracy
0.486	0	0.385
P=0.900, R=0.333	P=0.000, R=0.000	IC=0.60, DT=0.33, PQ=0.00

Figure 8: Finance bad case: report-year sampling is mistaken as data-year coverage.

Bad Case 3: First Ten Mentioned Ecosystems Replace Ranked Top Ten

Domain: Ecosystem Model: claude-opus-4.5

Input

Query: Create a slope chart for the top 10 Fintech ecosystems by `Total Early-Stage Funding`, comparing each ecosystem's `salary adjusted Seed = Median Seed Round / Software Engineer Salary` with `salary-adjusted Series A = Median Series A Round / Software Engineer Salary`, expressed in `engineer-years`, ordered descending by the latter, and excluding `Nur-Sultan`.

Context: report

Data Comparison

GT

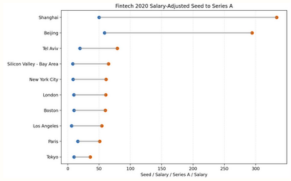
Source Data (all 28 candidates)

#	Ecosystem	Fund(B)	Seed(K)	A(M)	Sal(K)
1	Silicon Valley - Bay A…	18	1000	8	123
2	Shanghai	8.7	1500	10	30
3	New York City	8.3	850	6	98
4	Beijing	7.5	2000	10	34
5	London	6	650	4	66
6	Boston	3.9	1000	6	100
7	Los Angeles	3.6	560	5	92
8	Tel Aviv	3	1400	5.7	72
9	Paris	2.6	850	2.7	53
10	Tokyo	1.7	502	1.8	50
11	Toronto-Waterloo	1.6	500	4.3	64
12	Singapore	1.4	661	3.6	47
13	Chicago	1.2	695	3	81
14	Amsterdam-Delta	0.98	500	2.4	61
15	Stockholm	0.944	479	1.4	51
16	Sydney	0.768	404	2.8	64.5
17	Copenhagen	0.47	458	1.5	71
18	Melbourne	0.363	541	2.8	56
19	Riyadh	0.142	150	2	35
20	Curitiba	0.141	250	0.8	17
21	Frankfurt	0.12	660	1.8	59
22	Manila	0.102	360	2.3	8
23	Kuala Lumpur	0.099	187	2	12.7
24	Bahrain	0.05	70	2.5	34
25	Cairo	0.05	100	0.87	6
26	Kampala	0.05	100	1.1	10
27	Nur-Sultan	0.05	300	120	16
28	Sri Lanka	0.05	120	1.7	3

Derived / Final Data

Ecosystem	Seed yrs	A yrs
Shanghai	50	333
Beijing	58.8	294
Tel Aviv	19.4	79.2
Silicon Valley - Bay…	8.13	65
New York City	8.674	61.2
London	9.848	60.6
Boston	10	60
Los Angeles	6.087	54.3
Paris	16	50.9
Tokyo	10	36

Chart Preview



Generated

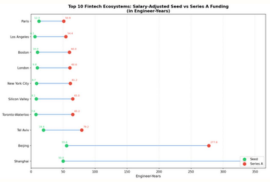
Source Data (generated 10 rows)

#	Ecosystem	Fund(B)	Seed(K)	A(M)	Sal(K)
1	Shanghai	8.7	1500	10	30
2	Beijing	7.5	2000	10	36
3	Tel Aviv	3	1400	5.7	72
4	Toronto-Waterloo	1.6	500	4.3	66
5	Silicon Valley	18	1000	8	123
6	New York City	8.3	850	6	98
7	London	6	650	4	66
8	Boston	3.9	1000	6	100
9	Los Angeles	3.6	560	5	92
10	Paris	2.6	650	2.7	53

Derived / Final Data

Ecosystem	Seed yrs	A yrs
Shanghai	50	333
Beijing	55.6	278
Tel Aviv	19.4	79.2
Toronto-Waterloo	7.58	65.2
Silicon Valley	8.13	65
New York City	8.67	61.2
London	9.85	60.6
Boston	10	60
Los Angeles	6.09	54.4
Paris	12.3	50.9

Chart Preview



Analysis

1. Candidate pool is incomplete

   GT compares all 28 candidate ecosystems in the PDF, then ranks by Total Early-Stage Funding. The model only keeps the first ten ecosystems appearing in the report.

2. Top-10 selection is wrong

   After ranking and excluding NurSultan, Tokyo should appear. Toronto-Waterloo is included instead, and several extracted numeric fields are also wrong.

3. Answers a different comparison

   The slope-chart form is close, but the city set, ordering, and salary-adjusted Seed/Series A values are not faithful to GT.

Scores

Source Data F1	Derivative Data F1	VLM Accuracy
0.288	0.550	0.538
P=0.950, R=0.170	P=0.550, R=0.550	IC=0.80, DT=0.67, PQ=0.20

Figure 9: Ecosystem bad case: first-mentioned entities replace the requested ranked top entities.