CHARTVL: BENCHMARKING SVG CHART GENERATION THROUGH VISUAL LAYER RECONSTRUCTION AND VERIFIABLE JUDGMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Evaluating SVG charts generated by large language models is challenging because appropriate layout, layering, and styling depend on the roles of chart elements and the task requirements. Existing approaches often use vision-language models (VLMs) to evaluate the visual quality of charts, but they are prone to missing or misjudging visual issues, and their judgments, expressed as scores or textual explanations, are difficult to verify. We introduce CHARTVL, a benchmark for SVG chart generation built on visual layer reconstruction and verifiable judgment. We reconstruct visual layers to inform VLM judgments and require executable conditions alongside each determinate visual-design judgment. Executing these conditions confirms or revises the initial verdicts using evidence from the rendered SVG. CHARTVL contains 360 tasks spanning four commonly used chart categories and three task levels with increasingly complex design requirements. Across four evaluated models, 83.1%–99.2% of outputs use the requested chart type and correctly present the required data and content, but only 9.7%–58.6% also pass the verified visual evaluation of layout, layering, and styling. Our findings show that chart generation requires more than correct content. Coordinating visual layers into a readable whole is a distinct challenge that calls for structured, verifiable evaluation.

1 INTRODUCTION

Large language models (LLMs) have made substantial progress in coding capabilities (OpenAI, 2026; Anthropic, 2026; Kimi Team, 2026) and are increasingly used to create charts by writing code with plotting libraries (Maddigan & Susnjak, 2023; Dibia, 2023; Ni et al., 2025) or directly generating scalable vector graphics (SVG) charts (Ni et al., 2026; Wang et al., 2026). Recent benchmarks evaluate model capabilities on chart generation by assessing both source code and rendered chart images (Yang et al., 2025; Tang et al., 2026a), often using vision-language models (VLMs) to evaluate visual quality (Chen et al., 2025a; Rahman et al., 2025). However, VLMs are prone to miss or misjudge visual issues in complex charts (Xie et al., 2026), such as text overlap (Zhang et al., 2025), canvas overflow (Chen et al., 2025a), and component occlusions (Wu et al., 2026), undermining evaluation reliability.

SVG charts arrange data marks, labels, and backgrounds spatially and draw them in sequence (Chen et al., 2026a). Explicit element annotations and layered visual prompts can aid chart understanding (Zhu et al., 2026). In Figure 1a, an annotation highlighting the 2022–2023 contraction and rebound covers part of the growth-rate line and nearly all of its 2022 value label. Reconstructing the relevant layers reveals these obscured elements (Figure 1b), providing additional structural evidence for VLM judgments.

VLM-as-judge methods typically return scores or textual explanations (Chen et al., 2025a; Xie et al., 2026), without testable conditions tied to specific chart elements. Checking these judgments against the rendered chart therefore requires additional human inspection or model judgment.

We present CHARTVL, a benchmark built on visual layer reconstruction and verifiable judgment. Using the task requirements, generated code, and rendered image, it reconstructs visual layers and maps them to SVG elements, exposing element roles and inter-layer relationships. Guided by pre-

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.

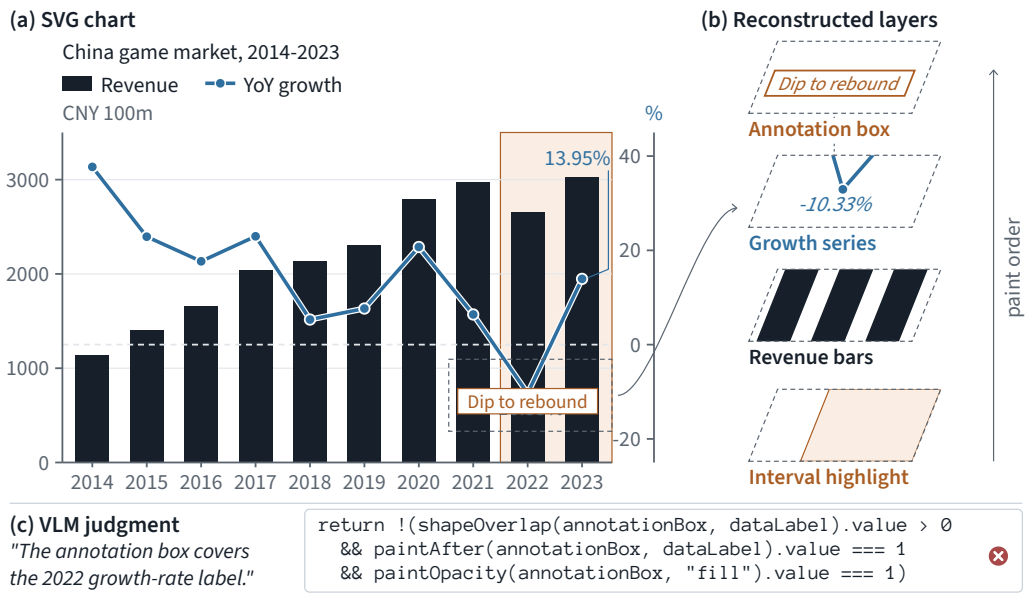

Figure 1: An example of hard-to-detect visual issues in SVG charts. (a) An annotation box hides part of the growth-rate line and nearly all of its 2022 label. (b) Reconstructed layers reveal the occluded elements. (c) An illustrative VLM judgment is paired with an assertion checking overlap, paint order, and fill opacity.

defined rubrics, a VLM evaluates the chart and provides judgments, reasons, and executable pass conditions over relevant elements. The harness executes these conditions against the rendered chart's geometry, style, and drawing order to confirm or revise the initial verdicts.

We construct 360 tasks spanning bar, line, pie, and glyph charts (Section 3.2). The 120 task groups each contain three levels of design requirements. Evaluation covers task fulfillment and three visual design dimensions, namely spatial planning, layer logic, and inter-layer design coherence (Section 3.3).

We evaluate four models on all 360 tasks (Section 5). Task fulfillment ranges from 83.1% to 99.2%, while success under the full protocol ranges from 9.7% to 58.6%. Spatial planning has the lowest design pass rate for every model. Three models also show lower success on tasks with more complex design requirements than on basic chart tasks, with paired differences of 20.0–28.3 percentage points despite small changes in fulfillment. These findings show why content fulfillment and visual design merit separate evaluation.

Our contributions are as follows.

- **Visual layer reconstruction.** We reconstruct visual layers and map them to rendered SVG elements, providing structural and semantic evidence for VLM evaluation.
- **Verifiable judgment.** We pair rubric-guided visual judgments with executable pass conditions over relevant elements, enabling programmatic confirmation or revision of the initial verdicts.
- **The ChartVL benchmark.** We provide 360 SVG chart generation tasks for comparing model performance across chart categories, task levels, and visual design dimensions.

2 Background and Related Work

2.1 Semantic Structure and Reconstruction of SVG Charts

Visualization grammars and scene representations make chart structure explicit (Satyanarayan et al., 2017; Liu et al., 2025). SVG exposes drawing primitives and groups, but their chart-level roles are

Table 1: Chart-generation evaluation protocols (✓ explicit support, ∼ task subset, ✗ not explicit).

Work	SVG Generation	Layout Defects	Component Relations	Structural Evidence	Executed Conditions
VisPlotBench	∼	✗	✗	✗	✗
DeepChart	✗	✓	✓	✗	✗
ChartMimic	✗	✓	✗	✓	✗
Chart2Code	✗	✗	✓	✓	✗
ChartEval	✗	✗	✓	✓	✗
VisEval	✗	✓	✓	✓	✗
ChartAnno	∼	✓	✓	✓	✗
CHARTVL	✓	✓	✓	✓	✓[†]

SVG denotes the generation target. Layout defects include overlap and clipping. Relations include data–axis, legend–series, and annotation–target links. Structural evidence is an explicit chart representation used in evaluation. Executed conditions are judge-written tests over referenced elements, run to check a verdict.
[†]Design judgments only, excluding task fulfillment.

not consistently represented across tools (Chen et al., 2026a; Lee et al., 2026). Deconstruction and semantic labeling recover encodings and component roles for reuse, interaction, and editing (Chen et al., 2024; Snyder & Heer, 2024; Lee et al., 2026).

LayerWiseBench derives layer roles, bindings, and visibility relations from executable chart templates to evaluate understanding and editing (Zhong et al., 2026). CHARTVL reconstructs layers in generated SVG charts for design evaluation. Task requirements constrain intended relationships, while code and image provide evidence about their realization. A label placed near the wrong series should retain its intended target. Inferred relationships remain candidates, not verified semantic ground truth.

