ID2C: EVALUATING AND ADVANCING DOCUMENT-TO-CHART GENERATION WITH IMPLICIT REQUIREMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generating a chart from a document requires more than locating values and writing plotting code: a system must also infer how the values should be transformed and which visual form best communicates the requested analysis. Existing text-to-chart benchmarks often make these decisions explicit, leaving their treatment by current models poorly understood. We introduce ID2C, a human-filtered benchmark of 1,000 implicit document-to-chart requests drawn from real documents and balanced across ten analytical task types. Each request leaves the construction procedure unspecified and is paired with a sample-specific rubric covering data selection, value transformation, chart-form selection, and visual encoding. Evaluation of open and proprietary models reveals persistent weaknesses in value transformation and chart-form selection, even when the generated charts are executable and visually polished. To address these failures, we construct a benchmark-independent visualization catalogue containing 199 chart forms and 744 value transformations, with each entry describing when and how it should be applied. We integrate this external knowledge source into a retrieval-augmented multi-agent harness whose form agent retrieves relevant catalogue entries before deciding the transformation and chart form. Experiments across multiple models show that the proposed harness consistently improves chart-generation quality, with the largest gains on the two planning decisions targeted by the catalogue. [1]

1 INTRODUCTION

Charts are a primary means of communicating quantitative findings in scientific, business, and public reports. By turning tables into visual comparisons, they help readers recognize trends, differences, and relationships that are difficult to extract from raw numbers alone. Producing an effective chart, however, requires both analytical reasoning and visualization expertise: one must decide what and how to plot, which quantities to derive, and how to encode them visually (Card et al., 1999). Against this backdrop, recent advances in large language models (LLMs) have enabled systems that generate executable visualizations from natural-language requests (Dibia, 2023; Yang et al., 2024), motivating a growing set of text-to-chart datasets and benchmarks (Zadeh et al., 2024; Rahman et al., 2025).

Many existing benchmarks begin with an isolated table and queries that explicitly specify much of the desired chart (Luo et al., 2021; Chen et al., 2025b). By contrast, realistic document-to-chart generation may begin with a complete document and an underspecified request, leaving two central decisions implicit. First, rather than receiving a preselected table, the model may need to determine which table, rows, columns, and values in the document support the user's request (Jain et al., 2025). The relevant evidence is therefore left unspecified, giving rise to *implicit data requirements*. Second, users may state what they want to understand without specifying what the result should show or how it should be visualized. A user might simply ask, *"Show me the trade-off between fuel efficiency and engine power."* Such a request does not prescribe a chart form or visual encoding, yet answering it effectively requires a form suited to the intended analysis, such as a scatter plot that reveals the relationship between the two variables. This gives rise to *implicit visual requirements*. The model

[1] https://huggingface.co/datasets/iclr2027anonymous-d2c/ID2C

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

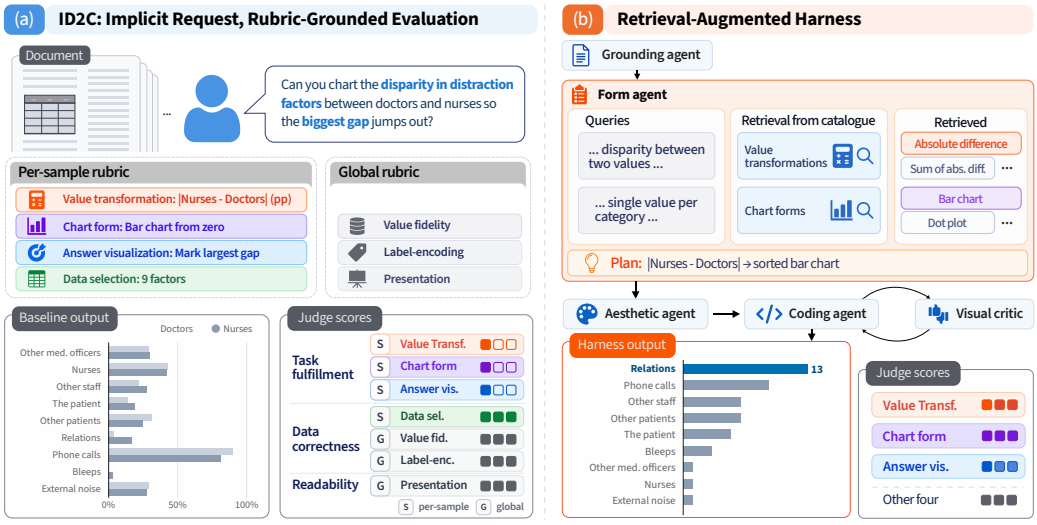

Figure 1: Overview of ID2C and our retrieval-augmented harness. (a) Each benchmark example pairs a real document with an implicit request and sample-specific rubrics. (b) Our multi-agent harness, augmented with catalogue retrieval, supplies request-relevant value-transformation and chart-form knowledge when planning the chart.

must therefore infer both which evidence and derived values answer the request and how to present them so that the intended insight becomes apparent.

To evaluate how well models handle the two forms of implicitness that arise in realistic document-to-chart generation, we introduce **ID2C** (**I**mplicit **D**ocument-to-**C**hart), a human-filtered benchmark of 1,000 implicit document-to-chart requests balanced across ten analytical task types. Each example pairs a real document with a request that specifies an information need while leaving the relevant evidence, necessary analysis, and appropriate visual form implicit, as illustrated in Figure 1(a). The two implicit requirements also motivate our evaluation design. Accordingly, we evaluate generated charts along three outcome-oriented dimensions: **data correctness**, **task fulfillment**, and **readability**. Data correctness captures the implicit data requirement by assessing whether the chart faithfully represents evidence from the source document, while task fulfillment captures the implicit visual requirement by assessing whether the chart form and visual encoding support the requested analysis. As a separate dimension, readability assesses whether the resulting answer or pattern is communicated clearly. Together, these dimensions align with the distinction between data, task, and visual-encoding abstractions in visualization analysis (Munzner, 2014).

Reliable evaluation along these dimensions is itself challenging. Exact-match and rule-based metrics may reject semantically valid charts that differ in implementation or visual form. By contrast, VLM judges offer a more flexible alternative by evaluating rendered outputs directly (Wu et al., 2025). Yet, when judging data correctness, task fulfillment, and readability based only on generic descriptions, they must reconstruct the intended analysis and visualization choices, which can lead to biased or unstable judgments (Zheng et al., 2023; Chen et al., 2024).

We therefore further decompose the three dimensions into seven fine-grained diagnostic criteria. For **data correctness**, we evaluate *data selection*, *value fidelity*, and *label and encoding fidelity*. For **task fulfillment**, we evaluate whether the model performs the appropriate *value transformation*, makes the intended *answer visible*, and selects an appropriate *chart form*. Finally, we evaluate **readability** directly from the rendered chart. In addition, each example includes sample-specific rubrics that specify the expected decision or acceptable alternatives for every applicable request-dependent criterion: value transformation, answer visibility, chart form, and data selection. Our ablation on the evaluation method shows that this criterion-level, sample-grounded evaluation aligns more closely with human judgments than holistic VLM judging with a generic rubric.

Our evaluation reveals that current open and proprietary models frequently fail at value transformation and chart-form selection, even when they generate executable and visually polished charts. These decisions require analytical and visualization knowledge that may not be reliably recalled from parametric knowledge during generation. To make such knowledge explicitly available,

we draw on the emerging Agent Skills paradigm (Li et al., 2026) and construct a benchmark-independent catalogue containing 199 chart forms and 744 analytical measures. Each entry describes when the corresponding chart form or analytical measure is useful and how it should be implemented. As illustrated in Figure 1(b), we integrate this catalogue into a multi-agent generation pipeline, where a dedicated agent decides the value transformation and chart form by formulating retrieval queries and consulting relevant catalogue entries. Experiments show that this retrieval-augmented harness consistently improves chart quality, with the largest gains on the planning decisions targeted by the catalogue.

Our contributions are as follows.

- We introduce ID2C, a human-filtered benchmark of 1,000 document-grounded requests that jointly test implicit data and visual requirements across ten task types.
- We formulate chart evaluation around data correctness, task fulfillment, and readability, decompose these dimensions into seven diagnostic criteria, and provide sample-specific rubrics that show high alignment with human judgments.
- We identify value transformation and chart-form selection as key bottlenecks and introduce a retrieval-augmented multi-agent harness supported by a benchmark-independent catalogue of chart forms and analytical measures.

2 Related Work

NL2VIS benchmarks have evolved from generating visualization specifications to producing executable chart programs. nvBench introduced a table-grounded benchmark based on natural-language questions and Vega-Zero specifications (Luo et al., 2021). VisEval and MatPlotBench shifted evaluation toward executable Python charts (Chen et al., 2025b; Yang et al., 2024), while Text2Vis expanded the range of chart types, domains, and request complexity (Rahman et al., 2025). Nevertheless, most benchmarks still provide an isolated table and a request that conveys much of the intended chart specification.

Recent work relaxes these assumptions. nvBench 2.0 addresses ambiguous requests by recognizing multiple valid data and visualization choices (Luo et al., 2025), while DOC2CHART uses financial reports and scientific papers, focusing on accurate data extraction from document context (Jain et al., 2025). Our benchmark instead draws from heterogeneous web documents and requires models to recover unstated analytical and visual decisions from requests with clear information needs.

LLM-based visualization systems have progressed from direct code generation to modular and iterative pipelines. Chat2VIS translates requests into visualization code (Maddigan & Susnjak, 2023), whereas LIDA and ChartGPT make intermediate decisions explicit (Dibia, 2023; Tian et al., 2025). Other systems refine outputs using data validation, execution feedback, or VLM-based critique (Jain et al., 2025; Yang et al., 2024; Rahman et al., 2025). Complementing them, our harness retrieves relevant value transformations and chart forms from a catalogue, providing procedural knowledge for deciding what to plot and how to encode it.

3 Implicit Document-to-Chart Benchmark

3.1 Task Definition

We study *implicit document-to-chart generation*. Given a document D containing one or more tables and a natural-language request q describing an analytical goal, a chart-generation system G produces executable code c:

$$c = G(D, q). \tag{1}$$

We execute c in a sandbox and denote the resulting chart image by I when execution succeeds. The target table and values are not separately provided to G; the system must identify the relevant evidence from D and generate a chart that communicates the analysis requested by q.

The task combines two forms of implicitness. Under implicit data requirements, the request does not identify the exact table, rows, or columns to use. The system must therefore locate the relevant

Table 1: The ten analytical query types in our benchmark.

Query type	Analytical goal
`growth_speed`	Compare rates of growth or decline rather than raw levels.
`momentum_turning`	Identify when growth accelerates, slows, peaks, or reverses.
`baseline_index`	Compare trajectories relative to distinct starting points or a common reference.
`paired_gap`	Compare entity-level gaps between two matched measurements.
`composition_snapshot`	Reveal part-to-whole structure at one point or condition.
`composition_compare`	Compare part-to-whole structure across groups or snapshots.
`mix_trend`	Show how composition changes over ordered time.
`concentration_long_tail`	Determine whether a few entities dominate an additive total.
`tradeoff`	Reveal the relationship between two quantitative measures.
`stability_volatility`	Compare consistency or variability rather than average level.

evidence in the source document. Under implicit visual requirements, the request does not prescribe either the quantity to visualize or the chart form and visual encoding. The system must determine whether to show raw values or derived quantities such as differences, growth rates, ratios, or shares, and then select a visual structure that makes the requested answer or pattern readily apparent.

3.2 Benchmark Construction Pipeline

We construct the benchmark in five stages. First, to capture both source diversity and the natural distribution of real-world documents, we collect documents from FinePDFs (Kydlíček et al., 2025), extract their tables together with surrounding document context, and apply basic structural filters. Second, given the ten predefined analytical query types summarized in Table 1, an LLM examines each remaining table against type-specific criteria and assigns it to the types it can support. Third, for each selected table–type pair, an LLM constructs a fully specified internal task description. This description identifies the relevant table, rows, and columns; states any required value transformation; and specifies suitable chart forms and the insight that the resulting visualization should communicate. The same step produces sample-specific rubric anchors recording these requirements.

Fourth, a separate LLM paraphrases the internal description into a natural implicit request. The paraphrase preserves the user's analytical goal while omitting procedural details such as the exact target columns, transformation formula, chart form, and low-level visual instructions. Finally, automated consistency checks and human review remove requests that disclose the intended procedure or are ambiguous, unanswerable, unnatural, or inconsistent with the source document or rubric. This process ensures that the construction procedure remains implicit while the underlying information need remains well defined.

3.3 Benchmark Composition

Our benchmark contains 1,000 examples drawn from 1,000 distinct tables in 960 documents, with exactly 100 examples per query type. Source documents span 345–18,791 words (median 4,738) and typically contain multiple tables, with the target appearing as late as the 78th; target tables range from 3 to 53 rows and 3 to 23 columns. Target tables are also spread throughout their documents (median relative position 54.4%), so locating the relevant evidence cannot rely on positional shortcuts. Detailed statistics are provided in Appendix A.3.

