Beyond Static Charts: Can Language and Vision–Language Models Generate Interactive Data Visualization Interfaces?

Anonymous ACL submission

Abstract

Data visualization is central to analytical reasoning, but real-world analysis increasingly requires language-driven interactive interfaces rather than static charts. Although recent large language and vision–language models (LLMs/VLMs) have shown promise in generating static charts from natural language, their ability to generate interactive data visualization interfaces remains largely unexplored due to the lack of benchmarks. We introduce VIS-GEN, a benchmark for evaluating how well LLMs/VLMs can generate interactive visualization interfaces from natural language queries. VIS-GEN comprises 3,042 samples covering diverse analytical intents, including data filtering, temporal analysis, and visualization editing, each paired with dataset metadata and natural language queries that are designed to reflect realistic, goal-driven data exploration scenarios. We benchmark 14 state-of-the-art open-source and closed-source LLMs/VLMs, revealing large performance gaps and frequent failures on queries involving implicit intent, multiple interaction alternatives, and complex editing operations, highlighting interactive interface generation as a key open challenge beyond static chart synthesis. To address this, we propose a structured multi-stage interface generation framework that decomposes the task into visualization design representation, generation of multiple interface candidates, constraint-aware critique, and self-refinement. This approach improves the best model's pass rate by 15.9 percentage points, demonstrating a practical path toward more reliable language-driven interactive visualization systems. We release VIS-GEN at < redacted >.

1 Introduction

Data visualization transforms raw data into visual representations that enable people to make data-driven decisions across domains such as finance, healthcare, and business intelligence (Hoque et al., 2022; Rahman et al., 2025b). As a core component of the data science workflow, visualizations are routinely used for exploratory data analysis, comparison, trend identification, and communication of findings (Rahman et al., 2025a). In practice, real-world analysis rarely relies on static charts alone. Instead, users heavily depend on interactive data visualization interfaces that allow them to select and filter subsets of data, adjust temporal ranges, and iteratively refine visual encodings to support exploration and sensemaking (Yi et al., 2007).

Despite their importance, building interactive visualization interfaces remains time-consuming and technically demanding, even for experts (Bach et al., 2023). Unlike static charts, which can often be specified concisely, interactive interfaces require substantial coding to coordinate data transformations, visual encodings, and interaction logic. This process demands expertise in data analysis and visualization frameworks such as D3 or Vega-Lite (Satyanarayan et al., 2016), posing a significant barrier for non-technical users (Ali et al., 2016; Waskom, 2021; Bisong and Bisong, 2019). While commercial tools such as Tableau and Power BI offer graphical interfaces, effective use still requires design and data transformation skills and provides limited support for flexible, iterative exploration. These limitations raise a fundamental question: *can interactive data visualization interfaces be generated directly from natural language?*

Recent advances in large language and vision–language models have shown promise in automating visualization-related tasks, particularly static chart generation from natural language queries (Chen et al., 2024; Rahman et al., 2025b). However, their ability to generate interactive visualization interfaces remains largely unexplored (Vaithilingam et al., 2024). This task is substantially more challenging than producing static charts, as it requires multi-step reasoning over analytical intent, data semantics, visualization design,

Figure 1: Examples of different query types and expected outputs in VIS-GEN. In both exploration and editing, the model is given a data table and a user query and generates an interactive visualization; in editing, the model is additionally provided with an initial visualization (in Vega-Lite form). Implicit queries describe the analytic goal without specifying operations or controls, whereas explicit queries state the intended operations in plain language.

and interaction constraints. Models must infer appropriate widgets, editing controls, and their coordination with visual encodings—capabilities that existing benchmarks do not capture.

To address these gaps, we introduce VIS-GEN, the first benchmark for evaluating how well vision–language models can generate interactive data visualization interfaces from natural language queries grounded in real datasets. VIS-GEN goes beyond chart generation by requiring models to infer interaction widgets for exploration (e.g., sliders, dropdowns, selectors), editing controls for redesign (e.g., chart type and encoding switches), and their coordination with visual representations. The benchmark comprises 3,042 samples spanning diverse analytical intents, including filtering, comparison, temporal analysis, and visualization editing (Figure 1). Importantly, the queries reflect realistic, goal-driven data exploration scenarios rather than explicit UI instructions, making widget and interface inference a non-trivial reasoning problem.

We benchmark 14 state-of-the-art open-source and closed-source LLMs/VLMs on VIS-GEN and observe substantial performance gaps, particularly for queries involving implicit intent, multiple interaction alternatives, and complex editing operations. Under the direct generation setting, performance remains limited despite near-perfect executability: the strongest closed-source model, Claude-4.5-Opus, achieves only a 41.50% final pass rate, while the best open-source model reaches 10.91%.

To move beyond single-pass generation, we introduce a structured multi-stage interface generation framework that decomposes the task into intent extraction, intermediate interface representation, candidate interface synthesis, constraint-aware critique, and validation-driven self-refinement, ultimately producing executable interfaces. This approach yields substantial gains, improving final pass rates by 25.1 and 15.9 percentage points for GPT-4o and Claude-4.5-Opus, respectively, and producing more reliable interactive visualizations.

In summary, our contributions are threefold: (1) **VIS-GEN**, the first benchmark for interactive visualization interface generation from natural language; (2) a **structured interface generation framework** that enables principled reasoning, critique, and selection of interaction widgets and editing controls; and (3) **extensive evaluations** of state-of-the-art LLMs/VLMs, uncovering key limitations and highlighting interface-level reasoning as a central challenge for future research in language-driven data visualization.

2 Related Work

Text-to-Visualization Benchmarks. Many existing text-to-visualization benchmarks frame the

tasks as natural language–to–SQL translation or direct mappings from text to visualization specifications (Zhong et al., 2017; Luo et al., 2021; Srinivasan et al., 2021; Liu et al., 2021; Chen et al., 2024). Benchmarks such as WikiSQL (Zhong et al., 2017) and nvBench (Luo et al., 2021) focus primarily on NL2SQL, implicitly assuming that visualization can be trivially derived from query results, while datasets like NLV-Utterance (Srinivasan et al., 2021) and ADVISor (Liu et al., 2021) map language to visualization specifications but restrict queries to explicit chart-type mentions and simple analytical intents. VisEval (Chen et al., 2024) moves toward evaluating visualization generation but remains limited by a small number of data tables. Finally, Text2Vis (Rahman et al., 2025b) improves chart coverage and reasoning complexity, yet still focuses exclusively on static chart generation. None of the existing benchmarks evaluate the generation of interactive visualization interfaces, which motivates our work.

LLMs for Automated Visualization. Hybrid methods such as RGVisNet (Song et al., 2022) and ADVISor (Liu et al., 2021) improved robustness by combining structured pipelines with learned components, while recent LLM-based systems demonstrate strong performance in generating executable visualization code directly from natural language (Hoque and Islam, 2024; Maddigan and Susnjak, 2023). For example, Chat2VIS (Maddigan and Susnjak, 2023) applies prompt engineering for chart generation, ChartLlama (Han et al., 2023) focuses on chart understanding via instruction tuning, and Text2Vis-RL (Rahman et al., 2025b) introduces a multi-objective reinforcement learning that jointly optimizes textual accuracy, code validity, and visualization quality (Rahman et al., 2026).

Generative UI. Recent advances in VLMs have spurred research on generative user interfaces, where models generate interface structures rather than content alone (Leviathan et al., 2025). Most prior work focuses on general-purpose UI generation, particularly website and application interfaces, using inputs such as natural language prompts (Chen et al., 2025; Leviathan et al., 2025), screenshots (Laurençon et al., 2024; Si et al., 2025), or sketches (Li et al., 2025), as well as hierarchical or task-model–driven UI synthesis (Gui et al., 2025; Cao et al., 2025). However, these approaches do not account for the data semantics, visual encodings, and analytical intent that fundamentally constrain data visualization interfaces. Work that explicitly targets interactive visualization remains sparse, including ambiguity-resolution widgets (Hoque et al., 2017), grammar-based autocompletion for visualization queries (Setlur et al., 2020), SQL-to-interactive visualization systems (Chen and Wu, 2022), and limited LLM-based visualization editing (Vaithilingam et al., 2024; Chen et al., 2025). No prior work provides a comprehensive benchmark or systematic evaluation of interactive visualization interface generation—a gap addressed by VIS-GEN (Tab. 5).

3 VIS-GEN

We develop VIS-GEN through a carefully designed benchmark construction pipeline that integrates realistic data tables, high-quality human- and LLM-authored queries, executable visualization specifications, and rich annotations, enabling systematic evaluation of both visual correctness and interface-level reasoning.