2.2 EVALUATING CHART AND SVG GENERATION

ChartMimic and Chart2Code evaluate chart reproduction against references (Yang et al., 2025; Tang et al., 2026a). VisPlotBench and DeepChart evaluate generation from instructions and data (Ni et al., 2026; Tang et al., 2026b), while general SVG benchmarks also cover icons and illustrations (Chen et al., 2025b; Lin et al., 2025).

Structural evidence enters evaluation at different stages (Table 1). ChartMimic and Chart2Code compute structural metrics alongside model scores, while ChartEval compares parsed scene graphs (Goswami et al., 2025). VisEval provides overlap and overflow measurements to its judge (Chen et al., 2025a). ChartAnno combines structural statistics with criteria for annotation placement and target correspondence (Chen et al., 2026b). These protocols already support element-level checks, but such checks do not themselves make a VLM's design verdict executable.

VPEval evaluates images through programs invoking visual modules (Cho et al., 2023), while PROVE verifies constructed question-answer pairs with scene-graph programs (Prabhu et al., 2025). VisJudge-Bench studies agreement with expert ratings (Xie et al., 2026), while ChartJudgeBench tests judgments of controlled defects (Wu et al., 2026).

CHARTVL pairs each determinate design judgment with an executable pass condition over anchored SVG elements. The judge writes the condition before seeing runtime measurements, and execution can confirm or overturn its initial verdict.

3 THE CHARTVL BENCHMARK

CHARTVL evaluates whether LLMs can generate SVG charts that satisfy task requirements while preserving readability across interacting visual layers.

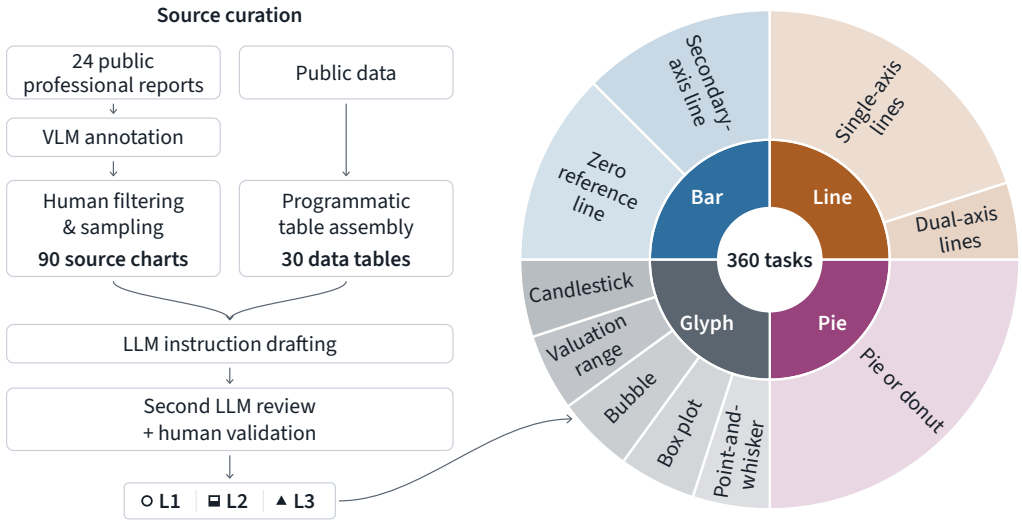

Figure 2: Construction and final distribution of CHARTVL. Report charts and public data yield 120 task groups and 360 tasks through source curation, instruction drafting, and review. The final dataset is balanced across four chart categories (inner ring), and the outer ring shows subcategories.

3.1 TASK FORMULATION

Each task instance consists of an English instruction specifying an analysis context, chart type, and content and design requirements, together with a data table. For task group g and level $\ell \in \{1, 2, 3\}$, we write the task and a model's generation as:

$$x_{g,\ell} = (q_{g,\ell}, T_{g,\ell}), \qquad h_{g,\ell} \sim p_M(\cdot \mid s, q_{g,\ell}, T_{g,\ell}), \tag{1}$$

where $q_{g,\ell}$ is the instruction, $T_{g,\ell}$ is the input table, s is the shared system prompt, and p_M is the model's output distribution under the specified generation configuration. The output $h_{g,\ell}$ must be a complete HTML document whose primary graphic is a static, inline SVG chart on a white background. The model may write SVG directly or construct it through JavaScript libraries such as D3. The system prompt requests a 960×540-pixel canvas and excludes animation and SVG/CSS clipping paths. We use one generation attempt per task without feedback-based repair or response post-processing.

Tasks are organized into three levels of design requirements. **L1** focuses on the data marks and their supporting axes, labels, or legends. **L2** specifies additional or alternative supporting layers, such as a reference line across bars, a shaded band behind lines, or external labels around pie slices. **L3** requires visual emphasis and an annotation or summary, together with other specified layers. These requirements concern how elements share space, how they overlap, and how their relationships remain identifiable. All three levels are evaluated under the same criteria for task fulfillment and visual design.

3.2 BENCHMARK CONSTRUCTION

We collected 24 publicly available professional reports containing quantitative data. A VLM annotated chart types and basic visual components, extracted data, chart text, and analysis context, and performed an initial screening for conventional chart structure and the presence of components appropriate to each chart type. We then manually excluded charts whose data were judged too simple and sampled by chart type, selecting 30 source charts each for bar, line, and pie charts.

To increase data diversity, we also constructed glyph-chart tasks from publicly available web pages, spreadsheets, and structured data files from institutional sources. We programmatically assembled 30 input data tables, covering point-and-whisker, box, valuation-range, bubble, and candlestick charts, with six instances of each type.

Using the data and analysis context of each instance, an LLM drafted task instructions at three levels of design requirements. A second LLM reviewed each instruction against its level specification, and the resulting task groups underwent further human validation. The 90 source charts and 30 glyph data tables yielded 120 *task groups*, each containing one task at L1, L2, and L3. Together, these groups yield 360 tasks, including 270 derived from report charts and 90 glyph tasks, with 90 tasks in each chart category (Figure 2).

Task groups retain the chart category across levels and reuse an identical data table in 118 of 120 groups. The remaining two rename a column or add reference values. Requirements can replace those at a preceding level. For example, pie tasks replace the L1 legend with external labels at L2. Within-group comparisons measure performance under different design requirements on closely matched data.

3.3 Evaluation Metrics

All rates count task instances rather than individual judgments, whose number and scope vary with the chart. We report visual design dimensions separately, following the multi-aspect reporting of HEIM (Lee et al., 2023), without averaging them into a weighted score. Design metrics use outcomes after assertion execution (Section 4.3). An unresolved or invalid judgment cannot establish a pass, but may coexist with a confirmed defect. We report evaluator uncertainty and structural verification failures separately.

Task-level metrics. The *task fulfillment pass rate* is the percentage of all tasks whose output renders a chart with the requested type, correct data values and mappings, and required content. This separates content fulfillment from visual design, as in VisEval (Chen et al., 2025a). We report instruction and content-completeness outcomes separately. Execution success alone, also measured by ChartMimic and Plot2Code (Yang et al., 2025; Wu et al., 2025), does not establish task fulfillment. The *success rate* is the percentage of all tasks that fulfill the task, pass structural verification, and pass every applicable design dimension. This all-conditions requirement parallels the hard satisfaction rate of FollowBench (Jiang et al., 2024). Explicit non-applicability does not block success, whereas unresolved or invalid outcomes do. We also report success after excluding task-fulfilling charts whose structural verification fails. Both task-level rates are reported overall, by level, and as paired L1–L3 differences within groups.

For design metrics, an *eligible chart* fulfills the task and passes structural verification. Each dimension's pass rate divides the number of charts receiving a pass by all eligible charts in the reported task set. A dimension passes only if at least one rubric is applicable and all applicable rubrics pass. Non-applicable, unresolved, and invalid outcomes remain in the denominator and are reported separately. Because eligibility can differ across models, we report its count alongside these conditional rates and the task-level metrics.

Spatial planning. Alongside its pass rate, we report *out-of-bound* and *collision failure rates*, each counting eligible charts with at least one confirmed failure under the corresponding rubric. These failures concern canvas clipping that removes necessary content and spatial conflicts that impair independent reading. Geometric intersection or boundary crossing alone does not constitute a defect.

Layer logic. The pass rate measures whether covering, drawing order, and compositing preserve necessary information and satisfy explicit ordering requirements. All eligible charts undergo this evaluation, including tasks that require layers across or behind data marks, such as reference lines, shaded bands, or lines crossing bars. We report the rate over the full eligible set.