Figure 2 shows the distribution of source-document genres. The benchmark spans nine recurring genres across four broad families. An additional *Other* category accounts for 0.6% of documents, and no single genre accounts for more than 25.4%.

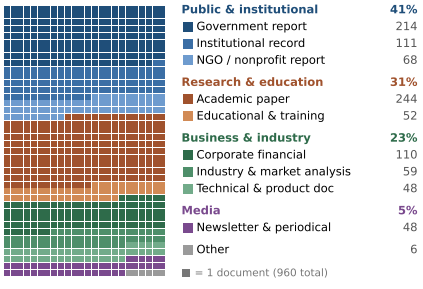

Figure 2: Distribution of document genres in ID2C. Each square represents one source document, with colors indicating genres.

3.4 CRITERION-DECOMPOSED, RUBRIC-BASED EVALUATION

A generated chart must remain faithful to the source document, satisfy the analytical request, and communicate its result clearly. We operationalize these three dimensions through seven diagnostic criteria, so the judge need not reconstruct the intended analysis when assessing its visual realization.

Data correctness. We decompose data correctness into three criteria. *Data selection* evaluates whether the chart uses the table, entities, periods, and variables needed to answer the request. *Value fidelity* checks whether plotted magnitudes and scales represent the source values or their intended transformations. *Label and encoding fidelity* checks whether axis labels, category names, units, legends, annotations, and visual encodings correctly identify and represent the displayed values.

Task fulfillment. We evaluate task fulfillment using up to three criteria. *Value transformation* evaluates whether the model performs the value transformations or analytical operations required to answer the request. For discrete-answer tasks, *answer visibility* evaluates whether the requested entity or period is readily identifiable; this criterion is omitted for pattern-answer tasks. *Chart form* evaluates whether the visual structure is appropriate for revealing the requested comparison, trend, composition, concentration, or relationship.

Readability. We evaluate readability directly from the rendered chart, asking whether the intended answer or pattern can be understood clearly given its legibility and visual organization, including text overlap, clutter, spacing, color use, and layout.

For each example, a VLM judge receives the request, the source table from which the task was constructed, the generated code, the rendered chart, and the applicable criterion rubrics. The judge scores each criterion as unsatisfied, partially satisfied, or fully satisfied (1–3), which we map to $\{0, 50, 100\}$ for reporting.

4 RETRIEVAL-AUGMENTED CHART GENERATION HARNESS

Our evaluation of current models on ID2C (Section 5.2) shows a consistent pattern: most models, open-weight or proprietary, score lowest on value transformation and chart-form selection, even when their charts render correctly and read well. These decisions require knowledge that is rarely stated in either the document or the request, such as how raw values should be converted into a share or growth rate and which chart form best reveals the requested pattern. We hypothesize that relevant analytical and visualization knowledge is not reliably recalled from model parameters and applied during chart generation. Our key intervention is therefore to externalize this knowledge in a structured visualization catalogue, which serves as the harness's external knowledge source.

Prior work has shown the effectiveness of modular and agentic pipelines for visualization generation (Dibia, 2023; Yang et al., 2024; Rahman et al., 2025). We adopt this structure and introduce a *form agent*, a component of the multi-agent pipeline that performs catalogue-augmented planning.

4.1 CATALOGUE-AUGMENTED FORM AGENT

The external catalogue organizes visualization knowledge around the two decisions that motivated the harness. A *chart-form card* describes a chart form, when it should be used, and how data are visually encoded, whereas a *value-transformation card* describes a derived quantity, the analytical purpose it serves, and its formula. We construct this catalogue independently of the benchmark. We first collect candidate concepts by traversing visualization and quantitative-measure categories and list pages on Wikipedia. An LLM then determines whether each candidate represents either a chart form that maps data to visual marks or a value transformation that computes a derived quantity from recorded values. For each retained concept, an LLM writes a compact card using the corresponding article as background. This process yields 199 chart-form cards and 744 value-transformation cards, which together provide a retrievable source of visualization and analytical knowledge.

At inference time, the *form agent* consults this catalogue through two successive model calls. In the first call, it formulates separate, general retrieval queries for the value transformation and chart form, which are used to retrieve relevant cards from the corresponding catalogues. In the second call, the agent examines the retrieved cards and decides how to transform the data and which chart form to use for the request. The retrieved cards are advisory rather than prescriptive, allowing the agent to

Table 2: Evaluation of various LLMs on ID2C, grouped by the human-evaluation axis each criterion maps to. Render failures score 0 on every criterion. *Avg.* denotes the unweighted mean of the three criteria within each axis, and *Overall* denotes the unweighted mean of all seven criteria.

	Task Fulfillment				**Data Correctness**				**Readability**	**Overall**
	Value Transf.	Answer Vis.	Chart Form	**Avg.**	Data Sel.	Value Fid.	Label-Enc.	**Avg.**	Presentation	
Closed-source										
GPT-5.6-Sol	82.0	**92.4**	68.7	81.0	86.7	90.2	71.8	**82.9**	94.5	83.7
Claude Opus 5	**87.2**	87.1	**76.3**	**83.5**	**88.0**	90.8	68.0	82.3	**96.5**	**84.8**
Open-weight										
GLM-5.3	83.5	84.5	71.1	79.7	86.8	88.5	**72.8**	82.7	92.8	82.9
KIMI K2.6	71.2	82.7	61.2	71.7	85.3	87.8	60.4	77.8	90.7	77.0
DeepSeek-V4-Flash	55.1	65.2	45.5	55.3	87.2	89.3	66.7	81.0	81.2	70.0
Gemma4-31B	53.5	53.4	47.0	51.3	85.7	**92.3**	64.0	80.7	63.4	65.6
Gemma4-26B-A4B	53.0	57.1	45.0	51.7	82.5	89.5	63.6	78.5	69.0	65.7
Qwen3.6-27B	49.5	49.5	40.0	46.3	80.6	85.5	60.1	75.4	60.0	60.7
Qwen3.5-122B-A10B	47.1	60.6	40.5	49.4	82.4	86.6	57.3	75.4	63.2	62.5
Qwen3.5-35B-A3B	44.2	60.8	37.6	47.5	77.0	81.1	54.1	70.7	60.5	59.3
Qwen3.5-27B	52.2	55.9	44.9	51.0	83.8	88.5	58.8	77.0	60.9	63.6
GPT-OSS-120B	59.0	57.9	46.8	54.6	82.4	85.8	53.7	74.0	57.4	63.3
Nemotron 3 Super	44.0	47.3	39.0	43.4	80.0	85.8	60.2	75.3	57.1	59.0
Mistral Small 4	46.8	48.5	41.1	45.4	76.3	81.2	50.9	69.5	54.1	57.0

disregard them when none is appropriate. Further details on card representation and retrieval are provided in Appendix B.2.

4.2 Multi-Agent Generation Pipeline

Our final multi-agent pipeline integrates the catalogue-augmented *form agent* with agents for data grounding, aesthetic design, coding, and visual critique. Given a document D and request q, these components successively ground the relevant data, plan the chart with retrieved knowledge, determine its visual treatment, and produce executable code c. Executing c yields a chart image I, which may be revised in response to visual feedback.

The *grounding agent* locates the relevant table and values in D and passes the selected values unchanged to the *form agent*. Given q and the selected values, the *form agent* follows the catalogue-augmented process in Section 4.1, retrieving relevant cards before deciding the value transformation and chart form. It then passes the resulting chart plan to an *aesthetic agent*, which determines layout, color, and text placement without changing the planned transformation or chart form.

Finally, a *coding agent* receives the selected data, chart plan, and aesthetic specification, performs the transformation in code, and generates c. After execution, a *visual critic* examines q, c, and I and identifies concrete visual problems and corresponding fixes. The *coding agent* revises the chart until the *visual critic* is satisfied or the revision budget is exhausted. Further implementation details, including the prompts, input–output specifications, and interaction protocol for each agent, are provided in Appendix B.3.

5 Experiments

5.1 Experimental Setup

We evaluate 14 LLMs for chart generation, spanning proprietary and open-weight families. Specifically, we consider two proprietary models, GPT-5.6-Sol (OpenAI, 2026) and Claude Opus 5 (Anthropic, 2026), alongside 12 open-weight models: GLM-5.3 (Z.ai, 2026), KIMI K2.6 (Moonshot AI, 2026), DeepSeek-V4-Flash (DeepSeek-AI et al., 2026), two Gemma4 variants (Gemma4-31B and Gemma4-26B-A4B) (Gemma Team et al., 2026), four Qwen variants (Qwen3.6-27B (Qwen Team, 2026b), Qwen3.5-122B-A10B, Qwen3.5-35B-A3B, and Qwen3.5-27B (Qwen Team, 2026a)), GPT-OSS-120B (OpenAI et al., 2025), Nemotron 3 Super (NVIDIA et al., 2026), and Mistral Small 4 (Mistral AI, 2026). Model configurations are provided in Appendix C.1. Generated code is executed in a `matplotlib` sandbox, and outputs that fail to render receive the minimum score on all criteria.

Table 3: Performance of chart-generation pipelines built on Qwen3.6-27B and Gemma4-31B on ID2C.

	Task fulf.	Data corr.	Readability	Overall
Qwen3.6-27B				
Single-shot	46.3	75.4	60.0	60.7
Multi-agent pipeline	60.9	**77.5**	**76.5**	70.2
+Catalogue retrieval (Ours)	**70.5**	77.2	70.2	**73.3**
Gemma4-31B				
Single-shot	51.3	**80.7**	63.4	65.6
Multi-agent pipeline	66.8	79.0	**77.0**	73.5
+Catalogue retrieval (Ours)	**73.6**	77.9	71.8	**75.2**

All generated charts are scored by a Qwen3.8-27B Qwen (2026) VLM judge under the proposed criterion-decomposed, rubric-anchored protocol of Section 3.4. Details of evaluation are specified in Appendix C.2. We report the overall mean score and its decomposition into data correctness, task fulfillment, and readability.

5.2 Evaluation of Models on ID2C

Table 2 reports criterion- and dimension-level results for all evaluated models, together with their overall scores. Three findings stand out.

Overall performance. Claude Opus 5 achieves the highest overall score of 84.8, followed closely by GPT-5.6-Sol and GLM-5.3. Recent large-scale open-weight models substantially narrow the gap to proprietary models, while smaller models generally trail the leading systems. Further analysis shows that 71.4% of Claude Opus 5 generations fail to achieve perfect scores across all criteria. Thus, considerable room for improvement remains even for the leading proprietary models, particularly in task fulfillment and data correctness, the two dimensions corresponding to the implicit requirements studied in ID2C.

Implicit requirements. Across most models, task fulfillment is weaker than data correctness, with *value transformation* and *chart form* emerging as the main weaknesses. Models often identify and faithfully reproduce the relevant source data, yet fail to infer the derived quantity or visual form needed to make the requested comparison apparent. This pattern directly motivates our proposed pipeline, which explicitly plans these two decisions using retrieved visualization knowledge.

Readability. Like task fulfillment, readability generally improves with overall model capability. Its relationship with parameter scale, however, is not monotonic. Models with native vision capabilities can outperform substantially larger text-only models; for example, Gemma4-26B achieves higher readability than GPT-OSS-120B. This suggests that native visual understanding may provide useful priors for chart organization and presentation, partially offsetting the advantage of greater scale.

5.3 Evaluation of the Multi-Agent Pipeline

Table 3 shows the evaluation of the multi-agent pipeline with both Qwen3.6-27B and Gemma4-31B. Across both backbones, the multi-agent pipeline consistently outperforms single-shot generation, and catalogue retrieval provides a further overall improvement. Relative to single-shot generation, our full harness improves the overall score by 12.6 points with Qwen3.6-27B and 9.6 points with Gemma4-31B. The gains from catalogue retrieval are concentrated in task fulfillment, particularly value transformation and chart-form selection, which are the two planning decisions directly supported by the catalogue. This consistent pattern indicates that external analytical and visualization knowledge helps the form agent infer how implicit requests should be analyzed and visualized.

Catalogue retrieval reduces readability relative to the multi-agent pipeline alone, reflecting its more frequent use of structurally complex forms such as Pareto and stacked-area charts. Nevertheless, the full harness remains substantially more readable than single-shot generation, while its improvements in task fulfillment yield a clear overall gain.

Table 4: Human alignment of different judge designs. Values report linearly weighted Cohen's κ for pairwise verdicts. Agreement between human raters is included as a reference.