3.1 Data Collection & Preparation

To reflect realistic data exploration and visualization editing scenarios, we curate datasets that (i) span diverse real-world topics, (ii) naturally support multiple visualization types, and (iii) afford rich interaction and editing behaviors. We therefore construct the benchmark using data tables collected from multiple complementary public sources:

(*i*) **Our World in Data (OWID)** (ourworldindata.org) covering global health, economic, social, and demographic indicators with interactive temporal and filtering views. (*ii*) **Organisation for Economic Co-operation and Development (OECD)** (oecd.org) providing policy-oriented datasets that support comparison, aggregation, and heterogeneous indicators. (*iii*) **Tableau Public** (public.tableau.com) offering practitioner dashboards across business, public policy, and social trends. (*iv*) A small fraction of **synthetic tables** are generated using LLMs to augment structural and semantic diversity, covering rare but important cases underrepresented in real-world sources, such as atypical schemas and complex widget–encoding interactions.

In total, we collect 1,307 data tables across these sources. We then perform manual quality control to remove malformed, redundant, or trivial tables and retain datasets that meaningfully support interactive exploration and visualization editing. The final VIS-GEN collection comprises 966 data tables and is dominated by real-world data, with syn-

thetic tables used primarily to augment diversity. In addition to the data tables, we also curate rich visualization metadata for each instance, including visualization titles, associated widget controls, and interface screenshots, which capture real-world interaction patterns and directly inform the construction of our interaction taxonomy (see Figure 4).

3.2 Query Generation and Annotations

VIS-GEN supports two complementary interaction scenarios—*exploration* and *editing*—and covers over 40 interaction widgets to capture realistic analytical behavior. We first curate a seed set of 400 expert-authored queries, evenly split between exploration and editing. Exploration queries require the model to infer both the visualization and interaction strategy directly from data, while editing queries specify analytical refinements to an existing static Vega-Lite visualization.

We then expand this seed set using a multi-step, LLM-assisted refinement process involving GPT-4o, GPT-5.2, Gemini 3.5 Pro, and Claude Sonnet 4.5, producing an additional 2,700 candidates (Tab. 7). To maintain quality, candidates are generated using few-shot examples drawn from the human-annotated seed set and refined through a multi-stage process before review. Leveraging multiple models increases linguistic and structural diversity. All LLM-generated candidates then undergo manual review by two annotators. Annotators assess each query for realism, clarity, schema grounding, analytical intent, complexity, diversity, and correctness of the associated widget/interface annotations. Details of the human review and revision protocol are provided in Appendix A.6. The initial direct-accept agreement is 82.7%, with 467 cases flagged for further review. These are manually revised and cross-checked for consistency; candidates are discarded if either annotator rejects them, resulting in 58 removals. In total, 809 queries are either expert-authored or substantially human-revised, forming a significant human-curated subset of the benchmark. The final benchmark contains 3,042 high-quality samples, each paired with a supported widget list, an executable Vega-Lite visualization, and rich metadata, enabling systematic evaluation of both visual generation and interface-level reasoning.

3.3 Dataset Analysis

VIS-GEN is constructed primarily from real-world data (93.23%), with a small fraction of synthetic tables (6.77%) used to augment rare or underrepresented scenarios, ensuring strong ecological validity alongside controlled diversity (Figure 2). The datasets span diverse topics and schema structures (see Figures 5 and 6). On average, each table contains 4,133.70 rows (min: 3; max: 184,632) and 9.98 columns (min: 1; max: 167), reflecting both lightweight and large-scale analytical scenarios.

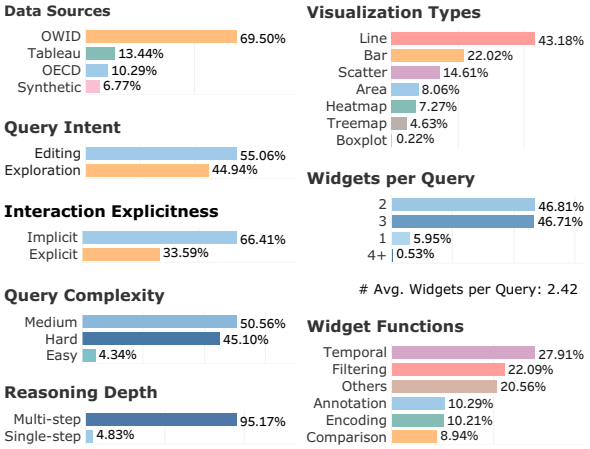


Figure 2: Summary of VIS-GEN dataset query statistics and diversity characteristics.

Analytical Difficulty. VIS-GEN strongly emphasizes non-trivial analytical reasoning. Most queries fall into medium or hard complexity categories, with only a small portion consisting of simple cases. The benchmark is overwhelmingly multi-step (95.17%), requiring models to reason across multiple analytical operations rather than perform direct lookups or single transformations. In addition, the natural language queries themselves are linguistically rich and complex, with a mean length of 196.46 characters reinforcing the benchmark's emphasis on realistic, high-cognitive analytical workflows.

Interaction Diversity. A defining characteristic of VIS-GEN is its rich and realistic interaction design space. The majority of queries (94.05%) require multiple coordinated widgets, mirroring real analytical workflows in which users iteratively filter, compare, and refine views. Temporal and filtering interactions are most prevalent, followed by substantial use of comparison, annotation, and encoding-related controls. Importantly, interaction intent is more often implicit (66.41%) than explicit, forcing models to infer appropriate interface elements from context alone—an essential capability for language-driven visualization systems.

Visualization Coverage. VIS-GEN spans a wide range of visualization types. Line (43.82%) and bar charts (22.02%) are most frequent, reflecting common analytical practice, while a meaning-

ful tail of less frequent but analytically important visual forms remains. This distribution ensures that models must generalize across different visualization idioms and reason about how visual encodings interact with data semantics and user intent, rather than overfitting to a narrow set of chart types.

4 Methodology

4.1 Task Definition

We define VIS-GEN as a web-based interactive visualization interface generation task that evaluates a model's ability to translate natural language analytical queries into executable visualization interfaces grounded in structured data. The dataset consists of $N = 3042$ examples, denoted as $\mathcal{D} = \{(t_i, q_i, s_i, \tau_i, v_i, w_i)\}_{i=1}^{N}$, where t_i is a data table, q_i is a natural language query, s_i is an optional base Vega-Lite specification, and $\tau_i \in \{\text{exploration}, \text{editing}\}$. Exploration tasks generate interfaces from scratch, whereas editing tasks augment existing visualizations. The model is tasked with generating a Vega-Lite visualization code v_i and a set of interaction widgets w_i from (t_i, q_i, s_i), where the widgets are bound to v_i to control visualization parameters. We adopt Vega-Lite because it provides a widely used declarative grammar with native support for interactive constructs (e.g., parameters and selections), enabling evaluation of language-driven interface reasoning without entangling low-level UI details. Together, (v_i, w_i) define an executable visualization interface.

4.2 Interface Generation Strategies

We evaluate two fundamentally different strategies for interactive interface synthesis: direct generation and a structured multi-stage pipeline with validation-driven refinement.

***(i)* Direct Generation.** In the direct setting, the model is prompted to generate the complete interactive interface in a single pass. Given the task inputs, the model jointly produces the visualization specification, interaction widgets, and bindings without any explicit intermediate structure or decomposition. This setting reflects common use of LLMs and serves as a strong baseline for evaluating whether models can internally coordinate analytical intent, visualization design, and interaction behavior.

***(ii)* Structured Multi-Stage Generation.** While direct generation is appealing for its simplicity, interactive interface design inherently involves multiple interdependent decisions and constraints. To address this, we decompose interface synthesis into meaningful stages, inspired by human-centered visualization design (Munzner, 2025), including design-space exploration, constraint-aware selection, and targeted repair (Figure 3). Given a dataset and query, the framework proceeds as follows:

(1) Structured Visualization Interface Representation (SVIR) The model first infers the underlying analytical intent expressed in the query (target variables, grouping/comparison, temporal scope, and required operations), along with the relevant data fields and interaction needs (A.2.3). The extracted intent is then formalized into a structured JSON representation (SVIR), which specifies required visual encodings (e.g., mark types), permissible interaction types (e.g., filtering, temporal navigation, ranking), and semantic constraints derived from the data schema. Overall, SVIR defines the complete interface design specification, capturing *what the interface must support* without committing to specific UI implementations.

(2) Candidate Interface Synthesis. The model generates multiple candidate interface designs that satisfy the SVIR defined in Step (1), each proposing a different combination of visual encodings and abstract interaction controls while respecting dataset-specific field and type constraints. By explicitly generating alternatives, this stage explores the design space rather than committing prematurely to a single solution.