Inter-layer design coherence. The pass rate measures whether spatial and style cues jointly support intended identity, correspondence, and distinguishability between related layers (Section 4.2). It applies to all eligible charts.

We compute 95% percentile confidence intervals by resampling task groups, preserving the dependence among their three levels. The same resampled groups are used across models and levels for paired comparisons.

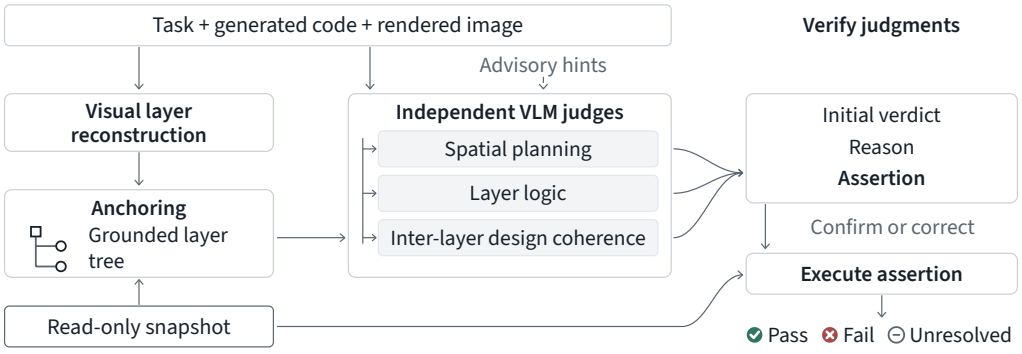

Figure 3: The evaluation framework of CHARTVL. Reconstructed layers are anchored to the rendered SVG. Three evaluators share the task, code, image, and layer tree, but judge separately under their respective rubrics. Advisory hints help locate relevant elements. Assertion execution can confirm or overturn the initial verdict when the required facts are available.

4 EVALUATION FRAMEWORK

CHARTVL combines visual layer reconstruction, rubric-guided VLM judgment, and executable assertions (Figure 3). Reconstruction connects chart-element descriptions to the rendered SVG. The judge uses these descriptions, the task, code, and image to evaluate local design choices. Each determinate judgment includes a program testing its pass condition against the captured rendering. Execution can confirm or revise the initial verdict.

4.1 VISUAL LAYER RECONSTRUCTION

The rectangles forming revenue bars, an interval highlight, and an annotation background in Figure 1 serve different roles. A VLM reads the task instruction and input data, generated code, and rendered image to write a *layer tree*. Its nodes describe individual or repeated SVG elements within the existing hierarchy, without artificial containers. A *visual layer* comprises elements serving a common role, such as data labels. Task requirements and code inform intended identities and relationships, while the image shows their realization.

The harness anchors the tree to an SVG snapshot captured in headless Chromium at 960×540 CSS pixels. Matching uses tags, parent-child relationships, sibling order, and optional text. A node receives element references only for a unique correspondence. Position and color cannot resolve ambiguity because they are themselves evaluated. Structural verification requires every reconstructed node to match uniquely and every required snapshot node to be accounted for, with limited runtime-verified exceptions for non-drawing elements (Appendix B). Anchoring establishes element identity, not the correctness of the description.

The reconstruction call also evaluates *task fulfillment*, including the requested chart type, data values and mappings, and required content. Visual defects such as a misplaced but present label belong to subsequent design evaluation. Only charts that fulfill the task and pass structural verification proceed to the three design dimensions. Task fulfillment is a model judgment without an executable assertion. For charts that fulfill the task, a failed structural verification is recorded as an evaluator error, distinct from a confirmed chart defect.

4.2 RUBRIC-GUIDED VLM JUDGMENT

Following explicit criteria in LLM-based evaluation (Liu et al., 2023; Kim et al., 2024), we define rubrics for spatial planning, layer logic, and inter-layer design coherence (Table 2). Each evaluator receives the task, code, original image, and anchored layer tree in a separate call, without other evaluators' judgments. The rubrics identify reading requirements and acceptable alternatives: the interval highlight in Figure 1 preserves overlapping data, whereas the annotation background hides necessary information, motivating a layer-logic judgment.

Table 2: The three visual design dimensions and representative defects. Each rubric evaluates the rendered result in its task context. Different dimensions may address the same objects.

Dimension	Evaluation criterion	Example defect
Spatial planning	Necessary content fits within the output canvas, and objects requiring independent reading have usable space. Harmless contact or decorative extension can be acceptable.	Canvas clipping removes a label, or a title and legend compete for readable space.
Layer logic	Drawing order, covering, and compositing preserve necessary information and satisfy explicit task requirements about front-to-back order.	An annotation background obscures a value label and part of a data series.
Inter-layer design coherence	Related layers use spatial and style cues that support their intended identity, correspondence, and distinguishability in context.	A label suggests the wrong target, or its text color makes it indistinguishable from its background.

The harness supplies advisory *hints* about candidate boundary violations, spatial intersections, structural context, and possible semantic relationships. Hints contain element references, sources, and applicability information, but no runtime measurements or preliminary verdicts. The judge may ignore, correct, combine, or expand them and evaluate other elements. Neither a candidate pair nor a DOM group fixes judgment scope, and an empty hint set does not bypass evaluation.

The judge chooses the objects, scope, and number of judgments, evaluating both acceptable and defective arrangements. Each judgment names relevant elements and provides an initial verdict and a rationale covering the reading requirement, visual evidence, and acceptable variation. Design coherence considers spatial and style evidence jointly; identical styling is not universally required. Each rubric receives an explicit applicability decision. Applicable rubrics require at least one judgment; non-applicability or insufficient evidence requires an explanation. An empty response never means a pass.

4.3 Verifiable Judgment

Building on program-based visual evaluation (Cho et al., 2023), we use an *executable assertion* to test a judgment's pass condition on referenced SVG elements. The assertion is a Boolean program over a read-only rendering snapshot and supported geometry and style queries. The judge receives interface declarations, but no runtime measurements or execution results. It writes the condition and initial verdict together, justifying numerical allowances through the rubric, task, and visual context.

A condition must address the requirement and element scope stated in the rationale. For example, the assertion in Figure 1 combines geometric intersection, drawing order, and fill opacity to test the specific covering configuration involving the annotation background and the value label. No universal overlap threshold determines whether a composition is acceptable. The available queries provide geometric and style evidence with defined limits, such as representing text by its bounding box rather than individual glyphs. Unsupported rendering effects or unavailable facts leave the relevant judgment unresolved.

The harness executes the assertion, interpreting `true` as a pass and `false` as a failure. Disagreement with the initial verdict is recorded as a correction. Invalid outputs receive at most one constrained repair attempt, while execution errors and missing evidence remain unresolved. The VLM selects and interprets the criterion, and execution verifies its stated condition on the captured evidence. Judgments, rationales, assertions, snapshots, and execution observations are retained for inspection (Appendix C).

5 Benchmark Experiments and Analysis

We examine task fulfillment, visual design, and performance across task levels for four models on all 360 tasks, yielding 1,440 generated outputs.

Table 3: Results on all 360 tasks per model. Rates are percentages, with 95% confidence intervals below the task-level rates. Success† excludes only task-fulfilling outputs that fail structural verification. The three design pass rates use all eligible charts, whose count is n. Layer logic is reported on the full eligible set. Rows are alphabetical.

	Task-level rates ↑				Design pass rates ↑		
Model	Fulfillment	Success	Success†	n	Spatial	Logic	Coherence
Claude Opus 5	96.4 [94.4, 98.1]	11.1 [7.8, 15.0]	11.3 [7.9, 15.1]	341	12.9	76.5	82.1
GPT 5.6 Luna	95.8 [93.6, 97.8]	23.3 [19.2, 27.5]	23.5 [19.2, 27.8]	342	30.4	79.5	70.8
GPT 5.6 Sol	99.2 [98.1, 100.0]	58.6 [53.3, 63.9]	59.3 [54.1, 64.4]	353	70.5	86.1	87.5
Qwen 3.5 Plus	83.1 [79.2, 86.9]	9.7 [6.9, 12.8]	9.9 [7.0, 13.0]	291	16.5	70.1	54.6

Setup. We evaluate Claude Opus 5, GPT 5.6 Luna, GPT 5.6 Sol, and Qwen 3.5 Plus under the same generation and evaluation settings. Each model produces one response per task, with a temperature of 1.0, a maximum output length of 64k tokens, and the provider's default reasoning setting. Layer reconstruction and visual evaluation use `gemini-3.8-flash` with a temperature of 0, the same output limit, and the provider's default reasoning setting. We report the metrics defined in Section 3.3, retaining all 360 tasks in the fulfillment and success denominators and all eligible charts in each design-dimension denominator.