	Task fulfillment	Data correctness	Readability	Overall
Human vs. human	0.365	0.094	0.448	0.519
Ours	**0.383**	**0.247**	**0.459**	**0.611**
w/o criterion decomposition	0.148	0.085	0.459	0.530
w/o sample-specific rubrics	0.241	0.236	0.459	0.513
w/o criterion-wise inference	0.341	0.152	0.323	0.442

6 Analysis

6.1 Judge Design Ablation

LLM-based judges offer a scalable means of evaluating complex generated artifacts, but their agreement with human assessments can depend substantially on how the evaluation is structured. We therefore use human pairwise preferences to validate our proposed VLM judge method and its three key design choices: criterion-level decomposition, sample-specific rubric anchors, and independent inference for each criterion.

We construct 106 pairwise comparisons using outputs generated by Qwen3.6-27B and Claude Opus 5 for the same randomly sampled instances from ID2C. For each judge variant, we measure agreement with human pairwise judgments using linearly weighted Cohen's κ. Agreement between two independent groups of human raters serves as an inter-rater reference. Further details of the human-alignment protocol are provided in Appendix C.3.

Table 4 reports the resulting agreement scores. Replacing the diagnostic criteria with dimension-level prompts yields the lowest agreement on task fulfillment and data correctness, demonstrating the benefit of decomposing broad evaluation dimensions into fine-grained criteria. Replacing sample-specific rubric anchors with generic definitions primarily reduces agreement on task fulfillment, as the judge no longer receives explicit guidance about the intended value transformation and chart form for each request. Finally, evaluating all criteria jointly in a single inference call lowers agreement across every dimension, supporting independent criterion-wise inference.

Notably, our evaluation protocol matches or exceeds human–human agreement on readability, task fulfillment, and overall preference. This shows that, even on a complex task for which independent human raters do not always agree, the judge aligns with human preferences at least as consistently as another group of human raters does. Taken together, these results indicate that criterion-level decomposition, sample-specific anchors, and independent criterion-wise inference yield an evaluation method that aligns closely with human judgments despite the inherent difficulty of the task.

6.2 Impact of Implicit Visual Requirements

We isolate the effect of implicit visual requirements by comparing each benchmark request with its fully specified precursor, an intermediate artifact produced during benchmark construction. This explicit version states the required value transformation, the intended answer or pattern to reveal, and an appropriate chart form, thereby providing the decisions evaluated under task fulfillment. All other inputs and evaluation settings remain unchanged.

Table 5: Implicit versus explicit requests. The latter specifies all task-fulfillment components.

	Implicit	Explicit	Δ
Task fulf.	46.3	**85.4**	**+39.1**
Data corr.	75.4	73.1	−2.3
Readability	60.0	60.2	+0.2
Overall	60.7	76.5	+15.8

As shown in Table 5, providing these decisions substantially improves task fulfillment, while readability remains nearly unchanged. This concentrated improvement indicates that the model can carry

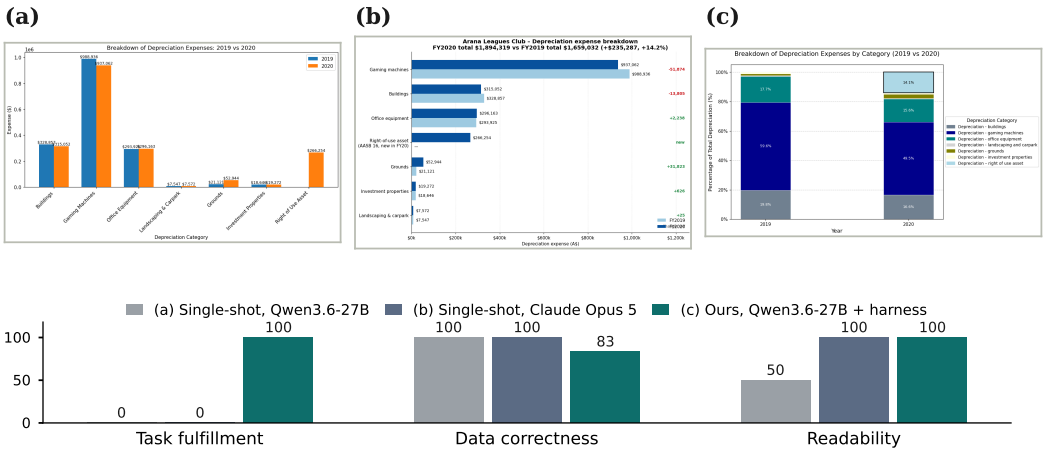

Figure 3: Generated charts for a sample of query type `composition_compare`: (a) Qwen3.6-27B, single-shot; (b) Claude Opus 5, single-shot; (c) Qwen3.6-27B with our harness. Bottom: per-dimension judge scores.

out the intended analysis, make the answer visible, and realize an appropriate chart form when the relevant requirements are specified, but does not reliably infer them from an implicit request. Therefore, evaluations based only on explicit instructions may provide an incomplete picture of model performance, because success on fully specified chart-generation tasks does not necessarily transfer to realistic requests that leave key analytical and visual decisions implicit. This gap also suggests that relevant analytical and visualization knowledge is not consistently recalled and applied during planning, which our retrieval-augmented harness addresses by supplying pertinent knowledge.

6.3 Case Study

Figure 3 illustrates a typical failure underlying the task-fulfillment gap. When asked how the *structure* of depreciation costs shifted, both single-shot systems extract every value correctly but plot the raw dollar amounts, obscuring the shift in relative shares and consequently receiving a task-fulfillment score of zero. Our harness retrieves the within-group-share transformation and 100%-stacked encoding from the catalogue, turning the same values into a chart that directly answers the question. This example motivates ID2C: user requests rarely specify the appropriate quantity transformation or visual form. Our retrieval-augmented harness thus closes the task-fulfillment gap on the same backbone by supplying the missing visualization knowledge.

7 Conclusion

We introduced ID2C, a benchmark designed to reflect the implicit data and visual requirements of realistic document-to-chart generation. Its requests specify an analytical goal while leaving the relevant evidence, necessary value transformation, and appropriate visual form unstated. By combining criterion-level evaluation with sample-specific rubrics, our evaluation aligns more closely with human judgments than holistic judging based on generic criteria. Experiments across open and proprietary models show that models ground source data more reliably than they fulfill the intended analytical task, with value transformation and chart-form selection emerging as the primary bottlenecks. Motivated by these findings, we developed a retrieval-augmented multi-agent harness that decomposes chart generation into specialized stages and supplies its form agent with relevant knowledge from a benchmark-independent catalogue of value transformations and chart forms. The harness consistently improves chart quality, with its largest gains concentrated in the planning decisions targeted by the catalogue. Overall, our findings suggest that better chart generation depends not only on producing polished visualizations, but on correctly inferring the analysis a user intends and bringing the right knowledge into the planning process before a chart is drawn.

AI USE STATEMENT

In this work, we used generative AI tools for generating synthetic data sets, implementing methods, assisting with translation, and cleaning and reformatting data sets. We did not use generative AI tools to develop theoretical models or conceptual frameworks, propose or refine hypotheses, design or provide feedback on research methodology or experiments, support qualitative or thematic data analysis, or interpret results. Formulating mathematical claims, providing critical ingredients for proving mathematical claims, and assisting in the writing of proofs were not applicable to this work. Additionally, we used generative AI tools to create or edit software code, summarize or analyze existing literature, search for information, and identify relevant literature. The authors reviewed every AI-assisted output before including it, using checks suited to each type of task. AI-assisted code was tested and reviewed; literature suggestions and factual claims were verified against the original sources; synthetic and reformatted data were validated programmatically and inspected manually; and translations were checked and revised by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Anthropic. Introducing Claude Opus 5, 2026. URL https://www.anthropic.com/news/claude-opus-5. Accessed: 2026-09-26.

S.K. Card, J. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Elsevier Science, 1999. ISBN 9781558605336.

Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. Humans or llms as the judge? A study on judgement bias. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 8301–8327, 2024.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 2025a.

Nan Chen, Yuge Zhang, Jiahang Xu, Kan Ren, and Yuqing Yang. Viseval: A benchmark for data visualization in the era of large language models. *IEEE Trans. Vis. Comput. Graph.*, 31(1):1301–1311, 2025b.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chengyu Hou, Chenhao Xu, Chenze Shao, Chong Ruan, Conner Sun, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Donghao Li, Dongjie Ji, Erhang Li, Fang Wei, Fangyun Lin, Fangzhou Yuan, Feiyu Xia, Fucong Dai, Guangbo Hao, Guanting Chen, Guoai Cao, Guolai Meng, Guowei Li, Han Yu, Han Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoling Zhang, Haoming Luo, Haoran Wei, Haotian Yuan, Haowei Zhang, Haowen Luo, Haoyu Chen, Haozhe Ji, Hengqing Zhang, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, J Yang, JQ Zhu, Jia Luo, Jia Song, Jia Yu, Jialiang Huang, Jialu Cai, Jian Liang, Jiangting Zhou, Jiasheng Ye, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jieyu Yang, Jin Chen, Jin Yan, Jingchang Chen, Jingli Zhou, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jingzi Zhou, Jinhua Zhu, Jiping Yu, Joseph Sun, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junmin Zheng, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Leyi Xia, Li Zhang, Liang Zhao, Lihua Guo, Lingxiao Luo, Linwang Ma, Linyan Zhu, Litong Wang, Liyu Cai, Liyue Zhang, Longhao Chen, MS Di, MY Xu, Max Mei, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Mingxu Zhou, Minmin Han, Ning Wang, Panpan Huang, Panpan Wang, Peixin Cong, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Qiwei Jiang, Rui Tian, Ruifan Xu, Ruijie Lu, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqian Chen, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, Ruyi Chen, SH Liu, Shanghao Lu, Shangmian Sun, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoheng Nie, Shaoqing Wu, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu

Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Shuying Yu, Songyang Zhou, Tao Ni, Tao Yun, Tian Jin, Tian Pei, Tian Ye, Tianle Lin, Tianran Ji, Tianyi Cui, Tianyuan Yue, Tingting Yu, Tun Wang, W Zhang, WL Xiao, Wangding Zeng, Wei An, Weilin Zhao, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjing Yao, Wenjun Gao, Wenkai Yang, Wenlve Huang, Wenqing Hou, Wentao Zhang, Wenting Ma, Xi Gao, Xiang He, Xiangwen Wang, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingchen Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyu Zhang, Xu Chen, Xuanyu Wang, Xuecheng Su, Xueyin Chen, Xuheng Lin, Xuwei Fu, YC Yan, YQ Wang, YW Ma, Yanfeng Luo, Yang Zhang, Yanhong Xu, Yanru Ma, Yanwen Huang, Yao Li, Yao Li, Yao Xu, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Shao, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yijia Wu, Yiliang Xiong, Yiling Ma, Ying He, Ying Tang, Ying Zhou, Yingjia Luo, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiang Zhang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, YuKun Li, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuanhao Li, Yuduan Wang, Yuehan Yang, Yuer Xu, Yuhan Wu, Yuhao Meng, Yuheng Zou, Yukun Zha, Yunfan Xiong, Yupeng Chen, Yuping Lin, Yuqian Cao, Yuqian Wang, Yushun Zhang, Yuting Yan, Yutong Lin, Yuxian Gu, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuxuan Zhou, Yuyang Zhou, Yuzhen Huang, ZF Wu, Zehao Wang, Zehua Zhao, Zehui Ren, Zekai Zhang, Zhangli Sha, Zhe Fu, Zhe Ju, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zheren Gao, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhixuan Chen, Zhiyu Wu, Zhizhou Ren, Zhongyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihua Qu, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Ziyi Wan, Zizheng Pan, and Zongqing Yao. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Victor Dibia. LIDA: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), ACL 2023, Toronto, Canada, July 10-12, 2023*, pp. 113–126, 2023.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, Mayank Chaturvedi, Aditya Chawla, Victor Cotruta, Alice Coucke, Phil Culliton, Robert Dadashi, Lucas Dixon, Mohamed Elhawaty, Utku Evci, Clément Farabet, Johan Ferret, Filippo Galgani, Sertan Girgin, Jean-Bastien Grill, Maarten Grootendorst, Jiaxian Guo, Cassidy Hardin, Yanzhang He, Steven M. Hernandez, Omri Homburger, Léonard Hussenot, Juyeong Ji, Armand Joulin, Aishwarya Kamath, Parnian Kassraie, Olivier Lacombe, Preethi Lahoti, Gaël Liu, Gus Martins, Luciano Martins, Tatiana Matejovicova, Ramona Merhej, Nikola Momchev, Sneha Mondal, Ryan Mullins, Sindhu Raghuram Panyam, Shreya Pathak, Sarah Perrin, André Susano Pinto, Etienne Pot, Angéline Pouget, Alexandre Ramé, Sabela Ramos, Douglas Reid, David Rim, Morgane Rivière, Karsten Roth, Louis Rouillard, Omar Sanseviero, Pier Giuseppe Sessa, Shane Settle, Danila Sinopalnikov, Sara Smoot, Piotr Stanczyk, Andreas Steiner, Lawrence Stewart, Ilya Tolstikhin, Michael Tschannen, Anton Tsitsulin, Nino Vieillard, Renjie Wu, Pingmei Xu, Haichuan Yang, Edouard Yvinec, Biao Zhang, Li Zhang, Joe Zou, Nicolas Aagnes, Abdelrahman Abdelhamed, Jakub Adamek, Shivani Agrawal, Shubham Agrawal, Ibrahim Alabdulmohsin, Jean Baptiste Alayrac, Uri Alon, Chandramouli Amarnath, Ankesh Anand, Chrysovalantis Anastasiou, Setareh Ariafar, François-Xavier Aubet, Kyriakos Axiotis, Federico Barbero, Joelle Barral, Alexei Bendebury, Urs Bergmann, Stanley Bileschi, Kat Black, Mathieu Blondel, Sebastian Borgeaud, Arthur Bražinskas, Ryan Burnell, Robert Busa-Fekete, Mu Cai, Daniele Calandriello, Glenn Cameron, Charlotte Caucheteux, Rahma Chaabouni, Garima Chadha, Jetha Chan, Blake Jianhang Chen, Jesse Chen, Lin Chen, Xu Chen, Derek Cheng, Tzu hsiang Chien, Nikolai Chinaev, Yi Chou, Zhaohui Chu, Benjamin Coleman, Pooja Consul, Sam Conway-Rahman, Scott Crowell, Dylan Cutler, Vivek Dani, Samira Daruki, Anil Das, Daniel Deutsch, Nishanth Dikkala, Li Ding, Qiuhan Ding, Shenil Dodhia, Konstantin Donhauser, Tulsee Doshi, Anca Dragan, Alex Druinsky, Sahil Dua, Zoltan Egyed, Danielle Eisenbud, Daniel Eppens, Cindy Fan, Bahare Fatemi, Yassir Fathullah, Vlad Feinberg, Milen Ferev, Sebastian Flennerhag, Takumi Fujimoto, João Gabriel Oliveira, Isaac Galatzer-Levy, João Gante, Simon Geisler, Soham Ghosal, Antonious M. Girgis, Tamara von Glehn, Alec Go, Alhaad Gokhale, Alex Grills, Yiming Gu, Mayank Gupta, Pramod Gupta, Guru Guruganesh, Raia Hadsell, Hamza Harkous, Jitendra Harlalka, Demis Hassabis, Anja Hauth, Joe Heyward, Arian Hosseini, Chih-Yang Hsia, I-Hung Hsu, Xiaopeng Huang, Yangsibo Huang,