(3) Candidate Critique and Selection. Each candidate interface is evaluated along four complementary dimensions: (i) *Widget Match*, which verifies that all required interaction controls are present; (ii) *Intent Coverage*, which measures how well the design satisfies the user's analytical objectives; (iii) *Visualization Fit*, which assesses the structural alignment between the chart design and the intended analytical task; and (iv) *Feasibility*, which checks schema consistency, field-type compatibility, encoding correctness, and whether the interface can render non-empty data on initial load. These criteria are combined using a weighted scoring function over these four dimensions, and the highest-scoring candidate is selected for realization, and candidates violating any hard constraint (missing required interactions, insufficient intent coverage) are discarded (see App. A.2). Among the remaining designs, the highest-scoring candidate is selected for realization, introducing explicit and transparent reasoning over design trade-offs that are typically implicit in end-to-end generation.

(4) Executable Interface Construction. The

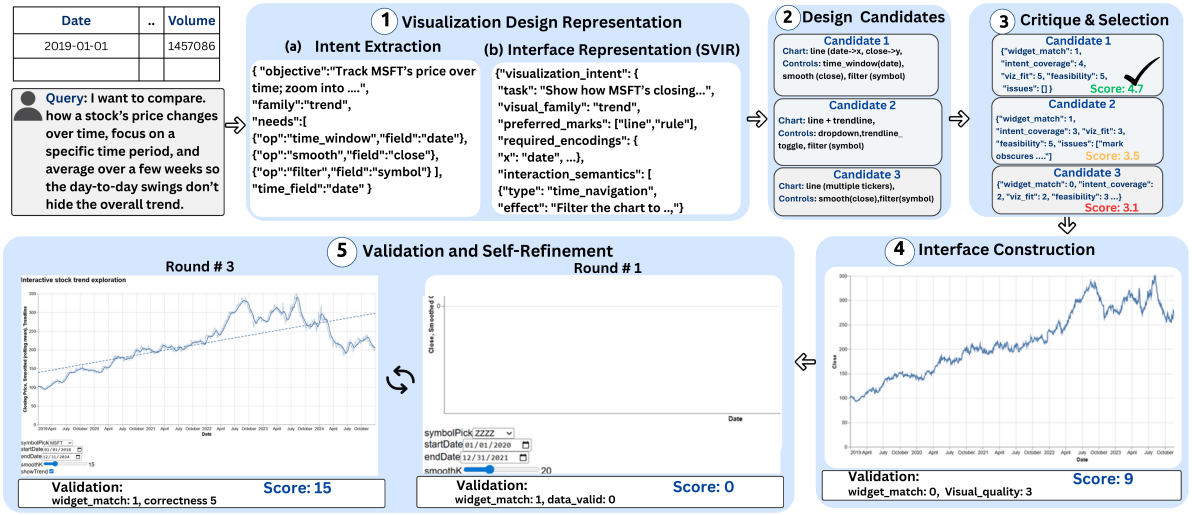


Figure 3: Structured multi-stage generation for interactive visualization interfaces: Given a dataset and query (and an optional base chart), the model derives a Structured Visualization Interface Representation, generates interface candidates, then selects the best design under constraints, and compiles it into an executable interface. Automated validation detects failures and triggers targeted self-refinement, yielding a functional interactive visualization.

selected design is translated into a complete interactive HTML interface. Abstract controls are instantiated as concrete UI elements (widgets) and bound to the Vega-Lite visualization through parameters and transforms.

(5) Validation and Self-Refinement. The generated interface is first subjected to hard validation gates requiring successful execution, presence of required controls, and non-empty data rendering. Each surviving interface is then scored on three criteria, correctness, readability, and visual quality (1-5 scale). The system iteratively refines the interface for up to three refinement rounds or until the scores saturate, and retains the best-scoring result.

5 Evaluation

5.1 Evaluation Criteria

We evaluate generated interactive visualization interfaces along complementary dimensions capturing both technical validity and analytical utility. The evaluation combines automated browser-based testing with VLM-based judgement for scalable and reliable assessment. Specifically, we measure: **(i) Executability**, by serving each generated HTML file through a local server and rendering it in a Chromium browser using Playwright (Microsoft, 2024), verifying successful page load, correct Vega-Lite rendering, and absence of fatal runtime errors; **(ii) Widget presence**, by checking whether the interface contains interactive controls that meaningfully support the analytical intent, accepting reasonable functional equivalents as correct when they satisfy the same intended interaction function.

We further evaluate: **(iii) Data validity**, verifying that charts display real plotted data rather than empty or broken visualizations; **(iv) Readability**, assessing labeling clarity and usability; **(v) Visual quality**, measuring aesthetic coherence and presentation; and **(vi) Correctness**, evaluating how well the interface and its interactions satisfy the analytical goal. Readability, visual quality, and correctness are rated on a 5-point ordinal scale.

An interface is marked **PASS** if executability$=1$,widget_match$=1$, data_valid$=1$, and readability, visual quality, and correctness are each ≥ 3. All VLM judgments use deterministic decoding (temperature $= 0$). GPT-4o serves as the primary judge for all models, with the exception of GPT-4o itself, whose generated outputs are evaluated by Gemini-2.5-Pro to avoid self-evaluation bias.

5.2 Models

We evaluate a diverse set of state-of-the-art models to establish strong baselines on VIS-GEN. Model selection prioritizes code generation ability, structured reasoning, multimodal understanding of tables and natural language, and deployment practicality. Closed-source models include GPT-4o (OpenAI, 2024), GPT-5-mini, o4-mini, Claude (4-Sonnet, 4.5-Opus) and Gemini 2.5 Pro (Team, 2024). For open-source baselines, we focus on models under 15B parameters, including Qwen2.5 (7B,14B,32B Instruct and Coder) and LLaMA-3.1-8B-Instruct (Grattafiori et al., 2024), CodeLlama-7B (Roziere et al., 2023), and Mistral-7B (Jiang et al., 2023).

Model	Code Executability	Widget Match	Data Validity	Readability	Visual Quality	Chart Correctness	Final Pass
Closed-Source Models							
GPT-4o	100	24.70	46.86	2.72	2.57	2.12	12.30
GPT-5-mini	100	**61.40**	36.10	3.53	3.16	2.62	28.30
O4-mini	99	36.50	26.89	2.97	2.81	1.95	14.10
Gemini-2.5-Pro	100	28.42	25.00	3.02	2.93	1.85	11.90
Claude-4-Sonnet	99	53.40	59.10	3.62	3.48	3.05	36.20
Claude-4.5-Opus	**100**	59.30	**64.40**	**3.78**	**3.68**	**3.30**	**41.50**
Open-Source Models							
Qwen-2.5-7B	100	19.60	33.52	2.16	2.07	1.70	7.20
Qwen-2.5-14B	99	24.25	24.56	2.49	2.36	1.64	9.05
Qwen-2.5-32B	100	25.96	31.49	2.82	2.70	1.86	10.91
Qwen2.5-Coder-7B	100	15.59	23.50	1.99	1.87	1.54	6.90
Qwen2.5-Coder-14B	99	20.25	24.81	2.30	2.18	1.61	8.24
LLaMA-3.1-8B	99	28.81	18.01	2.60	2.27	1.67	7.79
CodeLlama-7B	97	20.00	16.51	2.07	1.93	1.52	5.80
Mistral-7B	99	22.37	18.54	2.31	1.92	1.55	6.40

Table 1: Automatic evaluation results on VIS-GEN using **direct generation**. All values are reported in percentage (%) except Readability, Visual Quality, and Chart Correctness which are on a 5-point scale.

Model	Code Executability	Widget Match	Data Validity	Readability	Visual Quality	Chart Correctness	Final Pass
GPT-4o (Direct)	100	24.70	46.86	2.72	2.57	2.12	12.30
GPT-4o (Multi-Stage)	**100**	64.96	54.83	2.91	2.70	2.82	37.38
Claude-4.5-Opus (Direct)	**100**	59.30	64.40	3.78	3.68	3.30	41.50
Claude-4.5-Opus (Multi-Stage)	**100**	**67.52**	**77.15**	**3.89**	**3.86**	**3.66**	**57.40**
Ablation Study (GPT-4o)							
w/o Intent Extraction	100	42.48	54.50	2.78	2.62	2.55	22.78
w/o SVIR Representation	100	52.75	51.70	2.82	2.61	2.64	27.41
w/o Candidate Critique	100	54.90	51.58	2.84	2.73	2.64	26.13
w/o Self-Refinement	99	61.35	47.56	2.73	2.58	2.49	24.92
Single-Candidate Only	100	58.10	54.00	2.86	2.63	2.66	31.61
w/ Cross-Judge Selection	100	64.20	56.90	2.95	2.69	2.77	36.90

Table 2: Comparison of direct generation and the proposed multi-stage framework on VIS-GEN, with ablation results analyzing the contribution of each major component by selectively removing them.

Setting	Code Executability	Widget Match	Data Validity	Readability	Visual Quality	Chart Correctness	Final Pass	Preferred (%)
Direct	98.5	40.10	56.20	3.26	3.17	2.76	29.90	24.4
Multi-Stage	**99.3**	**71.30**	**69.10**	**3.62**	**3.42**	**3.39**	**50.20**	**75.6**

Table 3: Human evaluation on 1000 stratified samples (500 from GPT-4o and 500 from Claude) for both direct and multi-stage generation. Preferred (%) reflects annotator pairwise preference between the two settings.