We use 5,000 bootstrap resamples of the 120 task groups to compute 95% confidence intervals and paired differences, sharing resampled groups across models and levels. The intervals describe variation across task groups. Appendix D gives the statistical protocol and full counts.

Task fulfillment and overall success. Completing the requested content is substantially more common than satisfying all evaluation criteria (Table 3). Task fulfillment ranges from 83.1% to 99.2%, whereas success ranges from 9.7% to 58.6%. GPT 5.6 Sol achieves the highest observed success rate, succeeding on 211 of 360 tasks and fulfilling 357. Its success rate exceeds GPT 5.6 Luna's by 35.3 percentage points, with a paired interval of [29.2, 41.4]. The difference between Claude Opus 5 and Qwen 3.5 Plus has an interval of $[-2.5, 5.6]$ points, which includes zero. Excluding task-fulfilling charts that fail structural verification changes each success rate by at most 0.7 points.

Visual design after task fulfillment. Spatial planning has the lowest pass rate of the three design dimensions for every model, ranging from 12.9% to 70.5% on eligible charts. The corresponding ranges are 70.1–86.1% for layer logic and 54.6–87.5% for inter-layer design coherence. Confirmed spatial collisions are more frequent than out-of-bound failures for every model. For GPT 5.6 Sol, these affect 97 and 11 of 353 eligible charts, respectively. For Claude Opus 5, they affect 252 and 204 of 341 charts. A chart may have both defects. These results identify allocating readable space as a persistent difficulty even when the requested content is present. Appendix D reports the full design intervals, defect rates, and unresolved outcomes.

Performance across task levels. Success is lower at L3 than at L1 for GPT 5.6 Sol, GPT 5.6 Luna, and Qwen 3.5 Plus, with paired differences of 24.2, 28.3, and 20.0 percentage points, respectively (Table 4). All three intervals exclude zero. Task fulfillment differences are small and their intervals include zero for every model. For example, Luna fulfills 115 L1 tasks and 116 L3 tasks, but its successful outputs fall from 47 to 13. Claude Opus 5 remains at a low success rate across levels, with its L1–L3 interval including zero. These comparisons across design requirements reveal changes in overall success that task fulfillment alone does not capture.

Table 4: Performance across task levels, with 120 tasks per level. L1–L3 columns report success rates (%). The last two columns report paired L1 minus L3 differences in percentage points, with 95% confidence intervals below. Positive differences indicate a higher rate at L1.

	Success rate			L1 minus L3	
Model	L1	L2	L3	Success	Fulfillment
Claude Opus 5	10.8	13.3	9.2	1.7 [−5.0, 8.3]	−1.7 [−7.5, 4.2]
GPT 5.6 Luna	39.2	20.0	10.8	28.3 [18.3, 38.3]	−0.8 [−5.0, 3.3]
GPT 5.6 Sol	74.2	51.7	50.0	24.2 [11.7, 36.7]	−0.8 [−3.3, 1.7]
Qwen 3.5 Plus	21.7	5.8	1.7	20.0 [12.5, 27.5]	0.0 [−7.5, 7.5]

6 Discussion and Conclusion

We introduced ChartVL, a benchmark of 360 SVG chart-generation tasks and an evaluation framework combining visual layer reconstruction with executable assertions. Across the four models, task fulfillment is substantially more common than satisfying all criteria of the evaluation protocol. Spatial planning has the lowest design pass rate, and confirmed collision failures highlight difficulty allocating readable space among chart elements. Three models show lower overall success at L3 than L1 without corresponding evidence of lower task fulfillment, further distinguishing content fulfillment from accommodating different design requirements.

The framework connects local visual judgments to selected chart elements, rationales, and executable conditions. The retained initial verdict, assertion, and execution evidence allow readers to inspect which design choice was evaluated and what the final verdict depends on. This provides a concrete basis for diagnosing individual outcomes alongside the aggregate model results. Together, these findings motivate evaluating visual design alongside content fulfillment while preserving inspectable evidence for individual judgments. We will release the task instances, prompts, evaluation code, and configurations upon publication.

AI use statement

Generative AI assisted with benchmark construction (data processing and task instruction drafting and review), code and figure preparation, and literature review and manuscript preparation, including rewriting and polishing. The authors directed this work and take responsibility for all submitted content.

Reproducibility statement

Sections 3–5 specify the tasks, evaluation criteria and procedure, model settings, and statistical analysis. The complete generation prompt, task examples, and implementation specifications are provided in Appendices A–C. We will publicly release the task instances, prompts, evaluation code, and configurations upon publication.

References

Anthropic. Introducing Claude Opus 5.5, September 2026. URL https://www.anthropic.com/claude-opus-5-5.

Chen Chen, Bongshin Lee, Yunhai Wang, Yunjeong Chang, and Zhicheng Liu. Mystique: Deconstructing SVG charts for layout reuse. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):447–457, 2024. doi: 10.1109/TVCG.2023.3327354. URL https://hdi.cs.umd.edu/papers/Mystique_VIS23.pdf.

Chen Chen, Hannah K. Bako, Peihong Yu, John Hooker, Jeffrey Joyal, Simon C. Wang, Samuel Kim, Jessica Wu, Aoxue Ding, Lara Sandeep, Alex Chen, Chayanika Sinha, and Zhicheng Liu. VisAnatomy: An SVG chart corpus with fine-grained semantic labels. *IEEE Transactions on Visualization and Computer Graphics*, 32(1):560–570, 2026a. doi: 10.1109/TVCG.2025.3634263. URL https://arxiv.org/abs/2410.12268.

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 Transactions on Visualization and Computer Graphics*, 31(1):1301–1311, 2025a. doi: 10.1109/TVCG.2024.3456320.

Siqi Chen, Xinyu Dong, Haolei Xu, Xingyu Wu, Fei Tang, Hang Zhang, Yuchen Yan, Linjuan Wu, Wenqi Zhang, Guiyang Hou, Yongliang Shen, Weiming Lu, and Yueting Zhuang. SVGenius: Benchmarking LLMs in SVG understanding, editing and generation. In *Proceedings of the 33rd ACM International Conference on Multimedia (MM '25)*, pp. 13289–13296, 2025b. doi: 10.1145/3746027.3758287.

Zhenghan Chen, Zekai Shao, Lidan Tan, Xin Lin, Xingchen Zeng, Yi Shan, Ziyue Lin, Xiaoliang Fu, Xinyuan Liu, Yuetong Guo, Fen Wang, Bongshin Lee, and Siming Chen. ChartAnno: Benchmarking multimodal large language models for chart annotation generation. *arXiv preprint arXiv:2608.03464v2*, 2026b. URL https://arxiv.org/abs/2608.03464v2.

Jaemin Cho, Abhay Zala, and Mohit Bansal. Visual programming for step-by-step text-to-image generation and evaluation. In *Advances in Neural Information Processing Systems 36*, 2023. doi: 10.52202/075280-0265. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/13250eb13871b3c2c0a0667b54bad165-Abstract-Conference.html.

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

Kanika Goswami, Puneet Mathur, Ryan A. Rossi, Franck Dernoncourt, Vivek Gupta, and Dinesh Manocha. ChartEval: LLM-driven chart generation evaluation using scene graph parsing. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics: System Demonstrations*, pp. 86–93. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.ijcnlp-demo.10. URL https://aclanthology.org/2025.ijcnlp-demo.10/.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. FollowBench: A multi-level fine-grained constraints following benchmark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4667–4688. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.257.

Seungone Kim, Jay Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Ryan S Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/803485352e61e3ebf41221e4776c9fd4-Abstract-Conference.html.

Kimi Team. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026. doi: 10.48550/arXiv.2607.24653. URL https://arxiv.org/abs/2607.24653.

Jeongah Lee, Hima Varshini Surisetty, Durga Nirmaleswaran, Jahnavi Sharma, Srikiran Kavuri, Narges Mahyar, and Ali Sarvghad. Cohort-based semantic labeling: AI-enabled recovery of visualization semantics from deployed SVGs. *arXiv preprint arXiv:2606.09782*, 2026. URL https://arxiv.org/abs/2606.09782.

Tony Lee, Michihiro Yasunaga, Chenlin Meng, Yifan Mai, Joon Sung Park, Agrim Gupta, Yunzhi Zhang, Deepak Narayanan, Hannah Teufel, Marco Bellagente, Minguk Kang, Taesung Park, Jure

Leskovec, Jun-Yan Zhu, Fei-Fei Li, Jiajun Wu, Stefano Ermon, and Percy Liang. Holistic evaluation of text-to-image models. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023), Datasets and Benchmarks Track*, pp. 69981–70011. Curran Associates, Inc., 2023. doi: 10.52202/075280-3067.