Kevin Hui, Adrian Hutter, Te I, Fotis Iliopoulos, Advait Jain, Ganesh Jawahar, Ziwei Ji, Qilin Jin, Melvin Johnson, Kandarp Joshi, Arun Kandoor, Wang-Cheng Kang, Koray Kavukcuoglu, Mehran Kazemi, Kathleen Kenealy, Amr Khalifa, Phoebe Kirk, Ivan Korotkov, Suraj Kothawade, Vitaly Kovalev, Neel Kovelamudi, Adam Kraft, Ravin Kumar, Vivek Kumar, Harish Kuppam, Justin Lannin, Chen-Yu Lee, Seungji Lee, Dmitry Lepikhin, Alon Levkovitch, Dongdong Li, Qiujia Li, Valentin Liévin, Ethan Lin, Ziqian Lin, Casper Liu, Tianlin Liu, Tianqi Liu, Xin Liu, Ivan Lobov, Mayank Lunayach, Min Ma, Gagan Madan, Andrii Maksai, Eric Malmi, Michal Matuszak, Daniel McDuff, Gaurav Menghani, Maciej Mikuła, Daniil Mirylenka, Karolis Misiunas, Vedant Misra, Andreea Mitran, Kareem Mohamed, Maksim Mukha, Eric Noland, James O'Donnell, Brendan O'Donoghue, Kate Olszewska, Bernett Orlando, Wanqiong Pan, Rina Panigrahy, Unnati Parekh, Nicolas Perez-Nieves, Chunjong Park, Eric Paskie, Liqian Peng, Bryce Petrini, Slav Petrov, Jonas Pfeiffer, Bilal Piot, Martyna Plomecka, Siim Poder, Octavio Ponce, Arijit Pramanik, David Racz, Anish Rajan, Michelle Ramanovich, Anand Rao, Marvin Ritter, Vitor Rodrigues, Evan Rosen, Mikołaj Rybiński, Noveen Sachdeva, Michaël E. Sander, Rohit Sathyanarayana, Sagar Savla, Samuel Schmidgall, Tal Schuster, George Scrivener, Benoit Seguin, Andrew Sellergren, Aliaksei Severyn, Izhak Shafran, Dhruv Shah, Bobak Shahriari, Yuan Shangguan, Ashish Shenoy, Pradeep Shenoy, Rakesh Shivanna, Pauline Sho, Lucas Spangher, Wojciech Stokowiec, Tim Strother, Yao Su, Yinghao Sun, Mukund Sundararajan, Andrea Tacchetti, Mor Hazan Taege, Pouya Tafti, Jean Tarbouriech, Chetan Tekur, Shantanu Thakoor, Rahul Thapa, Madeleine Traverse, Lenart Treven, Tao Tu, Chien Te Tung, Çağlar Ünlü, Petar Veličković, Malini Pooni Venkat, Sagar Gubbi Venkatesh, Vidya Venkiteswaran, Francesco Visin, Alex Vitvitskyi, Kiran Vodrahalli, Weiyi Wang, Xin Wang, Tris Warkentin, Jan Wassenberg, John Wieting, Cindy Wu, Lechao Xiao, Hao Xu, Yuhui Xu, Fuzhao Xue, Arun Yadav, Jun Yan, Antoine Yang, Lin Yang, Ming-Hsuan Yang, Ziyu Ying, Jae Hyeon Yoo, Morteza Zadimoghaddam, Sajjad Zafar, Fred Zhang, Jiageng Zhang, Jianyi Zhang, Xiaofan Zhang, Chao Zhao, David Zhou, and Chen Zou. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Akriti Jain, Pritika Ramu, Aparna Garimella, and Apoorv Saxena. Doc2chart: Intent-driven zero-shot chart generation from documents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 34948–34963, 2025.

Hynek Kydlíček, Guilherme Penedo, and Leandro von Werra. Finepdfs. `https://huggingface.co/datasets/HuggingFaceFW/finepdfs`, 2025.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, Chujun Tao, Binxu Li, Xuandong Zhao, Hejia Geng, Xiaojun Wu, Junwei Zhou, Xiaokun Chen, Hanwen Xing, Yubo Li, Qunhong Zeng, Di Wang, Yuanli Wang, Roey Ben Chaim, Penghao Jiang, Haotian Shen, Luyang Kong, Xinyi Liu, Runhui Wang, Xuanqing Liu, Jiachen Li, Xin Lan, Yueqian Lin, Wengao Ye, Junwei He, Songlin Li, Yue Zhang, Yipeng Gao, Yijiang Li, Ze Ma, Liqiang Jing, Tianyu Wang, Kaixin Li, Yiqi Xue, Haoran Lyu, Yizhuo He, Yuchen Tian, Shutong Wu, Bowei Wang, Yixuan Gao, Bo Chen, Litong Liu, Sikai Cheng, Jiajun Bao, Shuaicheng Tong, Shuwen Xu, Terry Yue Zhuo, Tinghan Ye, Qi Qi, Miao Li, Longtai Liao, Zelin Tan, Chang Shi, Xilin Tang, Srinath Tankasala, Boqin Yuan, Yaoyao Qian, Jianhong Tu, Chenguang Wang, Yizhou Sun, Wei Wang, Aaron Taylor, Ziyue Yang, Changkun Guan, Zhikang Dong, Xinyu Zhang, Steven Dillmann, Han chung Lee, and Dawn Song. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026.

Tianqi Luo, Chuhan Huang, Leixian Shen, Boyan Li, Shuyu Shen, Wei Zeng, Nan Tang, and Yuyu Luo. nvbench 2.0: Resolving ambiguity in text-to-visualization through stepwise reasoning. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025.

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

Paula Maddigan and Teo Susnjak. Chat2vis: Generating data visualisations via natural language using chatgpt, codex and gpt-3 large language models. *arXiv preprint arXiv:2302.02094*, 2023.

Mistral AI. Introducing Mistral Small 4, 2026. URL https://mistral.ai/news/mistral-small-4/. Accessed: 2026-09-26.

Moonshot AI. Kimi K2.6: Advancing Open-Source Coding, 2026. URL https://www.kimi.ai/blog/kimi-k2-6. Accessed: 2026-09-26.

T. Munzner. *Visualization Analysis and Design*. AK Peters Visualization Series. CRC Press, 2014. ISBN 9781466508934.

NVIDIA, Aakshita Chandiramani, Aaron Blakeman, Abdullahi Olaoye, Abhibha Gupta, Abhilash Somasamudramath, Abhinav Khattar, Adeola Adesoba, Adi Renduchintala, Adil Asif, Aditya Agrawal, Aditya Vavre, Ahmad Kiswani, Aishwarya Padmakumar, Ajay Hotchandani, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, Aleksandr Shaposhnikov, Alex Gronskiy, Alex Kondratenko, Alex Neefus, Alex Steiner, Alex Yang, Alexander Bukharin, Alexander Young, Ali Hatamizadeh, Ali Taghibakhshi, Alina Galiautdinova, Alisa Liu, Alok Kumar, Ameya Sunil Mahabaleshwarkar, Amir Klein, Amit Zuker, Amnon Geifman, Anahita Bhiwandiwalla, Ananth Subramaniam, Andrew Tao, Anjaney Shrivastava, Anjulie Agrusa, Ankur Srivastava, Ankur Verma, Ann Guan, Anna Shors, Annamalai Chockalingam, Anubhav Mandarwal, Aparnaa Ramani, Arham Mehta, Arti Jain, Arun Venkatesan, Asha Anoosheh, Ashwath Aithal, Ashwin Poojary, Asif Ahamed, Asit Mishra, Asli Sabanci Demiroz, Asma Kuriparambil Thekkumpate, Atefeh Sohrabizadeh, Avinash Kaur, Ayush Dattagupta, Barath Subramaniam Anandan, Bardiya Sadeghi, Barnaby Simkin, Ben Lanir, Benedikt Schifferer, Benjamin Chislett, Besmira Nushi, Bilal Kartal, Bill Thiede, Bita Darvish Rouhani, Bobby Chen, Boris Ginsburg, Brandon Norick, Branislav Kisacanin, Brian Yu, Bryan Catanzaro, Buvaneswari Mani, Carlo del Mundo, Chankyu Lee, Chanran Kim, Chantal Hwang, Chao Ni, Charles Wang, Charlie Truong, Cheng-Ping Hsieh, Chenhan Yu, Chenjie Luo, Cherie Wang, Chetan Mungekar, Chintan Patel, Chris Alexiuk, Chris Holguin, Chris Wing, Christian Munley, Christopher Parisien, Chuck Desai, Chunyang Sheng, Collin Neale, Cyril Meurillon, Dakshi Kumar, Dan Gil, Dan Su, Dane Corneil, Daniel Afrimi, Daniel Burkhardt Eliuth Triana, Daniel Egert, Daniel Fatade, Daniel Lo, Daniel Rohrer, Daniel Serebrenik, Daniil Sorokin, Daria Gitman, Daria Levy, Darko Stosic, David Edelsohn, David Messina, David Mosallanezhad, David Tamok, Deena Donia, Deepak Narayanan, Devin O'Kelly, Dheeraj Peri, Dhruv Nathawani, Di Wu, Dima Rekesh, Dina Yared, Divyanshu Kakwani, Dmitry Konyagin Brandon Tuttle, Dong Ahn, Dongfu Jiang, Dorrin Poorkay, Douglas O'Flaherty, Duncan Riach, Dusan Stosic, Dustin Van Stee, Edgar Minasyan, Edward Lin, Eileen Peters Long, Elad Segal, Elena Lantz, Elena Lewis, Ellie Evans, Elliott Ning, Eric Chung, Eric Harper, Eric Pham-Hung, Eric W. Tramel, Erick Galinkin, Erik Pounds, Esti Etrog, Evan Briones, Evan Wu, Evelina Bakhturina, Evgeny Tsykunov, Ewa Dobrowolska, Farshad Saberi Movahed, Farzan Memarian, Fay Wang, Fei Jia, Felipe Soares, Felipe Vieira Frujeri, Feng Chen, Fengguang Lin, Ferenc Galko, Fortuna Zhang, Frankie Siino, Frida Hou, Gantavya Bhatt, Gargi Prasad, Geethapriya Venkataramani, Geetika Gupta, George Armstrong, Gerald Shen, Giulio Borghesi, Gordana Neskovic, Gorkem Batmaz, Grace Lam, Grace Wu, Greg Pauloski, Greyson Davis, Grigor Nalbandyan, Guoming Zhang, Guy Farber, Guyue Huang, Haifeng Qian, Haran Kumar Shiv Kumar, Harry Kim, Harsh Sharma, Hayate Iso, Hayley Ross, Herbert Hum, Herman Sahota, Hexin Wang, Himanshu Soni, Hiren Upadhyay, Huy Nguyen, Iain Cunningham, Ido Galil, Ido Shahaf, Igino Padovani, Igor Gitman, Igor Shovkun, Ikroop Dhillon, Ilya Loshchilov, Ingrid Kelly, Itamar Schen, Itay Levy, Ivan Moshkov, Izik Golan, Izzy Putterman, Jain Tu, Jan Baczek, Jan Kautz, Jane Polak Scowcroft, Janica Rosenberg, Jared Casper, Jarrod Pflum, Jason Grant, Jason Sewall, Jatin Mitra, Jeffrey Glick, Jenny Chen, Jesse Oliver, Jiacheng Xu, Jiafan Zhu, Jialin Song, Jian Zhang, Jiaqi Zeng, Jie Lou, Jill Milton, Jim Chow, Jimmy Zhang, Jinhang Choi, Jining Huang, Jocelyn Huang, Joel Caruso, Joey Conway, Joey Guman, Johan Jatko, John Kamalu, Johnny Greco, Jonathan Cohen, Jonathan Raiman, Joseph Jennings, Joyjit Daw, Juan Yu, Julio Tapia, Junkeun Yi, Jupinder Parmar, Jyothi Achar, Kari Briski, Kartik Mattoo, Katherine Cheung, Katherine Luna, Keith Wyss, Kevin Shih, Kezhi Kong, Khanh Nguyen, Khushi Bhardwaj, Kirill Buryak, Kirthi Shankar Sivamani, Konstantinos Krommydas, Kris Murphy, Krishna C. Puvvada, Krzysztof Pawelec, Kumar Anik, Laikh Tewari, Laya Sleiman, Leo Du, Leon Derczynski, Li Ding, Lilach Ilan, Lingjie Wu, Lizzie Wei, Luis Vega, Lun Su, Maarten Van Segbroeck, Maer Rodrigues de Melo, Magaret Zhang, Mahan Fathi, Makesh Narsimhan Sreedhar, Makesh Sreedhar, Makesh Tarun Chandran, Manuel Reyes Gomez, Maor Ashkenazi, Marc Cuevas, Marc Romeijn, Margaret Zhang, Mark Cai, Mark Gabel, Markus Kliegl, Martyna Patelka, Maryam