5.3 Main Results

Direct Generation Results: Table 1 reports results under single-pass interface generation. Among closed-source models, the Claude family leads: **Claude-4.5-Opus** achieves the highest final pass rate (**41.50%**), followed by Claude-4-Sonnet (36.20%) and GPT-5-mini (28.30%). These models produce more reliable interfaces across widget matching, data validity, readability, and correctness. In contrast, open-source models perform worse. The strongest open-source model, Qwen-2.5-32B, reaches only 10.91% final pass rate, while most others remain below 8%. This result highlights the severe limitations of open models in generating interactive visualizations.

Across all systems, a consistent pattern emerges: models can usually generate syntactically valid charts but struggle to produce functional interactive interfaces. Despite near-perfect executability, failures often arise from incorrect or incomplete interaction behavior, indicating that the primary challenge lies in coordinating interaction logic with visual encodings, generating valid visualization code.

Multi-Stage Generation Results. We evaluate the structured multi-stage framework on the full VIS-GEN benchmark (Table 2), focusing on two representative extremes: **GPT-4o**, which performs poorly under direct generation, and **Claude-4.5-Opus**, the strongest direct baseline. For GPT-4o, multi-stage generation yields a dramatic improvement, increasing the final pass rate from 12.30% to 37.38%, with large gains in widget matching, data validity, and chart correctness and further boosts Opus from 41.50% to 57.40%. This contrast shows that interface reliability is governed more by the generation strategy than by the underlying model

alone. Notably, structured GPT-4o approaches the performance of Opus (direct), while structured Opus establishes the strongest overall results, underscoring the importance of explicit decomposition, constraint-aware critique, and validation-driven refinement for dependable language-driven interactive visualization (Fig. 8). Although we evaluate the multi-stage framework with GPT-4o and Claude-4.5-Opus, the framework is model-agnostic and can be applied to both closed-source and open-source models. Overall, performance drops most sharply on implicit, hard, multi-step, and multi-widget queries, indicating that failures are driven more by interface-level reasoning (see Appendix A.4 for further breakdowns).

5.4 Ablation Studies

Table 2 shows that the effectiveness of our multi-stage framework arises from the complementary contributions of its sequential stages, rather than any single step in isolation. The full pipeline achieves a 37.38% final pass rate, while removing individual stages consistently degrades performance to 22.78–31.61%, confirming that reliable interface generation benefits from structured decomposition over one-shot synthesis. The largest drops occur when intent extraction or the SVIR representation is removed, underscoring that explicitly modeling analytical goals and required interactions is critical. Removing candidate critique or self-refinement further degrades results. We additionally include a cross-judge variant, where Gemini-2.5-Pro performs Stage 3 selection and refinement evaluation; comparable performance shows that our structured rubric is robust to the choice of self- vs. cross-evaluator.

5.5 Human Evaluation

We conducted a human evaluation with two annotators on 1000 stratified samples (500 from GPT-4o and 500 from Claude 4.5 Opus), covering both direct and multi-stage generation. Annotators applied the same criteria as the automated evaluation and provided pairwise preferences (Fig. 15). As shown in Table 3, the multi-stage framework achieves large improvements over direct generation across widget matching, data validity, chart correctness, and final pass rate, and is strongly preferred by annotators (75.6% of cases). We also conducted a larger widget-reactivity audit on all PASS samples from 1000 multi-stage generations, since samples that already fail any required criterion remain non-PASS regardless of reactivity. Among 502 PASS samples, 493/502 (98.21%) exhibited correct widget–visualization linkage, confirming that interfaces satisfying the final criteria almost always support the expected interactions. Finally, we found strong correlation between human and model judgments for each metric, with high Pearson (r) and Spearman (ρ) coefficients ranging from 0.85 to 1.00 (see Table 9).

5.6 Error Analysis

We conducted a qualitative error analysis on 250 samples to identify key error patterns (see fig.9):

Missing or incorrect widgets. In direct generation, models often produce static charts, include too few controls, or add widgets that do not satisfy the query, causing failures despite plausible visuals.

Data validity failures. Some interfaces produce empty or incorrect views due to hallucinated field names, type mismatches (e.g., numeric vs. categorical), incorrect aggregation or filtering, or malformed parameter definitions, including incompatible assumptions across visualization grammars.

Visual and semantic mismatches. Some outputs produce readable charts but violate visualization design principles, e.g., using pie charts for temporal trends, resulting in low chart correctness.

Broken bindings and state updates. Even when correct widgets are present, interactions may fail to update the visualization state due to disconnected parameters, missing references between parameters and encodings or filters, or incorrect signal wiring, resulting in non-responsive interfaces.

6 Conclusion

We introduce **VIS-GEN**, the first large-scale benchmark for language-driven generation of interactive data visualization interfaces, moving beyond the static-chart focus of prior work. Our evaluation of open- and closed-source models reveals fundamental limitations in interface-level reasoning, particularly for implicit intent, coordinated interactions, and complex editing tasks. To address these challenges, we propose a structured multi-stage generation framework that decomposes interface synthesis into intent modeling, semantic representation, candidate generation and critique, and validation-driven refinement. This approach substantially improves the performance, enabling models to produce more usable and faithful interactive visualization interfaces. Together, VIS-GEN and our multi-stage framework establish a strong foundation for advancing interface-level reasoning in VLMs.

Limitations

While VIS-GEN provides the first benchmark for evaluating language-driven interactive visualization generation, it has several limitations. First, although the benchmark is constructed from diverse real-world and synthetic data, it represents a curated subset of visualization tasks and does not aim to exhaustively cover all analytical contexts. Second, while VIS-GEN supports a wide range of commonly used visualization types, it does not include some less common or more complex visual encodings (e.g., parallel coordinates or large multi-view dashboards). Finally, VIS-GEN is instantiated using Vega-Lite as the visualization grammar, a widely adopted declarative framework supported across both JavaScript and Python ecosystems (e.g., via Altair). The benchmark does not evaluate generation for other interactive visualization libraries, but emphasizes reasoning over analytical intent and interaction structure rather than low-level UI implementation. Similar evaluation principles could be extended to other visualization libraries with comparable interaction primitives (e.g., D3.js).

Ethical Considerations

All datasets used in this work are obtained from publicly available sources, including Our World in Data, OECD, and Tableau Public, or generated synthetically. The study focuses on evaluating the technical capabilities of vision–language models for interactive visualization generation and does not involve human subjects or user data. As such, we do not anticipate direct ethical risks associated with data collection or use. Nevertheless, like other generative systems, models evaluated in this work could potentially be misused to produce misleading or incorrect visualizations if deployed without appropriate human oversight. We encourage responsible deployment practices and careful validation in real-world applications. Finally, we used AI-based writing assistants only to improve the presentation of the paper.

References

Syed Mohd Ali, Noopur Gupta, Gopal Krishna Nayak, and Rakesh Kumar Lenka. 2016. Big data visualization: Tools and challenges. In *2016 2nd International conference on contemporary computing and informatics (IC3I)*, pages 656–660. IEEE.

Benjamin Bach, Mandy Keck, Fateme Rajabiyazdi, Tatiana Losev, Isabel Meirelles, Jason Dykes, Robert S Laramee, Mashael AlKadi, Christina Stoiber, Samuel Huron, et al. 2023. Challenges and opportunities in data visualization education: A call to action. *IEEE Transactions on visualization and computer graphics*, 30(1):649–660.

Ekaba Bisong and Ekaba Bisong. 2019. Matplotlib and seaborn. *Building machine learning and deep learning models on google cloud platform: A comprehensive guide for beginners*, pages 151–165.

Yining Cao, Peiling Jiang, and Haijun Xia. 2025. Generative and malleable user interfaces with generative and evolving task-driven data model. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–20.

Jiaqi Chen, Yanzhe Zhang, Yutong Zhang, Yijia Shao, and Diyi Yang. 2025. Generative interfaces for language models. *arXiv preprint arXiv:2508.19227*.

Nan Chen, Yuge Zhang, Jiahang Xu, Kan Ren, and Yuqing Yang. 2024. Viseval: A benchmark for data visualization in the era of large language models. *IEEE Transactions on Visualization and Computer Graphics*.

Yiru Chen and Eugene Wu. 2022. Pi2: End-to-end interactive visualization interface generation from queries. In *Proceedings of the 2022 International Conference on Management of Data*, pages 1711–1725.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yi Gui, Yao Wan, Zhen Li, Zhongyi Zhang, Dongping Chen, Hongyu Zhang, Yi Su, Bohua Chen, Xing Zhou, Wenbin Jiang, et al. 2025. Uicopilot: Automating ui synthesis via hierarchical code generation from webpage designs. In *Proceedings of the ACM on Web Conference 2025*, pages 1846–1855.