Kevin Qinghong Lin, Yuhao Zheng, Hangyu Ran, Dantong Zhu, Dongxing Mao, Linjie Li, Philip Torr, and Alex Jinpeng Wang. VCode: A multimodal coding benchmark with SVG as symbolic visual representation. *arXiv preprint arXiv:2511.02778*, 2025. URL https://arxiv.org/abs/2511.02778.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG evaluation using GPT-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2511–2522. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153/.

Zhicheng Liu, Chen Chen, and John Hooker. Manipulable semantic components: A computational representation of data visualization scenes. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):732–742, 2025. doi: 10.1109/TVCG.2024.3456296. URL https://arxiv.org/abs/2408.04798.

Paula Maddigan and Teo Susnjak. Chat2VIS: Generating data visualizations via natural language using ChatGPT, Codex and GPT-3 large language models. *IEEE Access*, 11:45181–45193, 2023. doi: 10.1109/ACCESS.2023.3274199.

Yuansheng Ni, Ping Nie, Kai Zou, Xiang Yue, and Wenhu Chen. VisCoder: Fine-tuning LLMs for executable Python visualization code generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 2956–2983. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-emnlp.160.

Yuansheng Ni, Songcheng Cai, Xiangchao Chen, Jiarong Liang, Zhiheng Lyu, Jiaqi Deng, Kai Zou, Ping Nie, Fei Yuan, Xiang Yue, and Wenhu Chen. VisCoder2: Building multi-language visualization coding agents. In *The Fourteenth International Conference on Learning Representations*, 2026. arXiv:2510.23642.

OpenAI. GPT-6 Astra system card, September 2026. URL https://deploymentsafety.openai.com/gpt-6-astra.

Viraj Prabhu, Senthil Purushwalkam, An Yan, Caiming Xiong, and Ran Xu. Trust but verify: Programmatic VLM evaluation in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3258–3267, 2025. URL https://arxiv.org/abs/2410.13121.

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*, pp. 31849–31874. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1622.

Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. Vega-Lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23 (1):341–350, 2017. doi: 10.1109/TVCG.2016.2599030.

Luke S. Snyder and Jeffrey Heer. DIVI: Dynamically interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):403–413, 2024. doi: 10.1109/TVCG.2023.3327172. URL https://idl.cs.washington.edu/files/2024-DIVI-TVCG.pdf.

Jiahao Tang, Henry Hengyuan Zhao, Lijian Wu, Zijian Zhang, Yifei Tao, Dongxing Mao, Yang Wan, Jingru Tan, Min Zeng, Min Li, and Alex Jinpeng Wang. From charts to code: A hierarchical benchmark for multimodal models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13467–13566. Association for Computational Linguistics, 2026a. doi: 10.18653/v1/2026.acl-long.616.

Jiahui Tang, Kuicai Dong, Dexun Li, Hongchao Gu, Haocheng Yu, Wei Han, Chen Zhang, Yong Liu, Hao Wang, and Enhong Chen. DeepChart: How far are LLMs from faithful data-science chart generation? *arXiv preprint arXiv:2608.26757*, 2026b. URL https://arxiv.org/abs/2608.26757.

Yiming Wang, Ye Chen, Hanqi Chen, and Bingbing Ni. SVGEval: A vision-grounded framework for perceptual-quality benchmarking and evaluation in text-to-SVG generation. *arXiv preprint arXiv:2608.01977*, 2026. doi: 10.48550/arXiv.2608.01977. Accepted to ECCV 2026.

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*, pp. 3006–3028. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.164.

Lijian Wu, Henry Hengyuan Zhao, Zijian Zhang, Jiahao Tang, Jiajun Wu, and Alex Jinpeng Wang. ChartJudgeBench: Evaluating LMM judges for chart-to-code generation. *arXiv preprint arXiv:2609.24210*, 2026. URL https://arxiv.org/abs/2609.24210.

Yupeng Xie, Zhiyang Zhang, Yifan Wu, Sirong Lu, Jiayi Zhang, Zhaoyang Yu, Jinlin Wang, Sirui Hong, Bang Liu, Chenglin Wu, and Yuyu Luo. VisJudge-Bench: Aesthetics and quality assessment of visualizations. In *The Fourteenth International Conference on Learning Representations*, 2026. arXiv:2510.22373.

Cheng Yang, Chufan Shi, Yaxin Liu, Bo Shui, Junjie Wang, Mohan Jing, Linran Xu, Xinyu Zhu, Siheng Li, Yuxiang Zhang, Gongye Liu, Xiaomei Nie, Deng Cai, and Yujiu Yang. ChartMimic: Evaluating LMM's cross-modal reasoning capability via chart-to-code generation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/42806406dd99e30c3796bc98b2670fa2-Abstract-Conference.html.

Jiajun Zhang, Jianke Zhang, Zeyu Cui, Jiaxi Yang, Lei Zhang, Binyuan Hui, Qiang Liu, Zilei Wang, Liang Wang, and Junyang Lin. PlotCraft: Pushing the limits of LLMs for complex and interactive data visualization. *arXiv preprint arXiv:2511.00010*, 2025. doi: 10.48550/arXiv.2511.00010.

Xiaochuan Zhong, Yifan Hou, Chenxi Pang, and Shaobo Cui. Charts are beyond pixels: Probing for layer-wise chart understanding and editing. *arXiv preprint arXiv:2609.08657*, 2026. doi: 10.48550/arXiv.2609.08657.

Jiangning Zhu, Yuxing Zhou, Zheng Wang, Juntao Yao, Yima Gu, Yuhui Yuan, and Shixia Liu. InfoDet: A dataset for infographic element detection. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Wj0Sc9WBHZ.

A Task Data and Generation Setup

This appendix gives the generation prompt (Appendix A.1), one example task group per chart category (Appendix A.2), the sources, construction, and license of the tasks (Appendix A.3), and statistics of the benchmark (Appendix A.4).

A.1 Generation

Each task uses one generation attempt without feedback-based repair. The system message is the prompt below, identical for all tasks and models, and the user message is the complete task prompt. The response is loaded into a browser page exactly as returned, so it must itself be a complete HTML document. No code is extracted from Markdown fences or surrounding prose, and any such text becomes part of the page. Appendix D reports the generation settings.

> Return one complete HTML document rendering a static chart at 960 by 540 pixels on a white background, with one inline SVG as the primary graphics surface.
>
> You may construct the SVG directly or generate it at runtime with JavaScript, including D3. HTML containers and CSS are allowed, as are CDN libraries, external stylesheets and remote fonts.
>
> Use no animation, SVG clipPath, or CSS/SVG clip-path. Follow all task-specific content requirements and data in the user prompt.

The prompt excludes animation and clipping paths to support evaluation from a static rendering and simplify geometric queries. Measurement limits are described in Appendix C.

A.2 Example Task Groups

We quote one task group from each chart category. The instructions are verbatim. Each of these prompts ends with the data table, which the three levels share.

Bar (`chartvl_d008`).

> **L1.** In an industry outlook covering the development of China's game market and the sector's adoption of AI, I need a chart showing the trajectory of actual sales revenue and the year-over-year growth rate from 2014 to 2023. Please use bars for the annual actual sales revenue of the China game market (in CNY 100 million) and present each growth rate (in %) only as exact text aligned beside its corresponding year on the shared category x-axis; keep amounts on their own labeled vertical axis and keep the percentage values visually distinct from the bars without drawing a line, markers, or a separate percentage axis. Do not invent any missing value.
>
> **L2.** For a sector review on China's game industry growth and the expanding role of AI in development, please create a bar-and-line chart showing the long-run revenue trajectory and growth-rate fluctuations from 2014 to 2023. Use bars for the China game market actual sales revenue (CNY 100 million) and one continuous line with clear point markers for the year-over-year growth rate (%), both on the same horizontal year axis. Give the bars a labeled vertical axis for CNY 100 million and the line a distinctly labeled vertical axis for percentages, and keep the line legible where it crosses the bars. Do not add a zero-growth reference line or any interval emphasis annotation.
>
> **L3.** In an equity-research update on China's game sector and the integration of AI across development pipelines, I want a dual-axis bar-line chart for 2014-2023 that tracks China game market actual sales revenue (CNY 100 million, left axis, bars) and the year-over-year growth rate (%, right axis, line with point markers). To frame positive-versus-negative growth context, include a low-contrast zero-growth reference line on the percentage axis. Emphasize the most notable growth interval—the 2022 contraction followed by the 2023 rebound—and add one concise annotation explaining that emphasized interval, while keeping ownership of bars, line, axes, and annotation unambiguous.