Moosaei, Matthew Varacalli, Matvei Novikov, Mauricio Ferrato, Mehrzad Samadi, Melissa Corpuz, Meng Xin, Mengdi Wang, Mengru Wang, Meredith Price, Micah Schaffer, Michael Andersch, Michael Boone, Michael Evans, Michael Z Wang, Miguel Martinez, Mikail Khona, Mike Chrzanowski, Mike Hollinger, Mingyuan Ma, Minseok Lee, Mohammad Dabbah, Mohammad Shoeybi, Mostofa Patwary, Nabin Mulepati, Nader Khalil, Najeeb Nabwani, Nancy Agarwal, Nanthini Balasubramaniam, Narimane Hennouni, Narsi Kodukula, Natalie Hereth, Nathaniel Pinckney, Nave Assaf, Negar Habibi, Nestor Qin, Neta Zmora, Netanel Haber, Nick Reamaroon, Nickson Quak, Nidhi Bhatia, Nikhil Jukar, Nikki Pope, Nikolai Ludwig, Nima Tajbakhsh, Nir Ailon, Nirmal Juluru, Nirmalya De, Nowel Pitt, Oleg Rybakov, Oleksii Hrinchuk, Oleksii Kuchaiev, Olivier Delalleau, Oluwatobi Olabiyi, Omer Ullman Argov, Omri Almog, Omri Puny, Oren Tropp, Otavio Padovani, Ouye Xie, Parth Chadha, Pasha Shamis, Paul Gibbons, Pavlo Molchanov, Peter Belcak, Peter Jin, Pinky Xu, Piotr Januszewski, Pooya Jannaty, Prachi Shevate, Pradeep Thalasta, Pranav Prashant Thombre, Prasoon Varshney, Prerana Gambhir, Pritam Gundecha, Przemek Tredak, Qing Miao, Qiyu Wan, Quan Tran Minh, Rabeeh Karimi Mahabadi, Rachel Oberman, Rachit Garg, Rahul Kandu, Raina Zhong, Ran El-Yaniv, Ran Zilberstein, Rasoul Shafipour, Renee Yao, Renjie Pi, Richard Mazzarese, Richard Wang, Rick Izzo, Ridhima Singla, Rima Shahbazyan, Rishabh Garg, Ritika Borkar, Ritu Gala, Riyad Islam, Robert Clark, Robert Hesse, Roger Waleffe, Rohit Varma Kalidindi, Rohit Watve, Roi Koren, Ron Fan, Ruchika Kharwar, Ruisi Cai, Ruoxi Zhang, Russell J. Hewett, Ryan Prenger, Ryan Timbrook, Ryota Egashira, Sadegh Mahdavi, Sagar Singh Ashutosh Joshi, Sahil Modi, Samuel Kriman, Sandeep Pombra, Sanjay Kariyappa, Sanjeev Satheesh, Santiago Pombo, Saori Kaji, Satish Pasumarthi, Saurav Mishra, Saurav Muralidharan, Scott Hara, Sean Narenthiran, Sebastian Rogawski, Seonjin Na, Seonmyeong Bak, Sepehr Sameni, Seth Poulos, Shahar Mor, Shantanu Acharya, Shaona Ghosh Adam Lord, Sharath Turuvekere Sreenivas, Shaun Kotek, Shaya Gharghabi, Shelby Thomas, Sheng-Chieh Lin, Shibani Likhite, Shiqing Fan, Shiyang Chen, Shreya Gopal, Shrimai Prabhumoye, Shubham Pachori, Shubham Toshniwal, Shuo Zhang, Shuoyang Ding, Shyam Renjith, Shyamala Prayaga, Siddhartha Jain, Simeng Sun, Sirisha Rella, Sirshak Das, Smita Ithape, Sneha Harishchandra S, Somshubra Majumdar, Soumye Singhal, Sri Harsha Singudasu, Sriharsha Niverty, Stas Sergienko, Stefana Gloginic, Stefania Alborghetti, Stephen Ge, Stephen McCullough, Sugam Dipak Devare, Suguna Varshini Velury, Sukrit Rao, Sumeet Kumar Barua, Sunny Gai, Suseella Panguluri, Sushil Koundinyan, Swathi Patnam, Sweta Priyadarshi, Swetha Bhendigeri, Syeda Nahida Akter, Sylendran Arunagiri, Tailling Yuan, Talor Abramovich, Tan Bui, Tan Yu, Terry Kong, Thanh Do, Thomas Gburek, Thorgane Marques, Tiffany Moore, Tijmen Blankevoort, Tim Moon, Timothy Ma, Tiyasa Mitra, Tomasz Grzegorzek, Tomer Asida, Tomer Bar Natan, Tomer Keren, Tomer Ronen, Traian Rebedea, Trenton Starkey, Tugrul Konuk, Twinkle Vashishth, Tyler Condensa, Udi Karpas, Ushnish De, Vahid Noorozi, Vahid Noroozi, Vanshil Atul Shah, Veena Vaidyanathan, Venkat Srinivasan, Venmugil Elango, Victor Cui, Vijay Korthikanti, Vikas Mehta, Virginia Adams, Virginia Wu, Vitaly Kurin, Vitaly Lavrukhin, Vladimir Anisimov, Wan Seo, Wanli Jiang, Wasi Uddin Ahmad, Wei Du, Wei Ping, Wei-Ming Chen, Wendy Quan, Wenliang Dai, Wenwen Gao, Will Jennings, William Zhang, Xiaowei Ren, Xiaowen Xin, Xin Li, Yang Yu, Yangyi Chen, Yaniv Galron, Yashaswi Karnati, Yejin Choi, Yev Meyer, Yi-Fu Wu, Yian Zhang, Ying Lin, Yonatan Geifman, Yonggan Fu, Yoshi Suhara, Youngeun Kwon, Yuan Zhang, Yuki Huang, Zach Moshe, Zhilin Wang, Zhiyu Cheng, Zhongbo Zhu, Zhuolin Yang, Zihan Liu, Zijia Chen, Zijie Yan, and Zuhair Ahmed. Nemotron 3 super: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. *arXiv preprint arXiv:2604.12374*, 2026.

OpenAI. GPT-5.6 system card, 2026. URL `https://deploymentsafety.openai.com/gpt-5-6`. Accessed: 2026-09-26.

OpenAI, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, Kai Chen, Mark Chen, Enoch Cheung, Aidan Clark, Dan Cook, Marat Dukhan, Casey Dvorak, Kevin Fives, Vlad Fomenko, Timur Garipov, Kristian Georgiev, Mia Glaese, Tarun Gogineni, Adam Goucher, Lukas Gross, Katia Gil Guzman, John Hallman, Jackie Hehir, Johannes Heidecke, Alec Helyar, Haitang Hu, Romain Huet, Jacob Huh, Saachi Jain, Zach Johnson, Chris Koch, Irina Kofman, Dominik Kundel, Jason Kwon, Volodymyr Kyrylov, Elaine Ya Le, Guillaume Leclerc, James Park Lennon, Scott Lessans, Mario Lezcano-Casado, Yuanzhi Li, Zhuohan Li, Ji Lin,

Jordan Liss, Lily, Liu, Jiancheng Liu, Kevin Lu, Chris Lu, Zoran Martinovic, Lindsay McCallum, Josh McGrath, Scott McKinney, Aidan McLaughlin, Song Mei, Steve Mostovoy, Tong Mu, Gideon Myles, Alexander Neitz, Alex Nichol, Jakub Pachocki, Alex Paino, Dana Palmie, Ashley Pantuliano, Giambattista Parascandolo, Jongsoo Park, Leher Pathak, Carolina Paz, Ludovic Peran, Dmitry Pimenov, Michelle Pokrass, Elizabeth Proehl, Huida Qiu, Gaby Raila, Filippo Raso, Hongyu Ren, Kimmy Richardson, David Robinson, Bob Rotsted, Hadi Salman, Suvansh Sanjeev, Max Schwarzer, D. Sculley, Harshit Sikchi, Kendal Simon, Karan Singhal, Yang Song, Dane Stuckey, Zhiqing Sun, Philippe Tillet, Sam Toizer, Foivos Tsimpourlas, Nikhil Vyas, Eric Wallace, Xin Wang, Miles Wang, Olivia Watkins, Kevin Weil, Amy Wendling, Kevin Whinnery, Cedric Whitney, Hannah Wong, Lin Yang, Yu Yang, Michihiro Yasunaga, Kristen Ying, Wojciech Zaremba, Wenting Zhan, Cyril Zhang, Brian Zhang, Eddie Zhang, and Shengjia Zhao. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Qwen. Qwen3.8-27B, 2026. URL https://huggingface.co/Qwen/Qwen3.8-27B. Accessed: 2026-09-25.

Qwen Team. Qwen3.5: towards native multimodal agents, 2026a. URL https://qwen.ai/blog?id=qwen3.5. Accessed: 2026-09-26.

Qwen Team. Qwen3.6-27B: flagship-level coding in a 27b dense model, 2026b. URL https://qwen.ai/blog?id=qwen3.6-27b. Accessed: 2026-09-26.

Mizanur Rahman, Md. Tahmid Rahman Laskar, Shafiq Joty, and Enamul Hoque. 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, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 31849–31874, 2025.

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

Chengyue Wu, Zhixuan Liang, Yixiao Ge, Qiushan Guo, Zeyu Lu, Jiahao Wang, Ying Shan, and Ping Luo. Plot2code: A comprehensive benchmark for evaluating multi-modal large language models in code generation from scientific plots. In *Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, volume NAACL 2025 of *Findings of ACL*, pp. 3006–3028, 2025.

Zhiyu Yang, Zihan Zhou, Shuo Wang, Xin Cong, Xu Han, Yukun Yan, Zhenghao Liu, Zhixing Tan, Pengyuan Liu, Dong Yu, Zhiyuan Liu, Xiaodong Shi, and Maosong Sun. Matplotagent: Method and evaluation for llm-based agentic scientific data visualization. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of *Findings of ACL*, pp. 11789–11804, 2024.

Fatemeh Pesaran Zadeh, Juyeon Kim, Jin-Hwa Kim, and Gunhee Kim. Text2chart31: Instruction tuning for chart generation with automatic feedback. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 11459–11480, 2024.

Z.ai. GLM-5.3: Frontier Coding with Emergent Cyber Capabilities, 2026. URL https://z.ai/blog/glm-5.3. Accessed: 2026-09-26.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

A BENCHMARK DETAILS

A.1 BENCHMARK CONSTRUCTION DETAILS

Document and table filtering. We construct the benchmark from documents in FinePDFs, retaining the complete extracted document rather than an isolated table so that locating the relevant evidence remains part of the task. We extract Markdown tables together with their captions and surrounding text and apply deterministic structural filters before invoking an LLM. We retain documents containing between 3,000 and 100,000 characters and tables containing at least three rows, three columns, four numeric cells, and a numeric-cell ratio of at least 0.2. Tables longer than 8,000 characters and duplicate tables with identical normalized content are removed.