Yucheng Han, Chi Zhang, Xin Chen, Xu Yang, Zhibin Wang, Gang Yu, Bin Fu, and Hanwang Zhang. 2023. Chartllama: A multimodal llm for chart understanding and generation. *arXiv preprint arXiv:2311.16483*.

Enamul Hoque and M Saidul Islam. 2024. Natural language generation for visualizations: State of the art, challenges and future directions. In *Computer Graphics Forum*, page e15266. Wiley Online Library.

Enamul Hoque, Parsa Kavehzadeh, and Ahmed Masry. 2022. Chart question answering: State of the art and future directions. In *Computer Graphics Forum*, volume 41, pages 555–572. Wiley Online Library.

Enamul Hoque, Vidya Setlur, Melanie Tory, and Isaac Dykeman. 2017. Applying pragmatics principles for interaction with visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):309–318.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Hugo Laurençon, Léo Tronchon, and Victor Sanh. 2024. Unlocking the conversion of web screenshots into html code with the websight dataset. *arXiv preprint arXiv:2403.09029*.

Yaniv Leviathan, Dani Kalman, Matan Valevski, Danny Lumen, Eyal Molad, Eyal Segalis, Shlomi Pasternak, Vishnu Natchu, Valerie Nygaard, Srinivasan Cheenu Venkatachary Matias, and James Manyika. 2025. Generative ui: Llms are effective ui generators. *arXiv preprint*.

Ryan Li, Yanzhe Zhang, and Diyi Yang. 2025. Sketch2code: Evaluating vision-language models for interactive web design prototyping. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3921–3955.

Can Liu, Yun Han, Ruike Jiang, and Xiaoru Yuan. 2021. Advisor: Automatic visualization answer for natural-language question on tabular data. In *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*, pages 11–20. IEEE.

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

Paula Maddigan and Teo Susnjak. 2023. Chat2vis: Generating data visualizations via natural language using chatgpt, codex and gpt-3 large language models. *Ieee Access*, 11:45181–45193.

Microsoft. 2024. Playwright. `https://playwright.dev`. Accessed: 2026-01.

Tamara Munzner. 2025. Visualization analysis and design. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Courses*, pages 1–2.

OpenAI. 2024. Gpt-4 technical report.

Mizanur Rahman, Amran Bhuiyan, Mohammed Saidul Islam, Md Tahmid Rahman Laskar, Ridwan Mahbub, Ahmed Masry, Shafiq Joty, and Enamul Hoque. 2025a. Llm-based data science agents: A survey of capabilities, challenges, and future directions. *arXiv preprint arXiv:2510.04023*.

Mizanur Rahman, Mohammed Saidul Islam, Md Tahmid Rahman Laskar, Shafiq Joty, and Enamul Hoque. 2026. Aligning text, code, and vision: A multi-objective reinforcement learning framework for text-to-visualization. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6733–6750, Rabat, Morocco. Association for Computational Linguistics.

Mizanur Rahman, Md Tahmid Rahman Laskar, Shafiq Joty, and Enamul Hoque. 2025b. Text2vis: A challenging and diverse benchmark for generating multimodal visualizations from text. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 31837–31862.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2016. Vega-lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics*, 23(1):341–350.

Vidya Setlur, Enamul Hoque, Dae Hyun Kim, and Angel X Chang. 2020. Sneak pique: Exploring autocompletion as a data discovery scaffold for supporting visual analysis. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pages 966–978.

Chenglei Si, Yanzhe Zhang, Ryan Li, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. 2025. Design2code: Benchmarking multimodal code generation for automated front-end engineering. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3956–3974.

Yuanfeng Song, Xuefang Zhao, Raymond Chi-Wing Wong, and Di Jiang. 2022. Rgvisnet: A hybrid retrieval-generation neural framework towards automatic data visualization generation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1646–1655.

Arjun Srinivasan, Nikhila Nyapathy, Bongshin Lee, Steven M Drucker, and John Stasko. 2021. Collecting and characterizing natural language utterances for specifying data visualizations. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–10.

Gemini Team. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context.

Priyan Vaithilingam, Elena L Glassman, Jeevana Priya Inala, and Chenglong Wang. 2024. Dynavis: Dynamically synthesized ui widgets for visualization editing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pages 1–17.

Michael L Waskom. 2021. Seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021.

Ji Soo Yi, Youn ah Kang, John Stasko, and Julie A Jacko. 2007. Toward a deeper understanding of the role of interaction in information visualization. *IEEE transactions on visualization and computer graphics*, 13(6):1224–1231.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

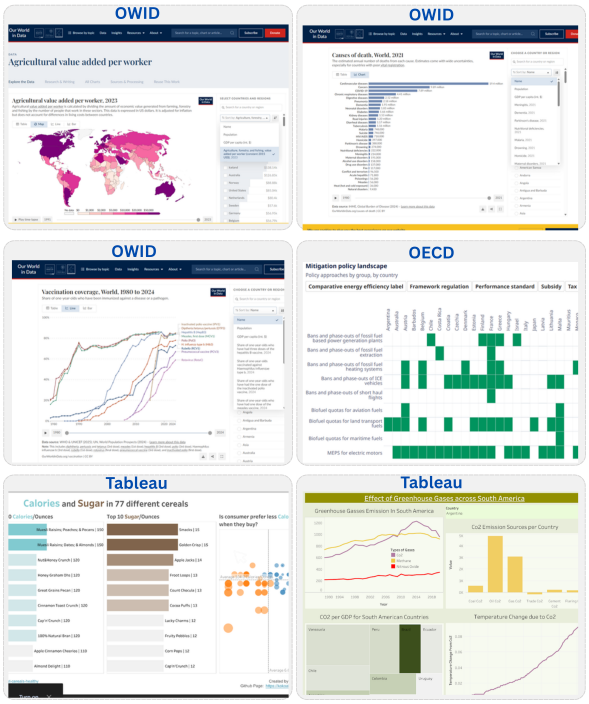


Figure 4: We collect real interactive visualizations from OUR WORLD IN DATA, OECD, and TABLEAU PUBLIC, extracting both the underlying data tables and rendered interface screenshots. From these, we identify the visual encodings and concrete interaction widgets used in practice, and construct natural-language queries that encode both analytical goals and required interactions, grounding VIS-GEN in realistic visualization workflows.

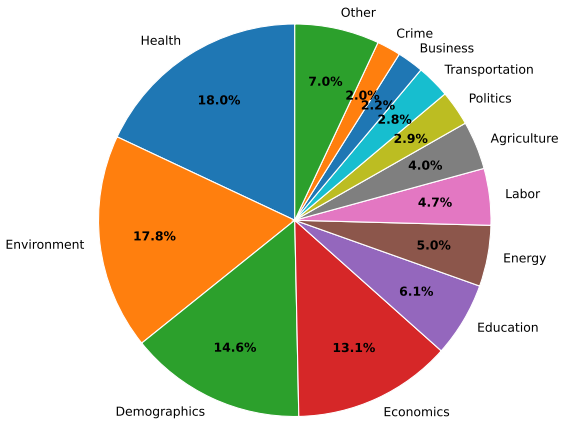


Figure 5: Topic distribution of VIS-GEN. The 12 most frequent domains are shown explicitly; remaining categories are aggregated as Other.

A Appendices

A.1 Dataset Collection

Table 6 shows the distribution of data tables from different sources in VIS-GEN.

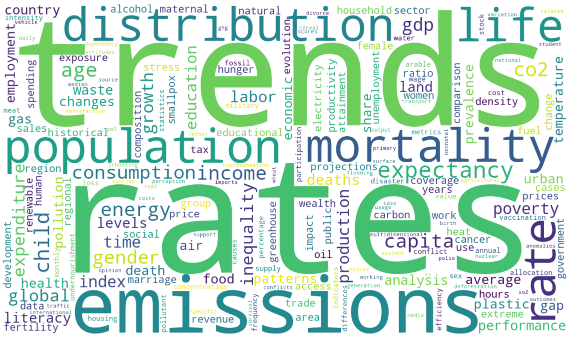


Figure 6: Word cloud illustrating the distribution of fine-grained analytical topics in VIS-GEN. Term size is proportional to frequency of occurrence.

A.2 Methodology

A.2.1 Candidate Selection Details

The candidate selection score is computed as:

$$\text{Score} = 0.3 \cdot \text{WidgetMatch} + 0.3 \cdot \text{IntentCoverage} + 0.2 \cdot \text{VizFit} + 0.2 \cdot \text{Feasibility}. \quad (1)$$

The weights are chosen to prioritize analytical intent and interaction completeness, as deficiencies in *WidgetMatch* and *IntentCoverage* constitute the dominant failure modes observed in preliminary analysis. Visualization fit and feasibility remain essential, but are treated as secondary once the core analytical requirements and interaction structure are satisfied. We further analyze the robustness of these weights through a sensitivity study on 250 stratified samples in Table 8.