The table gives, for each year from 2014 to 2023, the actual sales revenue of the China game market in CNY 100 million and its year-over-year growth rate in percent.

Year	Revenue	Growth (%)	Year	Revenue	Growth (%)
2014	1140	37.7	2019	2310	7.66
2015	1410	22.9	2020	2790	20.71
2016	1660	17.67	2021	2970	6.4
2017	2040	22.98	2022	2660	−10.33
2018	2140	5.32	2023	3030	13.95

Line (`chartvl_d013`).

> **L1.** I'm finalizing Section 5.2 of a media-sector equity research note, which argues that the AIGC theme catalyzed the 2023 re-rating of the media sector. I want a dual-axis line chart that traces the co-movement of the Shenwan Media index and ChatGPT total daily active users from October 2022 through January 2024, so readers can see how AI adoption tracked sector performance. Please plot both series clearly with data-point marks at every observation, keeping each series and every value easy to distinguish.
>
> **L2.** I'm updating a media-sector industry outlook that attributes the 2023 valuation rebound to the AIGC theme. To support this, I need a dual-axis line chart comparing the Shenwan Media index against ChatGPT total daily active users from October 2022 to January 2024, so the reader can judge how closely AI user growth moved with sector performance. Please plot both series with data-point marks at every observation, and add one continuous translucent band behind the lines spanning the vertical gap between the two series at each date, so the distance between them is directly visible without obscuring the lines or points.
>
> **L3.** I'm preparing a risk-and-catalyst review of the media sector for an equity research client, focused on how the AIGC-driven re-rating played out in 2023. I'd like a dual-axis line chart plotting the Shenwan Media index and ChatGPT total daily active users from October 2022 to January 2024, with data-point marks at every observation. To give overall context, add a low-contrast horizontal reference line on the media index axis at the initial October 2022 level of 450 points. Emphasize the steep rally interval from 2023-01-10 to 2023-04-10, when the index climbed from 550 to 850 as ChatGPT DAU rose from 25 to 55 million, and add one concise annotation noting that accelerating AIGC adoption drove the sector re-rating during that window.

The table lists 16 monthly dates from 2022-10-10 to 2024-01-10, with the index level in points and the daily active users in millions.

Pie (`chartvl_d005`).

> **L1.** As part of a book retail industry outlook review, I need to show how sales channels are distributed by market share. Please create a pie or donut chart where each arc is sized accurately by its share, and identify every channel name and exact value clearly in a compact legend.
>
> **L2.** For an industry channel performance update, I want to visualize the share of each book retail sales channel. Please use a pie or donut chart, size each arc accurately by its share, and place one complete set of name-and-value labels outside the chart. Use short leader lines only where needed so every label remains clearly tied to the correct arc.
>
> **L3.** In a book retail channel performance review, I need a donut chart showing the market share of each sales channel. Please size each arc accurately by its share, coordinate clear external labels with necessary leader lines for unambiguous association, visually emphasize the largest component, and provide a center summary noting that traditional e-commerce and short video e-commerce together account for 83.59% of the market.

The table gives the shares of five channels, namely traditional e-commerce (50.52%), short video e-commerce (33.07%), brick-and-mortar retail (7.51%), group buying (4.70%), and library distribution (4.20%).

Glyph (`chartvl_d019`).

> **L1.** I'm preparing a monetary-policy review of the Bank of Canada's First Quarter 2026 Market Participants Survey and need to compare the reference estimates and their cross-sectional interquartile ranges for the expected policy rate across the surveyed horizons. Please build a point-and-whisker chart for every supplied item, showing each item's lower

and upper bounds with the reference estimate marked as a point, along with item labels and a clearly labeled value axis in percent.

L2. For a policy-rate analysis based on the Bank of Canada's First Quarter 2026 Market Participants Survey, I'd like to visualize the reference estimates and their cross-sectional interquartile ranges across the surveyed forecast horizons. Create a point-and-whisker chart for every supplied item, displaying lower and upper bounds and the stated reference estimate as a point, with item labels and a clearly labeled value axis in percent. Add one complete set of exact reference-estimate labels positioned beside the estimate points.

L3. I'm reviewing the dispersion of policy-rate expectations in the Bank of Canada's First Quarter 2026 Market Participants Survey and want to see how the interquartile ranges and reference estimates shift across horizons from 2027 January through 2028 Q2. Produce a point-and-whisker chart for every supplied item, showing lower and upper bounds and the reference estimate as a point, with exact reference-estimate labels beside each estimate point. Visually emphasize one item with the widest supplied interval and add one concise annotation that states its lower bound, reference estimate, upper bound, and the interval meaning as a cross-sectional interquartile range with median. Compute interval width as Upper minus Lower without intermediate rounding. If multiple items tie, select the first in table order; this selects 2027 Q2.

Glyph prompts place a source block between the instruction and the table. Here it names the survey, its publisher, the publication date, an analysis context, the displayed period, and the unit. The table gives, for seven horizons from 2027 January to 2028 Q2, the lower bound, reference estimate, and upper bound of the expected policy rate, together with the interval type, the unit, and the number of responses.

A.3 Sources, Construction, and License

The 270 bar, line, and pie tasks, in 90 task groups, derive their data and analysis context from 24 publicly available professional reports containing quantitative data. A VLM annotated chart types and basic visual components and extracted data, chart text, and analysis context. Initial screening considered conventional chart structure and the presence of components appropriate to each chart type. Manual screening then excluded charts whose data were judged too simple, and sampling by chart type yielded 30 source charts each for bar, line, and pie charts.

The resulting tables may simplify, translate, or reorganize historical figures from these reports, so they should not be used as financial data. In task group `chartvl_t034`, the values for the five periods before 2025H1 are approximate readings from a chart in the source, and its net profit and growth labels are separate approximations that should not be derived from each other.

The glyph tasks broaden the data sources through publicly available web pages, spreadsheets, and structured data files. Code assembled 30 input data tables for the 30 glyph task groups, which contribute 90 tasks. Each glyph prompt names its data publisher. Sources include central banks (6 task groups), international organizations such as the International Monetary Fund and the World Bank (11), a U.S. bank regulator (1), company filings with the U.S. Securities and Exchange Commission (6), and two Taiwanese exchanges (6).

For each task group, an LLM drafted the instructions of the three levels from a specification of each level. For the bar, line, and pie task groups, it drafted the three instructions together. For the glyph task groups, the data of each prompt were assembled by code, and the LLM wrote only the instructions. A second LLM then reviewed every level against its specification, and the task groups underwent further human validation. A task group that failed review was replaced by one built from a new source rather than repaired.

Upon publication, the task instances and prompts will be released under CC BY 4.0 (Section 6). The license covers only material we wrote and grants no rights in the underlying facts, figures, reports, or filings of third parties. The source documents are not redistributed.

A.4 Statistics

Table 5 summarizes the benchmark. Every task group has exactly one task per level, and every prompt contains exactly one data table. Bar task groups comprise 23 charts with one bar series and

Table 5: Statistics of CHARTVL by chart category. Data rows count the rows of the input table, with the median and range over task groups. Word counts are medians over tasks, and instruction words exclude the data table and, for glyph tasks, the source block.

	Task	Data rows	Prompt	Instruction words		
Chart category	groups	median (range)	words	L1	L2	L3
Bar	30	10.5 (3–33)	152	105.5	103.5	109
Line	30	11 (6–21)	143	84	98	101.5
Pie	30	6 (3–10)	99	71.5	74	84
Glyph	30	7.5 (3–10)	229.5	74.5	88	107
All	120	8 (3–33)	147	81	91.5	96

7 with grouped bars, and line task groups 24 single-axis and 6 dual-axis charts with two to six series. The bar subcategories in Figure 2 follow the L2 design requirements, with 15 groups requesting a zero reference line and 15 requesting a line on a secondary axis. Pie tasks leave the choice between pie and donut to the model except where an instruction names one, and the glyph category has six task groups each of box plots, points with error bars, valuation ranges, bubble charts, and candlestick charts.

The data table is identical across the three levels in 118 task groups. In `chartvl_t010`, one column header is renamed across levels, and in `chartvl_t004`, the L3 table adds two industry-average columns for the reference line that L3 requests. Each level has its own instruction, which opens with an analysis context such as a sector review, and no task group keeps its opening sentence across the three levels. The median instruction has 81 words at L1, 91.5 at L2, and 96 at L3. Some L1 and L2 instructions explicitly exclude elements that a higher level adds, for example "Do not add a zero-growth reference line or any interval emphasis annotation." At L3, 110 instructions name the emphasized element or interval or state how to select it, and the other 10, from five bar and five line task groups, ask for the most notable interval without naming it. All 30 pie instructions at L3 ask for center text or an annotation, and 3 of them require a summary at the center.