Analytical-type matching. For each retained table, we first compute structural signals such as the number of numeric columns, candidate entities, and temporal observations. These signals are used only to eliminate analytical types whose necessary preconditions are definitively violated; missing or uncertain signals do not cause a type to be removed. An LLM then evaluates each remaining table–type pair using the table, its caption, and its surrounding context. The model determines whether the table supports the analytical type and provides evidence for its decision. When a table supports multiple types, we assign it to an underrepresented type to maintain a balanced candidate pool. We use at most one analytical type per table and limit the contribution of any single document. Table 6 provides the full preconditions, intended transformations, and canonical and acceptable chart forms for all ten analytical types.

Explicit task and rubric generation. For every selected table–type pair, a separate LLM call constructs a fully specified internal task. This task identifies the relevant rows, columns, entities, and periods; states the required value transformation; records the intended answer or pattern; and specifies chart forms capable of revealing that answer. The same call produces sample-specific anchors for data selection, value transformation, answer visibility, and chart form. The chart-form anchors distinguish canonical, acceptable, and inappropriate alternatives so that evaluation does not require every system to reproduce a single reference visualization. The internal task and rubric are used only during benchmark construction and evaluation and are never shown to the chart-generation system.

Implicit-request generation. Another LLM rewrites the explicit task as a natural user request. The rewrite must preserve the analytical goal and intended answer while omitting the construction procedure, including exact column lists, transformation formulas, chart-form names, and low-level presentation instructions. Automated validation rejects malformed or overly vague requests and checks for leakage of chart-form terminology, formulas, and presentation instructions. For requests with a discrete answer, we additionally require an outcome-oriented cue, such as asking that the answer be easy to identify, without prescribing how it should be visually emphasized.

Human review and final selection. Human reviewers inspect each remaining request together with the target table, surrounding context, fully specified internal task, and sample-specific rubric. A candidate is retained only when the request is natural, answerable from the source document, sufficiently implicit, uniquely grounded in the target evidence, and consistent with the intended transformation, answer, and chart-form rubric. Candidates with factual inconsistencies, leaked construction instructions, ambiguous scope, incomplete rubrics, or implausible analytical goals are rejected. The final benchmark contains 1,000 retained examples, with 100 examples for each analytical type, drawn from 1,000 distinct tables in 960 documents.

A.2 ANALYTICAL TASK-TYPE SPECIFICATIONS

The benchmark covers ten analytical task types that require decisions beyond directly plotting recorded values. Each type specifies the data conditions under which the task is meaningful, the value transformation or analytical operation needed to answer the request, and the chart forms that can effectively communicate the result. These specifications guide both candidate construction and sample-specific rubric generation. The stated support conditions are necessary but not sufficient, because structural properties alone do not establish whether a transformation or comparison is semantically meaningful. An LLM therefore additionally examines the table and its surrounding context when determining whether a table supports a particular task type. Table 6 summarizes the support conditions, required analytical operations, and chart-form tiers for all ten types.

Table 6: Specifications for the ten analytical task types. Canonical and acceptable forms correspond to the highest and intermediate chart-form rubric tiers, respectively. The sample-specific rubrics instantiate these specifications using the entities, values, and analytical goal of each example.

Task type	Support conditions	Required analytical operation	Chart-form tiers
`growth_speed`	At least two entities observed over at least two periods, with comparable raw levels in the same unit.	Compare rates of change using percentage growth, CAGR, slope, or a baseline-relative trajectory.	Canonical: bars, lollipops, or dots encoding the growth measure. Acceptable: indexed trajectory lines or a first-to-last-period slope chart.
`momentum_turning`	A primary numeric series observed over at least five consecutive, evenly spaced periods, with an identifiable turn or extremum.	Compute period-over-period change or growth to determine when momentum slows, accelerates, peaks, or reverses.	Canonical: bars or lines of period-over-period change with a zero reference line. Acceptable: a raw-level chart augmented with explicit per-period changes.
`baseline_index`	Multiple comparable series with different starting levels and a meaningful baseline period, condition, or reference entity. The values must not already be normalized.	Express each series relative to its baseline, either as an index or as percentage change from the baseline.	Canonical: baseline-relative lines with a reference line at the base level. Acceptable: aligned small multiples or baseline-relative bars when the number of periods is small.
`paired_gap`	At least four entities with two comparable, same-unit measurements per entity.	Compute the direction and magnitude of the difference between the paired measurements for each entity.	Canonical: diverging bars or lollipops of the difference around zero. Acceptable: dumbbell charts, slope charts, or two-measure scatterplots with an identity line.
`composition_snapshot`	At least three non-overlapping components that form a meaningful whole in one period or context. The component sum must agree with the stated total within approximately 5%.	Compute each component's share of the total while excluding any total or subtotal from the component set.	Canonical: a readable pie or donut chart, or a single 100% stacked bar. Acceptable: a treemap, waffle chart, or free-standing bars of component shares.
`composition_compare`	At least two groups containing the same set of at least three additive components, with components summing to each group's total.	Compute component shares separately within each group.	Canonical: 100% stacked bars with consistent component ordering and colors. Acceptable: aligned pies or donuts, mirrored 100% bars, grouped share bars, or share-based slope charts.
`mix_trend`	The same additive components observed over at least three ordered periods, with components summing to each period's total.	Compute component shares within each period to reveal how the mix changes over time.	Canonical: a 100% stacked area chart, or 100% stacked bars when periods are few. Acceptable: component-share lines or aligned small multiples of share trajectories.
`concentration_long_tail`	At least six entities with additive values for which contribution to a total is meaningful.	Sort contributions in descending order and compute their cumulative share of the total.	Canonical: a Pareto chart combining descending bars with a cumulative-share line. Acceptable: descending bars alone or a cumulative-share curve.
`tradeoff`	At least four entities measured on two mathematically distinct quantities that a reader would meaningfully weigh against each other.	Represent both quantities jointly for every entity without collapsing them into an arbitrary combined score.	Canonical: a scatterplot with one quantity on each axis. Acceptable: a bubble chart with a meaningful third quantity or a normalized parallel-axes or slope representation.
`stability_volatility`	At least two entities with at least five raw observations each. The table must not already report the target dispersion statistic.	Derive a defensible measure or view of dispersion, such as standard deviation, coefficient of variation, range, or interquartile range.	Canonical: entity-level boxplots or violin plots, or coefficient-of-variation bars when means differ substantially. Acceptable: standard-deviation, range, or IQR bars; means with error bars; or entity-level strip plots.

Chart-form tiers. The chart-form tiers describe how directly a visualization supports the intended analytical task. Canonical forms receive the highest chart-form score because they directly encode the required comparison or derived quantity. Acceptable alternatives remain analytically valid but communicate the target relationship less directly. Charts that display only raw values, omit the required transformation, or otherwise fail to expose the requested relationship receive the lowest chart-form score. These specifications define type-level constraints, while each sample-specific rubric instantiates them using the entities, values, and analytical goal of the corresponding example.

Discrete- and pattern-answer tasks. The task types also differ in whether the request seeks a discrete answer or a broader visual pattern. For `growth_speed`, `momentum_turning`, `baseline_index`, `paired_gap`, and `stability_volatility`, the request asks the reader to identify a particular entity or period. Their rubrics therefore include an answer-visibility criterion that assesses whether the target is readily identifiable. The implicit request expresses this outcome as a user intent, such as wanting the answer to be apparent at a glance, without prescribing a particular highlighting or annotation technique.

Table 7: Dataset statistics. Document length and tables per document are computed over 960 unique documents; all other rows are computed over 1,000 examples. Document length is measured in whitespace-delimited words. Tables per document counts every markdown table in the source document, and target-table ordinal is the position of the target table among them. Relative position is the starting character offset of the target table divided by document length.

Property	Mean	Median	Range
Document length	5,531	4,738	345–18,791
Tables per document	9.1	5	1–80
Table rows	10.5	8	3–53
Table columns	5.6	5	3–23
Target-table ordinal	4.9	3	1–78
Relative position (%)	52.2	54.4	0.0–99.2

The remaining five types ask for an overall structure or relationship rather than a single entity or period. We omit the answer-visibility criterion for these pattern-answer tasks because the visual pattern communicated by the chart is itself the answer. For these tasks, task fulfillment is therefore assessed through value transformation and chart form, without a separate answer-visibility criterion.

A.3 Detailed Document Statistics

Benchmark coverage. The benchmark contains 1,000 examples constructed from 1,000 distinct target tables in 960 source documents. No target table is used in more than one example. Among the source documents, 920 contribute one example and 40 contribute two examples based on different target tables. The benchmark contains exactly 100 examples for each of the ten analytical task types. Table 7 summarizes the lengths and table counts of the source documents, as well as the dimensions and locations of the target tables.

Document complexity. Document-level statistics are computed over the 960 unique source documents, whereas target-table statistics are computed over all 1,000 examples. Document length is measured in whitespace-delimited words, and the number of tables includes every Markdown table extracted from the complete source document rather than only those retained as benchmark candidates. The documents contain between 1 and 80 tables, with a median of 5, requiring systems to distinguish the relevant evidence from multiple plausible sources. The ordinal position of the target table ranges from the first to the 78th table in a document.

Target-table variation. Target tables range from 3 to 53 rows and from 3 to 23 columns, covering both compact summaries and comparatively large tables. We define the relative position of a target table as the character offset at which the table begins divided by the total character length of the document. Target-table positions span nearly the entire document, from 0.0% to 99.2%, with a mean of 52.2% and a median of 54.4%. This distribution limits the usefulness of positional shortcuts and requires models to identify the table that semantically supports the request.

B Harness Details

B.1 Catalogue Construction

Construction independence. We construct the visualization catalogue independently of ID2C and its analytical task types. Neither the benchmark requests nor their rubrics are used to collect, filter, or describe catalogue entries. Instead, the construction process uses only general definitions of chart forms and derived quantities, allowing the resulting catalogue to serve as an external knowledge source rather than a benchmark-specific repository.

Candidate collection. We collect candidate concepts from English Wikipedia through category traversal, curated list pages, and a small set of named concepts that are not reliably reached through the category graph. For chart forms, the seed categories cover statistical graphics, charts, plots, financial charts, thematic maps, and data visualization. For value transformations, the seeds cover

summary statistics, ratios, measures of dispersion, index numbers, economic and financial indicators, rates, concentration measures, and time-series quantities. We traverse each category only to a predefined depth of zero to two levels.

This process produces 3,273 unique candidate articles: 876 chart-form candidates and 2,397 value-transformation candidates. We apply deterministic filters based on article metadata to remove pages that are clearly people, organizations, software, creative works, places, non-article pages, or articles without sufficient descriptive text. After this filtering, 765 chart-form candidates and 2,291 value-transformation candidates remain.

Semantic filtering. We use an LLM to classify each remaining candidate from its title, description, and opening text. A chart-form candidate is retained only if it describes a data graphic in which quantitative or categorical values are mapped to visual marks or channels such as position, length, area, angle, or color. We exclude structural and process diagrams, general visualization concepts, entities, symbols, map projections, and drawing techniques. A value-transformation candidate is retained only if it describes a derived quantity computed from recorded values to summarize, compare, or track them. We exclude directly measured physical quantities, statistical tests and models, probability distributions, units, general concepts, and publisher-specific indices whose definitions do not provide a reusable transformation. This stage retains 199 chart forms and 744 value transformations.

Card generation. For each retained concept, we retrieve the corresponding Wikipedia article and provide it to a separate LLM as background. The model writes a compact card in its own words rather than extracting article sentences. Every card contains a stable identifier, the concept name, a one-sentence definition, one to three situations describing when the concept is useful, and a link to its source article. A chart-form card additionally specifies how data are mapped to axes, marks, and other visual channels. A value-transformation card instead provides a formula expressed over descriptively named inputs. The usage situations are written in general analytical terms without dataset-specific entities, column names, or example requests so that they can be matched against new tasks.

B.2 Card Representation and Retrieval

Card representation. The catalogue stores chart forms and value transformations as two separate collections. Both card types contain a stable identifier, a name, a one-sentence definition, and one to three `use_when` statements describing the analytical situations in which the concept is useful. A chart-form card additionally contains an `encoding` field that specifies how variables are mapped to axes, marks, ordering, color, or other visual channels. A value-transformation card instead contains a `formula` field that expresses the computation using descriptively named inputs.

Retrieval representation. We construct the retrieval text for a card by concatenating its name, definition, and `use_when` statements. We exclude the `encoding` and `formula` fields from this representation because retrieval should match the analytical need expressed by the query rather than specific implementation details. These fields are nevertheless included in the full card returned after retrieval, allowing the *form agent* to use the encoding or formula when constructing its plan.