A.2.2 Interaction Reactivity Validation

We initially implemented a Playwright-based programmatic interaction test to verify widget reactivity by modifying control values and checking whether the visualization updated accordingly. While this test works in most cases, it can undercount valid interfaces because generated outputs may use heterogeneous or non-standard widget implementations that are difficult to detect automatically. To complement this, we manually audited all PASS samples from 1,000 multi-stage generations. Among these, 502 samples satisfied the final PASS criterion, and 493/502 (98.21%) exhibited correct widget–visualization linkage. We also observe that broken linkage typically manifests as either invalid/empty rendered data or visibly broken/inconsistent visualizations, which are already captured by our data validity, readability, visual quality, and correctness checks. Therefore, for samples that already fail the final PASS criterion, additional reactivity testing does not change the overall pass outcome. These results suggest that our final PASS criterion provides a more reliable estimate of functional interactivity than a purely automatic Playwright reactivity test, which may falsely penalize valid interfaces due to missed or non-standard widgets.

A.2.3 Structured Visualization Interface Representation (SVIR)

We define the Structured Visualization Interface Representation (SVIR) as a structured JSON representation that captures the semantic requirements needed to generate an interactive visualization interface. Formally, an SVIR instance can be represented as:

$$\text{SVIR} = \{I, D, V, W, C\},$$

where I denotes the analytical intent, D denotes relevant data-schema information, V denotes visualization design requirements, W denotes interaction requirements, and C denotes interface-level constraints.

- **Analytical intent** (I): target variables, grouping or comparison dimensions, temporal scope, and required analytical operations.
- **Data schema** (D): relevant fields from the input table and their semantic or data types.
- **Visualization requirements** (V): required or preferred visual encodings, such as mark type and field-to-channel mappings.
- **Interaction requirements** (W): required interaction functions, such as filtering, temporal navigation, comparison, annotation, encoding modification, or editing.
- **Constraints** (C): validity conditions, including valid field usage, required interactions, schema compatibility, and non-empty executable rendering.

SVIR specifies the interface at a semantic level: it captures what the generated interface must support without committing to a particular concrete UI implementation. This allows the subsequent stages to generate multiple candidate interfaces, evaluate them under shared constraints, and compile the selected design into an executable visualization interface.

SVIR Schema

```
{
  "intent": {
    "target_variables": ["value"],
    "grouping_variables": ["country"],
    "temporal_scope": "year",
    "operations": ["compare", "filter", "trend"]
  },
  "data_schema": {
    "fields": [
      {"name": "year", "type": "temporal"},
      {"name": "country", "type": "nominal"},
      {"name": "value", "type": "quantitative"}
    ]
  },
  "visualization": {
    "mark": "line",
    "encodings": {
      "x": "year",
      "y": "value",
      "color": "country"
    }
  },
  "interactions": [
    {"function": "filtering", "target": "country"},
    {"function": "temporal_navigation", "target": "year"}
  ],
  "constraints": {
    "use_valid_fields_only": true,
    "requires_non_empty_rendering": true,
    "required_interactions": [
      "filtering",
      "temporal_navigation"
    ]
  }
}
```

Figure 7: **Illustrative SVIR schema.** SVIR represents the semantic requirements of an interactive visualization interface, including analytical intent, data schema, visualization design, interaction requirements, and feasibility constraints.

Illustrative SVIR Example. A formal specification and illustrative example of the SVIR schema are provided in Figure 7.

A.3 Cost Analysis

Direct generation requires a single model call per sample. In contrast, the multi-stage framework requires approximately six calls, depending on the number of generated candidates and refinement iterations, representing a bounded overhead. These calls are not uniform in cost: early stages such as intent extraction, SVIR construction, and critique produce short structured outputs, while most tokens are consumed by final HTML/Vega-Lite generation and refinement. Despite this overhead, the framework substantially improves reliability, increasing GPT-4o performance from 12.30% to 37.38% and Claude-4.5-Opus from 41.50% to 57.40%. This cost is acceptable for interactive UI generation, which is typically a latency-tolerant, high-value task where correctness and usability are more important than marginal token cost. Moreover, structured GPT-4o approaches the performance of direct Claude-4.5-Opus, suggesting that the framework can partly offset cost by enabling a cheaper base model to match a stronger single-pass baseline.

A.4 Additional Results

Further breakdowns of model performance by query type, complexity, reasoning depth, and number of widgets are provided in Table 4.

A.5 Human–Judge Agreement and Evaluator Robustness

Correlation with Human Ratings. We compare human ratings with LLM-judge scores on 1000 stratified samples and observe strong agreement across all evaluation metrics (Table 9), with consistently high Pearson (r) and Spearman (ρ) correlations.

Stability Across Runs. To assess robustness, we rerun the LLM judge four times on the 1,000 multi-stage samples and find that the final pass rate remains within ± 1 percentage point across runs, indicating stable and reproducible evaluator behavior.

A.6 Human Review and Revision Protocol for LLM-Generated Queries

To reduce artifacts from LLM-assisted dataset expansion, every LLM-generated candidate query underwent a mandatory human review and revision process before inclusion in VIS-GEN. The goal of this process was to ensure that each query was natural, analytically meaningful, grounded in the corresponding table schema, and paired with valid visualization and interaction annotations.

Candidate pool. Starting from the 400 expert-authored seed queries, we generated additional candidate queries using four heterogeneous LLMs: GPT-4o, GPT-5.2, Claude Sonnet 4.5, and Gemini 2.5 Pro. To encourage the generated queries to reflect the style, diversity, and analytical intent of the human-authored seed set, each generation prompt included three randomly selected seed examples as few-shot demonstrations. These examples were sampled from the initial human-written query pool and varied across task type, interaction explicitness, visualization type, and widget function.

Query generation followed a multi-step agentic process. First, a generator model produced a candidate query together with its intended visualization and interaction annotations. Then, a separate critique step evaluated whether the candidate satisfied the required quality criteria: schema grounding, analytical validity, interaction correctness, visualization compatibility, naturalness and realism, clarity and unambiguity, and contribution to dataset diversity. If the critique identified issues, the candidate was revised and checked again. This generate–critique–revise process was repeated for multiple rounds, and only the final candidate from this iterative process was passed to human review. This ensured that the initial candidate pool had already been filtered for basic quality before manual annotation.

Each candidate consisted of a natural language query, the associated table, the task setting, the intended visualization type, the required widget/interface annotations, and relevant metadata.

Independent human review. Each LLM-generated candidate was independently reviewed by two annotators with experience in data visualization and natural-language-based analytical tasks. Annotators examined the query together with the corresponding table schema, available fields, expected visualization type, widget labels, and interaction annotations. For editing tasks, annotators also reviewed the initial Vega-Lite visualization.

Annotators made a binary decision: ACCEPT if the candidate satisfied all review criteria, and REVISE/REJECT if one or more criteria failed. The seven review criteria were:

1. **Schema grounding.** The query must refer only to fields, values, or analytical concepts supported by the corresponding table.
2. **Analytical validity.** The query must express a meaningful visualization-oriented analytical goal, such as filtering, comparison, temporal exploration, ranking, annotation, aggregation, or visualization editing.
3. **Interaction correctness.** The annotated widgets must correctly support the analytical intent of the query.
4. **Visualization compatibility.** The expected chart type and interaction design must be feasible for the table schema and data types.
5. **Naturalness and realism.** The query should resemble a realistic user request for data exploration or visualization editing, rather than a synthetic or template-like instruction.
6. **Clarity and unambiguity.** The intended analytical operation must be clear enough for a model or human to determine the required visualization and interactions.
7. **Difficulty and diversity.** The candidate should contribute useful coverage across task type, interaction explicitness, reasoning depth, visualization type, and widget function, without being a near-duplicate of existing examples.

A candidate was accepted only when all seven criteria were satisfied. If any criterion failed, the candidate was marked for revision or rejection.

Revision and disagreement resolution. The initial direct-accept agreement between the two annotators was 82.7%. A case was marked for further review whenever either annotator indicated that

the candidate should be revised or rejected rather than directly accepted. This resulted in 467 cases requiring additional review.

All 467 cases requiring additional review were manually revisited by both annotators. During this stage, the annotators reviewed the query, table schema, visualization specification, and widget annotations. Each case was revised, edited, or updated to ensure that it satisfied the seven review criteria: schema grounding, analytical validity, interaction correctness, visualization compatibility, naturalness and realism, clarity and unambiguity, and difficulty/diversity. Revisions included clarifying ambiguous wording, correcting unsupported field references, updating widget annotations, adjusting visualization types, or rewriting the query when needed.

After this revision stage, the updated candidates were cross-checked again by both annotators and assigned a final pass/fail decision. In the second review pass, a candidate was removed if either annotator determined that it should still be rejected. This resulted in 58 discarded candidates. These cases were removed to maintain a conservative quality standard.