The levels are not nested. In all 30 pie task groups, L2 asks for external labels, with leader lines where needed, instead of the legend of L1. In all 30 line task groups, L3 no longer asks for the band that L2 adds. In the 15 bar task groups whose L2 draws a second data series as a line on a secondary y-axis, L1 gives that series as text beside the categories, as in `chartvl_d008`. Eight bar instructions already ask for styling of some bars at L1, such as a distinct fill for a named bar or colors by sign, and in four of these task groups L3 emphasizes the bar farthest from zero instead of the bar or bars emphasized at L1 and L2.

The per-level counts in this appendix were taken from the instructions. We removed the sentences that exclude an element, such as the example above, matched phrases for the requested elements, and read the whole instruction wherever a match was ambiguous. A task group is counted only when its instruction names the element.

B VISUAL LAYER RECONSTRUCTION

Reconstruction input and representation. The reconstruction model receives the original task instruction, including its input data, the generated HTML, and the available rendered image. Only the outer Markdown response fence is removed from the HTML. The harness does not repair its contents. Neither the browser snapshot nor capture diagnostics, runtime element references, or model identity enter this request. The response contains the layer tree, proposed semantic relationships, and a task-fulfillment judgment.

The tree is an indented sequence of SVG tags with natural-language role descriptions. Two spaces indicate one level of nesting. The root is an SVG element, and each child describes an actual child in the generated SVG DOM. The model must not introduce a container for a conceptual group or a JavaScript loop. An optional literal records the complete `textContent` of a text-bearing element, including its descendants. Whitespace is normalized for matching. Literal text is separate from the

role description, and omitting an uncertain literal does not permit omitting the corresponding `text` or `tspan` element.

When the source determines the elements, the model enumerates them, accounting for filters and conditional creation. Otherwise, a tag followed by `[]` denotes one or more consecutive siblings with that tag. Such a repetition cannot cross a parent boundary or an intervening element that must be matched. A repeated container applies its child pattern separately to each member. The model supplies no counts, selectors, IDs, classes, or runtime references. Proposed relationships refer to tree lines and currently link legends to series, annotations to leaders, and leaders to targets. They provide candidates for later judgment rather than verified semantic facts.

Unique correspondence and coverage. For each parent, the harness compares the ordered child sequence in the tree with the snapshot, using tags, nesting, sibling order, and optional literal text. It considers all complete correspondences and binds a node only when every feasible correspondence assigns it the same elements. Consecutive repetitions of the same tag remain ambiguous even if one particular snapshot appears to admit a unique split. The harness never selects the first match or uses position, color, IDs, or classes to resolve ambiguity.

Coverage is required in both directions. Every declared node must have a unique correspondence, and every snapshot element that the rules require must be represented. If a child sequence has no complete correspondence, the harness retains any already anchored parent and marks its affected descendants as missing, unanchored, or ambiguous. Other branches may still be inspected, but partial success does not authorize partial design scoring. Expansion preserves actual parent references and copies role descriptions to matched members without inventing more specific semantic identities.

Limited omissions and failure handling. The matcher first seeks a complete correspondence without omissions. Only if this fails does it allow runtime-certified omissions of non-drawing elements. These are SVG scripts, empty groups with no element children or rendering dependencies, and zero-length lines with butt caps and no markers, subject to additional checks for unsupported effects and references. Zero area, a hidden appearance, or an auxiliary role alone is insufficient. Referenced definitions and containers with required descendants remain necessary. Omissions and their reasons are recorded separately from blocking coverage gaps, without deleting or renumbering snapshot elements.

The tree is structurally verified only when no anchoring issue or required coverage gap remains. This does not certify roles, semantic partners, data correctness, or a complete rendering tree. Reconstruction permits at most one review shared by format errors, unresolved task fulfillment, and eligible structural failures. The review receives the original evidence and prior response, without runtime diagnostics. The last response with a valid format is selected. When task fulfillment passes but structural verification fails, the case is an evaluator error and subsequent design evaluation is not run. Task-fulfillment failure or uncertainty independently blocks that evaluation.

C Rubrics and Pass Conditions

Common evidence. Each visual evaluator receives the original task, generated HTML including CSS and JavaScript, original image, one expanded layer tree with element references, and advisory hints. The evaluator's fixed instructions provide its rubrics and the assertion interface. Generation-system instructions, model identity, reconstruction diagnostics, and other evaluators' judgments are excluded. Code provides evidence about construction and data binding, not proof of visible correctness. The runtime snapshot and its measurements are withheld until assertion execution, although the source code itself can contain numeric positions and styles.

Spatial planning. The boundary criterion concerns necessary content cut off by the actual output canvas, not by an inferred plotting area. The collision criterion concerns objects whose positions or extents prevent independent reading, including indistinguishable text, confused contours, or content intruding into another object's reading space. A positive overflow or intersection measurement alone is insufficient. Harmless stroke bleed, justified decorative extensions, ordinary series crossings, text inside a mark, and readable close placement can be acceptable. Shared semantics or a shared parent

do not excuse a collision. A layering failure does not become a spatial-planning failure merely because moving an object could repair it.

Layer logic. This rubric checks whether covering and compositing preserve necessary information and satisfy explicit task requirements about drawing order. Necessary information includes existing labels, marks, trends, and meaningful boundaries, even when the instruction does not individually name them. A judge considers the complete local component and relevant covering layers. For a backed label, these may include its text, background, and intersecting series. No universal overlap fraction or opacity threshold establishes acceptability. Small occlusions can remove a minus sign or a necessary peak. When only an effect is requested, different layer orders achieving that effect are acceptable. General visual hierarchy and aesthetic emphasis are outside this rubric.

Inter-layer design coherence. Spatial and style evidence jointly establish whether related layers communicate identity, grouping, and reference. Position, containment, connections, colors, and line patterns must be interpreted in their semantic context. A clearly connected distant label can be preferable to a nearby label suggesting another series. Identical text colors can be inappropriate on different backgrounds. Text that is difficult to distinguish from its actual background is a style-coordination defect, whereas covering by another object is evaluated under layer logic. Neither a DOM parent nor nonzero text opacity identifies a readable background relationship. Small font size alone and general stylistic uniformity are outside these rubrics. The dimensions may evaluate the same objects independently.

Hints and judgment scope. Boundary and intersection candidates come from runtime geometry. Structural contexts preserve actual parents, while semantic relationship candidates come from reconstruction. Possible text backgrounds use geometric adjacency and supported drawing-order information. Each hint group states its source, method, and applicability, without measurements or preliminary verdicts. The model may ignore, correct, combine, split, or extend these advisory candidates, and select any valid tree references. Neither a candidate pair, a connected group, nor an empty candidate set establishes an evaluation scope or a pass.

Responses, repair, and aggregation. Each rubric receives an explicit applicability declaration. An applicable rubric requires at least one judgment. A judgment records its rubric, model-selected references, initial verdict, rationale, and assertion. Explicit uncertainty uses a null verdict and no program. A rubric declared inapplicable or unresolved requires a reason and has no judgments. Thus an empty response cannot mean that a chart passed.

The harness validates response structure, rubric identifiers, references, and the program interface. Malformed or incomplete records receive at most one repair, restricted to pending records. Accepted applicability decisions and judgments, including explicit uncertainty and execution errors, remain unchanged. Execution returns pass for `true`, fail for `false`, and unresolved for unavailable required facts or runtime errors. Invalid syntax or interface use remains distinct and may trigger repair. A stage fails if any judgment fails. Otherwise invalid records take precedence over unresolved records, and a stage is inapplicable only when all its rubrics are explicitly inapplicable. All remaining applicable judgments must pass for the stage to pass. Open issues remain recorded even in failed stages. A task succeeds only when task fulfillment passes and every design dimension passes or is explicitly inapplicable.

Capture and measurements. The capture protocol renders HTML in headless Chromium at 960×540 CSS pixels with device scale factor one. After page load, it waits 750 ms and then waits for font readiness, with a 10-second limit per operation. It collects the snapshot followed by an unmodified viewport PNG. The snapshot records SVG tags, parents, text, attributes, computed styles, local and viewport bounding boxes, coordinate transforms, and supported shape geometry. Capture diagnostics and protocol settings are archived. Infrastructure failures are distinguished from generated-chart defects.