We embed each retrieval text using `BAAI/bge-m3` (Chen et al., 2025a). The embedding is taken from the model's CLS token and L2-normalized.

Query formulation. In the first model call, the *form agent* produces one retrieval query for the required value transformation and another for the chart form. The agent is instructed to express the general analytical need behind each decision rather than repeat document-specific details, as specified in the retrieval-query prompt reproduced in Appendix **??**.

Dense retrieval. Each query is embedded using the same model as its corresponding card collection. Because both query and card vectors are L2-normalized, we rank cards using their vector dot product, which is equivalent to cosine similarity. For a query q and card c, the retrieval score is

$$s(q, c) = \mathbf{e}(q)^\top \mathbf{e}(c), \tag{2}$$

where $\mathbf{e}(\cdot)$ denotes the normalized embedding. We return the five highest-scoring cards from each catalogue.

Use of retrieved cards. The retrieved cards are presented to the *form agent* in a second model call. Each returned entry includes its name, definition, usage situations, and either its encoding

specification or formula. The agent then determines the value transformation and chart form for the current request. The cards are advisory rather than prescriptive, so the agent may combine their guidance with the grounded data or disregard them when none is appropriate.

B.3 MULTI-AGENT PIPELINE

Pipeline structure. The generation pipeline decomposes document-to-chart generation into data grounding, chart-form planning, aesthetic design, code generation, and visual critique. Each stage receives the user request together with a structured output from the preceding stages. The intermediate representations make the main chart-construction decisions explicit before code is generated.

Grounding agent. The grounding agent receives the complete source document D and user request q. It identifies the table or tables relevant to the request and copies the required values without computing derived quantities, rounding values, or converting units. Its output is a structured data packet containing the selected rows and columns, the source tables from which they were taken, contextual notes needed to interpret the values, and a brief selection rationale. The contextual notes preserve information such as units, scale factors, and qualifications that may appear outside the table cells. A request may draw evidence from multiple tables when all of them are necessary to answer it.

Under the normal pipeline, the grounding agent is the only component that reads the complete document. Subsequent agents receive the smaller data packet instead, reducing context length while preserving the provenance and interpretation of the selected values.

Form agent. The form agent receives the user request and grounded data packet. It makes two separate decisions corresponding to the two planning decisions targeted by the catalogue. The *derived quantity* specifies the value transformation to apply, including a verbal description of what should be plotted and a formula over the columns in the data packet. The *encoding* specifies the chart form, including the type of chart and how the transformed or original values should map to its marks and visual channels. The agent describes the value transformation but does not calculate the transformed values, leaving the arithmetic to the generated code. In the retrieval-augmented pipeline, the form agent uses the two-call catalogue procedure described in Appendix B.2 before finalizing the value transformation and chart form.

Aesthetic agent. The aesthetic agent receives the request, grounded data, and completed form specification. It determines the visual treatment of the chart, including layout, color, text placement, and the arrangement of marks. Its output is a visual-design specification and a short rationale that are passed to the coding agent.

Coding agent. The coding agent receives the user request, grounded data packet, form specification, and aesthetic specification. It writes self-contained Python code using the supplied values and computes the planned value transformation in code. The agent is instructed to preserve the values copied by the grounding agent rather than replacing them with manually calculated results. The generated program may use `pandas`, `numpy`, `matplotlib`, `math`, and `datetime`.

Visual critique and revision. After the code is executed, the visual critic receives the user request, generated code, and rendered chart. The critic determines whether the rendered chart serves the request and returns a satisfaction decision together with concrete visual problems and corresponding fixes. If the code fails to execute or does not produce an image, the execution error is returned directly to the coding agent without invoking the visual critic.

When the critic identifies a problem, the coding agent receives the critique and produces a complete revised program. This render–critique–revision cycle continues until the critic is satisfied or the revision budget is exhausted. If a later revision fails to render, the pipeline retains the most recent successfully rendered chart rather than replacing it with the failed revision.

Prompt for Chart Generation

````
<System Message>
You will be given a user query and a source document.
Write Python code that answers the query with a chart.

Respond in exactly this format:

<chart_plan>
A short description of the chart you will make.
</chart_plan>
```python
# complete executable Python code
```

The code must be self-contained: use only pandas, numpy, matplotlib, math, or datetime;
hard-code any values you need from the document (no file or network access);
save the chart with plt.savefig("chart.png").

<User Message>
user_query:
{implicit query}

source_document:
{full document text}
````

Figure 4: The chart-generation prompt. Every model in Table 2 receives this byte-identical prompt; red tokens mark per-example substitutions.

C Evaluation Details

C.1 Model and Inference Settings

All models are evaluated with temperature 0, reasoning disabled, and a maximum output length of 4,096 tokens, except for two models that do not support all of these settings. Claude Opus 5 runs with its vendor defaults (adaptive thinking, *high* effort, and a 16,000-token budget), as its API does not accept a temperature setting when thinking is enabled. GLM-5.3 does not support disabling reasoning and therefore runs at its default reasoning effort, with temperature 0 and a 49,152-token output budget.

C.2 Judge Implementation

The judge is run with one inference call per criterion (seven calls, six for pattern-answer tasks), greedy decoding, and thinking disabled, making the judge highly reproducible and free of API cost. Re-running the identical protocol reproduces per-record verdicts at 96.7–100% agreement across criteria.

C.3 Human-Alignment Study

We randomly sample 106 instances from ID2C and use the corresponding outputs generated by Qwen3.6-27B and Claude Opus 5 to construct pairwise comparisons. For each instance, human raters are shown the source document, the request, and the two rendered charts. Model identities are hidden, and the display order of the charts is randomized. Raters select A, B, or Tie for data correctness, task fulfillment, readability, and overall preference.

Two independent groups of human raters evaluate the same pairs. This allows human–human agreement to serve as a reference when interpreting agreement between the judge and human raters.

Each judge variant independently scores both charts in every pair. For each evaluation dimension and the overall score, we convert the score difference between the two charts into an A, B, or Tie verdict and compare it with the corresponding human judgment using linear-weighted Cohen's κ. This pairwise formulation avoids requiring human raters to calibrate absolute scores across examples.

D Prompts for Generation and Evaluation

Prompt for Per-Criterion Chart Evaluation

```
<User Message>
You are grading ONE criterion of a chart-generation attempt.

You are given ONE image: a chart produced as a response to the user query below.
That chart, as shown in the image, is what you are evaluating.

The source_table below is the oracle table: it holds the content the chart should
ideally represent. The candidate_code below is the code that produced the chart;
use it only as an auxiliary reference — always judge from the chart in the image,
never from the code alone.

You will be given a single criterion after the image. That criterion is the only
standard: grade against what it requires, not against a chart you would have drawn
yourself, and never invent requirements it does not state. Other aspects of the
chart are graded separately under other criteria — do not let them influence
this score.

If the criterion does not specify its own scale, follow this 3-point scale:
  3 = the requirement is fully satisfied;
  2 = a defensible but suboptimal realization — below the standard the criterion
      describes, yet not wrong;
  1 = wrong or missing.

user_query:
{implicit query}

source_table:
{source table markdown}

candidate_code:
{candidate code}

The chart to evaluate:
{candidate chart image}

criterion to grade — {criterion name}:
{criterion requirement}

Return valid JSON only, with exactly this shape:
{"criterion": "{criterion name}", "score": 1 | 2 | 3,
 "reason": "one concise sentence citing what you saw"}
```

For data selection, value fidelity, and label-encoding fidelity, the third paragraph is replaced by:

```
The source_table below is the oracle table: it holds the content the chart should
ideally represent. The candidate_code below is the code that produced the chart.
For this criterion, the candidate CODE and the SOURCE TABLE decide which rows,
columns and values were used and whether they are right — check what the code
hard-codes against the table, and count. Then confirm in the IMAGE that what
the code computes is what the chart actually shows.
```

Figure 5: The judge's per-criterion evaluation call. Each criterion is graded in its own call; for the three data-correctness criteria, the third paragraph is replaced as shown in the lower box.

Global Criterion Anchors for Chart Evaluation

Value fidelity:

```
Compare the numbers the candidate hard-coded and plotted against the SOURCE TABLE.
  3 = every plotted value traces to the table (or a correct derivation of it)
      within rounding;
  2 = minor slips that do not change the answer;
  1 = fabricated, misread, or unit-mangled values, or values taken from the
      wrong rows.
Wrong math on correctly-copied values belongs to analysis, not here.
```

Label-encoding fidelity:

```
Compare every textual claim the chart makes (title, axis labels, legend entries,
units, annotations) against what is actually plotted and against the SOURCE TABLE.
A label must truthfully describe the quantity, the transformation, the unit/scale,
and the scope of the encoded data — and a strong chart labels its data so the
reader can check it against the table.
  3 = the plotted values are labeled on or beside the data in the table's own
      units, and every textual claim is accurate on all four dimensions;
  2 = the claims are accurate but the labeling is thin (values unlabeled,
      a generic title, a period or scope left implicit, values transformed away
      from the table's numbers), or exactly one peripheral annotation fails the
      check while the chart's main reading and its labeled values remain correct;
  1 = the chart's central text misleads: a transformation named but not applied
      (title says "Growth Rate (%)" over absolute deltas), a unit or scale
      contradicting the plotted magnitudes (axis says "$K" over raw dollars),
      a period/scope claim contradicting the rows actually used (title says "Q2"
      over H1 data), legend entries that do not correspond to the plotted series,
      or multiple labeled values that do not match the table.
Score labels against the data AS PLOTTED, not against the query: whether the
transformation was the one the query demanded belongs to analysis; whether the
plotted numbers are correct belongs to value fidelity.
```

Presentation:

```
Judge, from the IMAGE, how well the chart reads as a whole — not only whether
anything is broken, but how much deliberate design effort is visible.
Text deliberately placed over the data (value labels on bars, an annotation
beside a point) is fine as long as it reads clearly.
  3 = visibly designed, not merely correct: award it only when the chart goes
      beyond a default rendering in at least TWO concrete ways you can point at —
      the key values are labeled on the data itself; the ordering is deliberately
      chosen (e.g. sorted by value) rather than arbitrary; colors carry meaning
      or emphasis rather than an unmodified default cycle; the title states the
      finding rather than just the topic; a total or summary figure is annotated
      on the chart. One small blemish at the periphery (a legend touching one
      label) does not disqualify an otherwise designed chart;
  2 = the default score for a competent chart: correct, readable, nothing
      broken — but a plain rendering with unlabeled values, arbitrary ordering,
      stock colors, and a merely descriptive title; also a designed chart with
      a moderate flaw that takes real effort to read past;
  1 = anything that looks broken or would mislead the reader: garbled or
      colliding text that cannot be read, a series drawn invisibly behind
      another, labels or annotations detached from the data they describe,
      label area rivaling the plot area, tangled or misordered lines, or an
      absurd aspect ratio that breaks the chart.
```

Figure 6: Global criterion anchors for value fidelity, label-encoding fidelity, and presentation. These three criteria are identical across records; the remaining criteria use the sample-specific rubrics shipped with each record.

Prompt for Ablation on Chart Evaluation: without criterion decomposition

```
[query fulfillment]
Judge, from the IMAGE, whether the chart does what the user query asked:
the quantity or comparison the query calls for is actually derived and plotted,
the chart form fits that analytical intent, and — when the query asks to spot
or compare something specific — the answer can be found at a glance.
  3 = the chart answers the query directly: the right quantity is plotted in
      a form that fits the intent, and the asked-for takeaway is visible;
  2 = the chart addresses the query but imperfectly — a defensible but
      suboptimal derivation, form, or emphasis that makes the answer harder
      to extract;
  1 = the chart does not answer the query: wrong or missing derivation, a form
      that hides the intended comparison, or the asked-for answer cannot
      be found.

[data fidelity]
Check the chart's data against the SOURCE TABLE, using the candidate CODE to
see which rows, columns and values were used.
  3 = the rows and columns the query concerns were selected, every plotted value
      traces to the table (or a correct derivation of it) within rounding,
      the plotted values are labeled on or beside the data in the table's own
      units, and the chart's text — title, axes, units, legend — truthfully
      describes what is plotted;
  2 = minor slips — a small selection or copying error, thin labeling (values
      unlabeled or transformed away from the table's numbers), or exactly one
      peripheral annotation that fails the check — that do not change the answer
      or mislead the reader;
  1 = wrong or fabricated values, rows that do not match the query's target,
      or central text that misleads about what is drawn: a transformation named
      but not applied, a unit contradicting the plotted magnitudes, or multiple
      labeled values that do not match the table.
```

Figure 7: Axis-level criteria used by the *w/o criterion decomposition* ablation, which grades the three human-evaluation axes directly instead of the seven criteria.