Final inclusion rule. A candidate was included in VIS-GEN only if it satisfied all of the following conditions:

1. The query was natural and analytically meaningful.
2. All referenced fields and values were grounded in the table.
3. The expected visualization was compatible with the task and data types.
4. The required widgets correctly supported the intended interaction.
5. The task labels and metadata were internally consistent.
6. The candidate was accepted by both annotators, either during the initial independent review or after disagreement resolution.

This conservative protocol ensures that VIS-GEN is not simply a collection of LLM-generated queries, but a human-validated benchmark for evaluating whether models can infer and implement interactive visualization interfaces from realistic analytical language.

A.7 Visualization Interface Generation Comparison

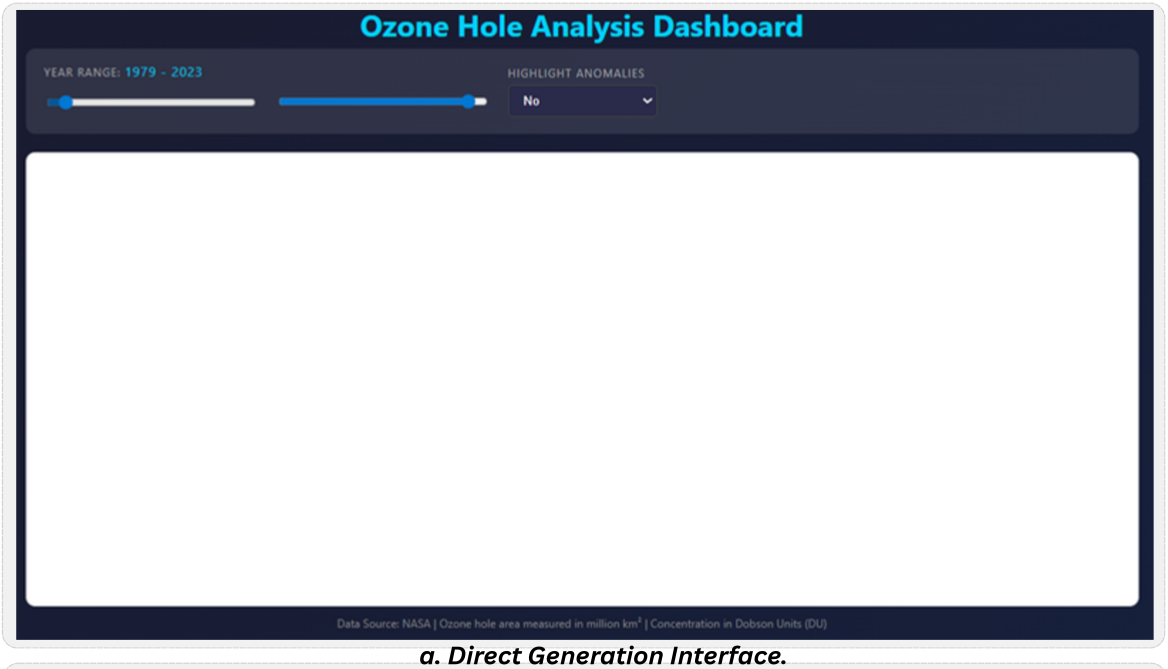
We provide a qualitative comparison between direct and multi-stage interface generation to illustrate the practical impact of the proposed framework. Figure 8 shows a representative example in which **Claude-4.5-Opus** under direct generation produces a visually complete interface that fails to render due to a data validation error, while the proposed multi-stage framework successfully recovers through structured critique, validation, and refinement, yielding a functional and semantically aligned interactive visualization.

A.8 Error analysis

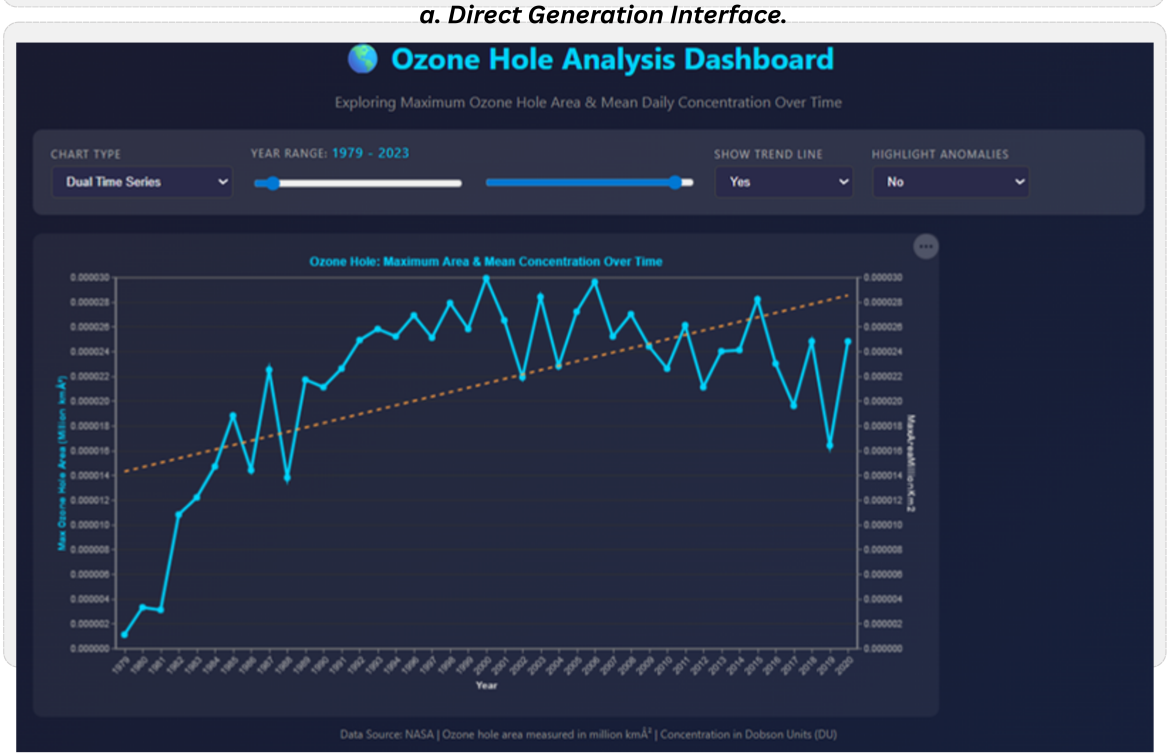
We conduct a qualitative error analysis to characterize the dominant failure modes in language-driven interactive visualization generation. Figure 9 summarizes representative error patterns observed across models, including data validity failures, visual and semantic mismatches, excessive interface complexity from dense widgets, broken bindings and state updates, missing or incorrect widgets, and silent rendering failures. These observations complement the quantitative results and highlight the key challenges that remain for reliable interface-level reasoning.

A.9 Prompt Design

We include representative sample prompts for the proposed multi-stage interactive generation framework in this section, and will release the full set of prompts, evaluation scripts, and implementation code as part of the public repository upon publication.



a. Direct Generation Interface.



b. Multi-stage Generation Interface.

Figure 8: **Visualization Interface Generation Comparison. (a) Direct Generation Interface** produced in a single-pass setting, in which the chart fails to render due to a data validation failure, despite the presence of a complete interface layout. **(b) Multi-stage Generation Interface** produced by the proposed framework, which iteratively refines candidate designs through structured validation, critique, and selection, yielding improved interaction completeness, execution reliability, and visual–semantic alignment.