Table 6 summarizes the query interface. Lengths and areas use CSS pixels and square CSS pixels, respectively. Text geometry uses whole-text boxes rather than glyph outlines. Curves are approximated, and supported dashed strokes use a continuous-stroke proxy for overlap and region distance. These approximations do not establish pixel-level coverage. Paint descriptors contain base colors

and opacity factors before compositing, not final displayed colors or effective contrast. Unsupported effects or missing geometry cannot be replaced by zero measurements.

Table 6: Measurements available to assertions and their representations. Each query returns a value with supporting observations and, for numeric measurements, a unit.

Query	Returned evidence	Representation and scope
`boundsOverflow`	Maximum overflow outside the viewport, floored at zero.	Uses layout bounds with stroke expansion. Supported containers aggregate visible descendants. Does not measure plot-area clipping.
`shapeOverlap`	Intersection area of transformed fill and stroke regions.	Basic shapes and supported single paths. Text uses boxes and dashed strokes use continuous proxies.
`paintAfter`	Relative drawing order of two elements.	Restricted static SVG ordering. DOM indices alone do not establish general compositing order.
`paintOpacity`, `frontOpacity`	Channel opacity, or the maximum foreground-channel opacity.	Solid paint and single-element opacity. An opaque outline does not make the interior opaque.
`paint`, `style`	Base RGB and alpha factors, or opaque solid-paint values.	No final composited color or contrast. Gradients, patterns, and heterogeneous text paint are unsupported.
`distanceTo`, `distanceToPath`	Point-to-region distance, or point-to-continuous-centerline distance.	The assertion derives a justified point from runtime geometry. Centerline distance excludes stroke width and dash gaps.

Assertion execution. Assertions are synchronous JavaScript functions with the interface `assertion(snapshot, geometry)`, returning a Boolean. Execution uses a read-only snapshot and permits no network, filesystem, imports, or nondeterministic clock access. It is bounded by an interpreter-step budget, a 10-second safety limit, a 64 MiB heap limit, and 256 geometry queries. Exceeding a resource limit leaves the judgment unresolved. Tool arguments and results are recorded, while direct snapshot reads are not individually traced. Reuse instances, group opacity, clipping, masks, filters, and unsupported composition can make required queries unavailable. The VLM determines the condition's semantic scope, and the harness tests that condition on the captured evidence.

D Extended Results and Statistical Analysis

Model settings. All four models use the shared generation prompt in Appendix A.1, a temperature of 1.0, a maximum output length of 64k tokens, and the provider's default reasoning setting. The reconstruction and evaluation model is `gemini-3.8-flash`, with a temperature of 0, the same output limit, and its provider's default reasoning setting. Each model generates one response per task.

Statistical procedure. We use 5,000 cluster bootstrap resamples with seed 20260926. Each resample draws 120 task groups with replacement, without stratification by chart category, and uses the same groups across models, levels, and metrics. Each rate is recomputed as the sum of its numerators divided by the sum of its denominators. We report the 2.5th and 97.5th percentiles. No reported rate has a zero denominator in any resample. For paired L1 minus L3 differences, we take the mean of the two binary outcomes' difference within each task group. Model differences are computed on the same resampled groups. Rates of 0 or 100% are accompanied by counts rather than degenerate intervals. The intervals quantify variation across sampled task groups.

Design dimensions and spatial defects. Table 7 gives the intervals omitted from the main design table. All three dimensions, including layer logic, use the full eligible set. Table 8 reports charts with at least one confirmed failure under each spatial rubric. The two defect counts can overlap and therefore do not partition spatial failures.

Table 7: Design pass rates (%) with 95% confidence intervals, followed by passing-chart counts. All rates use the eligible count n.

Model	n	Spatial planning	Layer logic	Design coherence
Claude Opus 5	341	12.9 [9.2, 16.9] (44/341)	76.5 [72.0, 80.9] (261/341)	82.1 [78.2, 85.7] (280/341)
GPT 5.6 Luna	342	30.4 [25.1, 35.6] (104/342)	79.5 [74.9, 84.0] (272/342)	70.8 [65.8, 75.6] (242/342)
GPT 5.6 Sol	353	70.5 [65.4, 75.4] (249/353)	86.1 [82.5, 89.5] (304/353)	87.5 [83.9, 91.0] (309/353)
Qwen 3.5 Plus	291	16.5 [12.5, 20.6] (48/291)	70.1 [64.8, 75.3] (204/291)	54.6 [49.0, 60.4] (159/291)

Table 8: Confirmed spatial defects on eligible charts. Rates (%) and 95% confidence intervals use the same denominator n as the design pass rates.

		Out-of-bound failures		Collision failures	
Model	n	Rate [CI]	Count	Rate [CI]	Count
Claude Opus 5	341	59.8 [53.9, 65.2]	204	73.9 [68.8, 78.7]	252
GPT 5.6 Luna	342	12.3 [8.9, 15.9]	42	64.0 [58.3, 69.7]	219
GPT 5.6 Sol	353	3.1 [1.4, 5.1]	11	27.5 [22.7, 32.6]	97
Qwen 3.5 Plus	291	38.5 [32.6, 44.6]	112	73.2 [68.6, 77.6]	213

Outcome states. Table 9 separates confirmed failures from unresolved or absent outcomes. Structural exclusions count only task-fulfilling charts whose layer tree fails verification. No eligible dimension is entirely non-applicable or invalid in this snapshot. For Qwen 3.5 Plus, two spatial failures and two coherence failures also contain unresolved judgments. Thus the numbers of eligible charts with any unresolved judgment are 4, 15, and 5 in its spatial, logic, and coherence dimensions. For the other models, these counts equal the unresolved-outcome counts in the table. An unresolved judgment is retained even when another judgment already establishes failure.

Table 9: Outcome counts. Fulfillment uses all 360 tasks and reports pass/fail/unresolved/not evaluated. Design columns use eligible charts and report pass/fail/unresolved. Struct. counts the task-fulfilling outputs excluded in Success†.

Model	Fulfillment	Struct.	Spatial	Logic	Coherence
Claude Opus 5	347/4/8/1	6	44/297/0	261/79/1	280/57/4
GPT 5.6 Luna	345/5/6/4	3	104/238/0	272/68/2	242/97/3
GPT 5.6 Sol	357/0/2/1	4	249/103/1	304/29/20	309/37/7
Qwen 3.5 Plus	299/51/6/4	8	48/241/2	204/72/15	159/129/3

Table 10 gives the instruction and completeness outcomes separately. These checks address overlapping aspects of task fulfillment, so their failures are not mutually exclusive error types. An absent subcheck is listed as not recorded rather than assigned a verdict.

Results by level. Table 11 reports both task-level metrics at every level. Each model-level cell covers 120 tasks. The paired level differences appear in Table 4. Task groups match closely related data across levels, while their instructions may replace design requirements. Level differences therefore do not isolate the effect of increasing layer count.

Table 10: Task-fulfillment subchecks over all 360 tasks, reported as pass/fail/not recorded. No individual subcheck is recorded as unresolved or invalid in this snapshot.

Model	Instruction	Completeness
Claude Opus 5	347/4/9	347/4/9
GPT 5.6 Luna	345/5/10	346/4/10
GPT 5.6 Sol	357/0/3	357/0/3
Qwen 3.5 Plus	300/50/10	316/34/10

Table 11: Task-level rates (%) with counts and 95% confidence intervals. Every denominator is 120. The 100% fulfillment rate is accompanied by its count without an interval.

		Task fulfillment		Success	
Model	Level	Count	Rate [CI]	Count	Rate [CI]
Claude Opus 5	L1	113	94.2 [89.2, 97.5]	13	10.8 [5.8, 16.7]
	L2	119	99.2 [97.5, 100.0]	16	13.3 [7.5, 19.2]
	L3	115	95.8 [91.7, 99.2]	11	9.2 [4.2, 15.0]
GPT 5.6 Luna	L1	115	95.8 [91.7, 99.2]	47	39.2 [30.8, 48.3]
	L2	114	95.0 [90.8, 98.3]	24	20.0 [13.3, 27.5]
	L3	116	96.7 [93.3, 99.2]	13	10.8 [5.8, 16.7]
GPT 5.6 Sol	L1	118	98.3 [95.8, 100.0]	89	74.2 [65.8, 81.7]
	L2	120	100.0 (120/120)	62	51.7 [42.5, 60.8]
	L3	119	99.2 [97.5, 100.0]	60	50.0 [40.8, 59.2]
Qwen 3.5 Plus	L1	102	85.0 [78.3, 91.7]	26	21.7 [14.2, 29.2]
	L2	95	79.2 [71.7, 86.7]	7	5.8 [1.7, 10.0]
	L3	102	85.0 [78.3, 90.8]	2	1.7 [0.0, 4.2]