Global Criterion for Ablation on Chart Evaluation: without sample-specific rubrics

```
[data selection]
Check that the chart uses exactly the part of the oracle table the user query
concerns. Every row, column, entity, and period the query needs must be present;
unrelated rows or columns, and total/aggregate rows the query did not ask for,
must be excluded.
  3 = full coverage of what the query needs, nothing irrelevant mixed in;
  2 = small omissions or extras that do not change the answer;
  1 = major omissions, or the wrong rows or columns.

[analysis]
Decide what quantity the user query actually asks about — often a derived one
(a change or growth rate between periods, a share of a total or within a group,
values indexed to a baseline, a cumulative share, a gap between paired values,
a dispersion measure), sometimes the raw values themselves — and check that the
plotted values are that quantity, computed correctly from the oracle table.
Mathematically equivalent formulations count as correct.
  3 = the quantity the query calls for is computed and plotted correctly;
  2 = the right computation with a minor flaw (e.g., a stray total row included);
  1 = the computation the query calls for is missing (raw values where
      a derivation is needed) or wrong.

[chart form]
The chart form must fit the question: it should directly encode the comparison,
trend, composition, relationship, or ranking the query asks about — for example,
sorted bars for comparing one quantity across entities, lines over time for
trends or indexed trajectories, 100%-stacked forms for composition, a scatter
for a two-metric trade-off, diverging bars around zero for signed changes,
a cumulative line for concentration.
  3 = a form that directly encodes what the query asks, laid out in a standard,
      readable way;
  2 = a defensible alternative from which a reader can still recover the answer;
  1 = a form that obscures or misrepresents the asked-for structure.

[answer visibility]
The user query asks to see something specific at a glance — a largest or
fastest-changing item, a turning point, a dominant contributor, or the structure
of a breakdown. That answer must be visible in the chart itself.
  3 = the answer is explicitly marked (an emphasis color, an annotation,
      a reference line) or immediately readable from the chart;
  2 = the answer can be identified without explicit marking (e.g., through
      sorting and readable labels) but takes a moment;
  1 = the answer cannot be identified from the chart.
```

Figure 8: Record-independent criterion texts used by the *w/o sample-specific rubrics* ablation in place of each record's own rubric.

Prompt for Ablation on Chart Evaluation: without criterion-wise inference

```
<User Message>
You are grading one chart-generation attempt against a rubric.

You are given ONE image: a chart produced as a response to the user query below.
That chart, as shown in the image, is what you are evaluating.

The source_table below is the oracle table: it holds the content the chart should
ideally represent. The candidate_code below is the code that produced the chart;
use it only as an auxiliary reference — always judge from the chart in the image,
never from the code alone. For data selection, value fidelity and label-encoding
fidelity, the candidate CODE and the SOURCE TABLE decide which rows, columns and
values were used and whether they are right — check what the code hard-codes
against the table, and count. Then confirm in the IMAGE that what the code
computes is what the chart actually shows.

You will be given the rubric after the image: {count} criteria, of which criteria
1-{record count} are specific to this record and criteria {first global}-{count}
are identical for every record. Each criterion is the only standard for its own
score: grade against what it requires, not against a chart you would have drawn
yourself, and never invent requirements it does not state. Grade each criterion
independently and literally — do not let one criterion influence another, and
never deduct the same defect under two criteria.

If a criterion does not specify its own scale, follow this 3-point scale:
  3 = the requirement is fully satisfied;
  2 = a defensible but suboptimal realization — below the standard the criterion
      describes, yet not wrong;
  1 = wrong or missing.

user_query:
{implicit query}

source_table:
{source table markdown}

candidate_code:
{candidate code}

The chart to evaluate:
{candidate chart image}

rubric:
{rubric — the record's criteria followed by the global criterion anchors}

Return valid JSON only, with EXACTLY {count} verdicts using EXACTLY these
criterion names in this order: {criterion name list}:
{
  "verdicts": [
    {"criterion": "...", "score": 1 | 2 | 3, "reason": "..."},
    ...
  ]
}
```

Figure 9: The single merged call used by the *w/o criterion-wise inference* ablation, which grades all criteria in one inference with unchanged criterion texts.

Prompt for the Grounding Agent in the Multi-Agent Harness

```
<System Message>
You are the data-sourcing step of a charting pipeline. You read a document and
a user's request, find the data that answers it, and hand the numbers to the
next step in a structured form. You do not draw anything.

<User Message>
Find the data that answers this request.

Pull out the values the request needs. They may all sit in one table, or they
may be spread across several — take whatever the request actually requires and
leave the rest. Copy the numbers exactly as printed: do not compute anything,
do not round, and do not convert units.

The later steps never see this document; everything they know about the data is
what you return. So carry across anything stated around the numbers that a
reader would need in order to read them correctly.

user_request:
{the user's request}

source_document:
{the source document, whole}

Return valid JSON only:
{"source_tables": ["each table you drew values from, as a markdown pipe table
                   copied from the document, including its caption or title
                   line if the document prints one"],
 "notes": ["anything stated outside the numbers themselves that a reader needs
           in order to read them correctly"],
 "columns": ["the column labels for the values below, in order"],
 "rows": [{"label": "the row label", "values": [number or null, one per column]}],
 "selection_reason": "one sentence on why these values answer the request"}
```

Figure 10: Prompt for the grounding agent, shared by both harnesses. It is the only stage that reads the source document; later stages see only the packet it returns.

Prompt for the Form Agent in the Multi-Agent Harness

```
<System Message>
You are the chart-design step of a charting pipeline. You decide what quantity
a chart plots and how that quantity is drawn, and you hand those decisions to
the step that writes the code. You do not write code and you do not choose
colors.

<User Message>
Decide the form of the chart that answers this request.

Two decisions, kept separate:

- derived_quantity: the quantity the chart actually plots. It may be the numbers
  exactly as they were handed to you, or something computed from them. Decide
  which quantity answers the request, and state it as a formula over the columns
  above so the coding step can compute it exactly. Do not compute it yourself;
  the code will.
- encoding: how that quantity is drawn — the chart type, and how the data maps
  onto it. Describe it in enough detail that the coding step has no geometry
  left to guess.

user_request:
{the user's request}

source_data:
{the retrieval packet: selected values as a markdown table, notes, source tables}

Return valid JSON only:
{"derived_quantity": {"description": "what is plotted, in words, including its
                      unit",
                     "formula": "how to compute it from the columns above;
                      write \"as given\" if the values are plotted as they are"},
 "encoding": "the chart type and how the data maps onto it, in prose",
 "rationale": "one or two sentences on why this form answers the request"}
```

Figure 11: Prompt for the form agent in the multi-agent pipeline (no external knowledge), which decides the value transformation and the chart encoding.

Prompts for the Catalogue-Augmented Form Agent in the Retrieval Harness

```
<System Message>
You are the chart-design step of a charting pipeline. You decide what quantity
a chart plots and how that quantity is drawn, and you hand those decisions to
the step that writes the code. You do not write code and you do not choose
colors.

<User Message>  (turn 1)
Decide the form of the chart that answers this request.

Two decisions, kept separate:

- derived_quantity: the quantity the chart actually plots. It may be the numbers
  exactly as they were handed to you, or something computed from them. Decide
  which quantity answers the request, and state it as a formula over the columns
  above so the coding step can compute it exactly. Do not compute it yourself;
  the code will.
- encoding: how that quantity is drawn — the chart type, and how the data maps
  onto it. Describe it in enough detail that the coding step has no geometry
  left to guess.

A reference catalogue is available: one list of chart forms and one of derived
quantities, each entry saying what it is and what a reader uses it for. Consult
it before you decide: give a lookup query for each of the two decisions and make
neither decision in this answer. The catalogue describes its entries in general
terms, so phrase a query the same way: say what kind of chart or quantity you
are looking for and what it should let a reader see, and do not mention this
data's column names, entities or values. You will receive the matching entries
and decide then.

user_request:
{the user's request}

source_data:
{the retrieval packet: selected values as a markdown table, notes, source tables}

Return valid JSON only:
{"lookup": {"derived_quantity": "a general query for the quantity",
            "encoding": "a general query for the chart form"},
 "rationale": "one or two sentences on what you need to find out"}

<User Message>  (turn 2)
The catalogue returned these entries.

chart forms matching "{the encoding query}":
{five cards, one per line}

derived quantities matching "{the derived quantity query}":
{five cards, one per line}

Now make the decision(s) still open: derived_quantity and encoding. Use an entry
when it fits the request, and decide on your own when none does. An entry's
formula names its inputs in general terms; restate any formula you use over the
columns of the data above. Do not compute it yourself; the code will.

Return valid JSON only:
{"derived_quantity": {"description": "what is plotted, in words, including its
                       unit",
                      "formula": "how to compute it from the columns above"},
 "encoding": "the chart type and how the data maps onto it, in prose",
 "consulted": ["the names of the entries you relied on; empty if none"],
 "rationale": "one or two sentences on why this form answers the request"}
```

Figure 12: Prompts for the catalogue-augmented form agent in our full harness. The first turn only formulates general lookup queries; the second turn receives the retrieved cards and makes both decisions.

Example Catalogue Card Returned to the Form Agent

```
A catalogue card, as it appears in turn 2:

- Pareto chart: A bar chart of category values sorted in descending order, with
  a line showing the cumulative total as a share. Use when: a few categories
  account for most of a total; the reader wants to see how many items cover a
  given share. Encoding: sorted bars for the values; a line on a secondary axis
  for the cumulative share, from 0 to 100%.
```

Figure 13: An example catalogue card as it appears in the form agent's second turn.

Prompt for the Aesthetic Agent in the Multi-Agent Harness

```
<System Message>
You are the visual-design step of a charting pipeline. The chart's form is
already decided; you decide how it looks, and you hand that to the step that
writes the code. You do not change the chart type or the quantity being plotted.

<User Message>
Decide how this chart should look.

The form is already fixed — the quantity and the chart type are not yours to
change. The visual treatment is: how the figure is laid out, what colours it
uses, and how its text and marks are arranged. Decide it deliberately, and
describe it concretely enough that the coding step can implement it without
guessing.

user_request:
{the user's request}

source_data:
{the retrieval packet: selected values as a markdown table, notes, source tables}

chart_form:
{the chart-form decision: derived quantity and encoding}

Return valid JSON only:
{"visual_design": "the visual treatment, concrete enough to implement directly",
 "rationale": "one or two sentences on why it suits this chart"}
```

Figure 14: Prompt for the aesthetic agent, which fixes layout, color, and text placement without changing the planned transformation or chart form.

Prompt for the Coding Agent in the Multi-Agent Harness

```
<System Message>
You will be given a user query and the design decisions a charting pipeline has
already made. Write Python code that answers the query with a chart.

Answer with the code alone, in a single Python block:

```python
# complete executable Python code
```

The code must be self-contained: use only pandas, numpy, matplotlib, math, or
datetime; hard-code any values you need (no file or network access); save the
chart with plt.savefig("chart.png").

<User Message>
user_query:
{the user's request}

source_data:
{the retrieval packet}

chart_form:
{the chart-form decision}

visual_design:
{the design decision}

Write the chart. Hard-code the values exactly as they are given above and let
the code do the arithmetic — compute the plotted quantity from the hard-coded
numbers rather than working it out yourself. Follow the decisions above; where
they are silent, use your judgement.
```

Figure 15: Prompt for the coding agent, which receives the three upstream decisions and writes the executable chart code.

Prompts for the Visual Critic and the Revision Turn

<User Message> (critic — fresh conversation every round, no system message)

```
You are reviewing a chart that an assistant produced for a user. You are given
the user's request, the Python code that made the chart, and the rendered chart
image.

Decide whether the chart, as rendered, serves the request well, and say what is
wrong with it if it does not.

Judge it from the image. The reader never sees the code, so something that is
right in the code but broken in the picture is broken. Read the request first,
then look at the chart and ask what a reader who made that request would
actually come away with: whether it is what they asked for, whether the chart
lets them get it without having to work for it, and whether everything the
chart states is true of what it plots.

Assume the numbers hard-coded in the code were read correctly from the source;
do NOT speculate about data you cannot see. Report only concrete problems that
a change to the code would fix, and pair each one with the change that would
fix it. Do not report matters of taste. If nothing is wrong, say the chart is
satisfactory.

user_request:
{the user's request}

chart_code:
{the current code}

The rendered chart image follows. Return valid JSON only:
{"satisfied": true | false,
 "problems": ["each concrete defect, one short sentence"],
 "suggestions": ["a concrete fix per problem"]}
```

<User Message> (coder — revision turn)

```
A reviewer looked at the rendered chart and found these problems:
{the critic's problems, one per line}

Suggested fixes:
{the critic's suggestions, one per line}

Make the smallest revision that fixes every problem; do not redesign the chart.
Reply with the complete, self-contained Python code in a single ```python
block, under the same constraints as before (pandas, numpy, matplotlib, math,
datetime only; no file or network access; save with plt.savefig("chart.png")).
```

Figure 16: Prompts for the visual critic, which reviews the rendered chart from the image, and the revision turn sent back to the coding agent (up to two rounds).