Category	Widget Match		Data Validity		Chart Correctness		Final Pass Rate	
	Direct	Multi	Direct	Multi	Direct	Multi	Direct	Multi
GPT-4o								
Query intent								
Exploration	15.00	61.81	48.65	51.94	2.09	2.94	10.17	35.26
Editing	32.60	67.52	45.37	57.19	2.14	2.72	14.03	39.10
Query complexity								
Easy	31.82	71.21	56.06	55.30	2.40	2.92	18.18	39.39
Medium	40.05	68.40	44.99	53.97	2.20	2.83	18.99	37.71
Hard	6.78	60.50	48.03	55.76	1.99	2.80	4.23	36.81
Interaction explicitness								
Explicit	48.34	81.51	33.66	46.48	2.07	2.77	18.59	38.06
Implicit	12.72	56.58	53.51	59.06	2.15	2.85	9.11	37.03
# Widgets required								
Single (1)	56.35	78.45	70.17	77.90	3.10	3.15	47.51	50.83
Multi ($\geq$2)	22.68	64.10	45.37	53.37	2.06	2.80	10.10	36.53
Reasoning length								
Short	55.78	73.47	61.90	64.63	2.70	3.05	34.69	42.18
Long	23.11	64.53	46.08	54.34	2.09	2.81	11.20	37.13
Claude-4.5-Opus								
Query intent								
Exploration	54.94	60.00	69.70	86.10	3.48	3.91	43.70	55.96
Editing	62.87	73.67	60.00	69.85	3.15	3.45	39.04	58.57
Query complexity								
Easy	75.76	77.27	77.30	84.10	3.65	3.93	55.30	61.36
Medium	73.20	75.03	63.00	73.86	3.43	3.72	50.13	59.62
Hard	42.13	58.16	64.70	80.17	3.12	3.58	29.73	54.52
Interaction explicitness								
Explicit	79.06	82.40	58.50	70.74	3.40	3.82	50.10	63.89
Implicit	49.30	60.10	67.40	80.40	3.26	3.59	36.78	54.11
# Widgets required								
Single (1)	89.00	79.60	76.20	86.19	4.20	4.28	65.75	68.51
Multi ($\geq$2)	57.40	66.76	63.60	76.58	3.25	3.62	40.00	56.70
Reasoning length								
Short	85.00	83.57	76.20	88.44	3.82	4.07	58.50	66.67
Long	57.98	66.70	63.80	76.58	3.27	3.64	40.60	56.93

Table 4: Model performance breakdown by query dimensions under Direct and Multi-stage settings. Metrics: WM (Widget Match, %), DV (Data Valid, %), CC (Chart Correctness, 1–5), PASS (Pass Rate, %).

Method	Input	Output	Eval./Benchmark Size	Supports Widgets	Multi-Widget Coord.	Editing Tasks	Implicit Queries	Executable UI
PI2 (Chen and Wu, 2022)	SQL / example queries	Interactive interface	N/A	△	△	✗	✗	✓
DynaVis (Vaithilingam et al., 2024)	NL edit / widget commands	Widget-augmented visualization	24-user study	△	△	✓	✗	✓
VisEval (Chen et al., 2024)	NL + data	Static chart	2,524 queries	✗	✗	✗	△	✗
Text2Vis (Rahman et al., 2025b)	NL + data	Static chart + answer	1,985 samples	✗	✗	✗	✓	✗
VIS-GEN (Ours)	NL + data	Full interactive interface	3,042 samples	✓	✓	✓	✓	✓

Table 5: **Comparison of VIS-GEN with prior visualization generation systems and benchmarks.** ✓ = Yes, △ = Partial, ✗ = No.

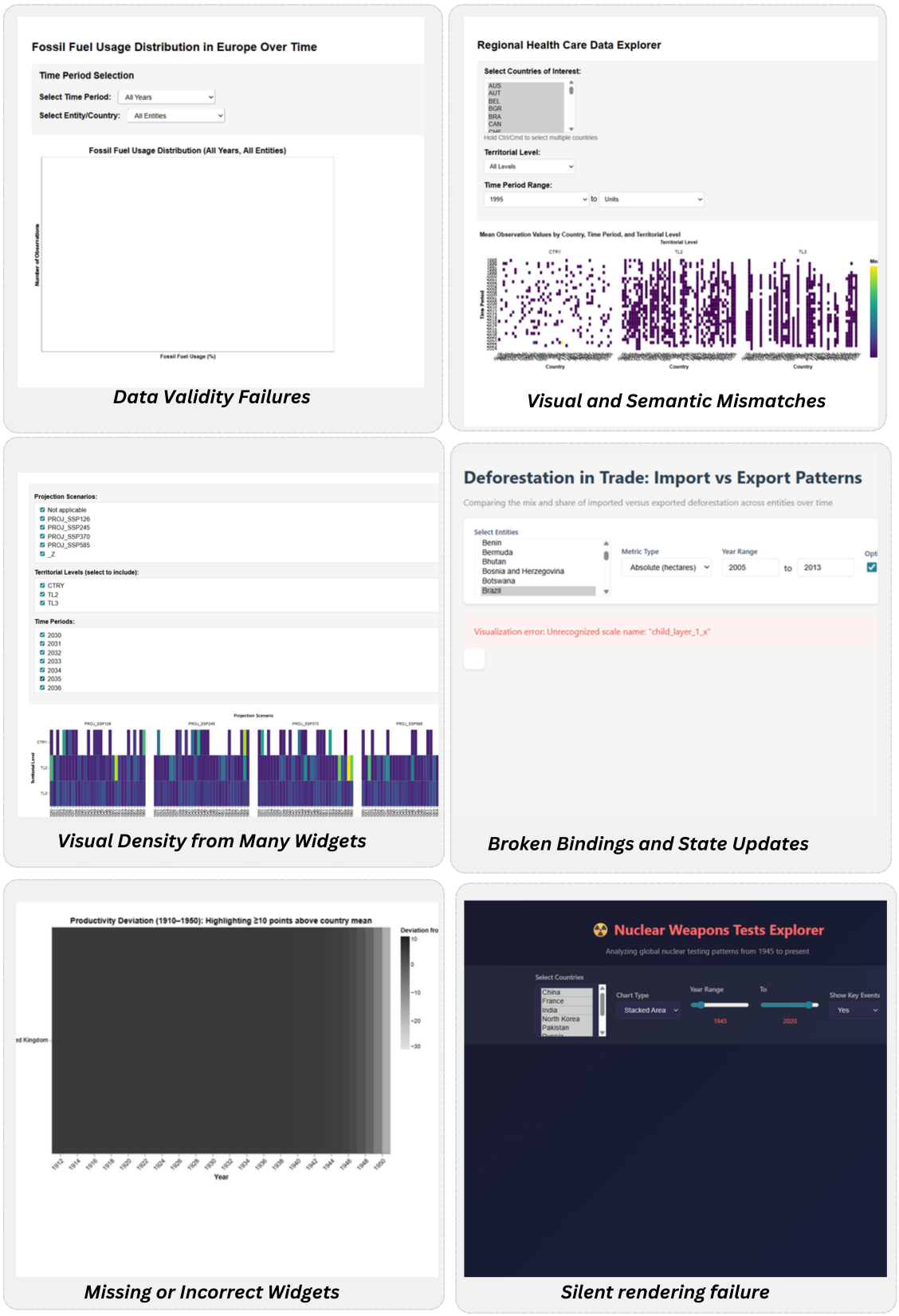


Figure 9: Representative failure modes in interactive visualization generation, including data validity failures, visual and semantic mismatches, visual density from excessive widgets, broken bindings and state updates, missing or incorrect widgets, and silent rendering failures.

Stage 1 (a): Intent Extraction Prompt

```
You are designing an interactive analytical visualization system.

Given:
(1) a dataset schema
(2) a natural language user query

Your job is to extract the analytical goals and the interactive controls required to support
the analysis.

Think in terms of interaction needs, not UI implementation.

Return only valid JSON in the following format:


{
  "objectives": [
    {
      "task": "<short analytical goal>",
      "priority": 1-5,
      "rationale": "<why this goal is necessary>"
    }
  ],

  "visual_analytic_form": {
    "family": "comparison|trend|distribution|
    relationship|composition|geospatial|
    ranking|anomaly|..."

  },

  "interaction_requirements": [
    {
      "operation": "filter|range|threshold|time_navigation|
      compare|sort|aggregate|search|
      focus|highlight|...."
,
      "field": "<dataset column>",
      "importance": 1-5,
      "rationale": "<why this control is required>"
    }
  ],

  "time_field": "<dataset column|null>"
}


Rules:
- Use ONLY dataset column names for fields.
- Do NOT mention widgets, UI, HTML, or implementation in the JSON.
- Include only controls that materially help answer the query.
- If the query implies time, set time_field.
```

Figure 10: **Stage 1(a) Prompt: Intent Extraction.** The system prompt used in the first stage of the proposed multi-stage framework, which extracts high-level analytical objectives and required interaction operations from a dataset schema and natural language query.

Stage 1(b): SVIR Construction Prompt

```
Convert the inferred intents into SVIR (Structured Visualization Interface Representation)
that separates semantics from implementation.

If an EXISTING_VEGA_LITE_SPEC block was provided before this prompt:
- Treat it as the current implementation that the user wants to edit or refine.
- Align visualization_intent and constraints with that spec when reasonable.

Return only valid JSON in the following format:


{
  "visualization_intent": {
    "task": "<one sentence summary of what the visualization must show>",
    "visual_family": "<must exactly match Stage1.visual_analytic_form.family>",
    "preferred_marks": ["point|bar|line|area|rect|text|circle|square|rule"|.."],
    "required_encodings": {
      "x": "<field|null>",
      "y": "<field|null>",
      "color": "<field|null>",
      "size": "<field|null>",
      "shape": "<field|null>",
      "facet": "<field|null>"
    },
    "encoding_rationale": "<why these specific encodings best serve the task>"
  },

  "exploration_intents": [
    {
      "type": "range_filter|category_filter|boolean_filter|threshold_filter|
      time_navigation|search|sort|aggregate|compare|focus|
      drilldown|rollup|highlight|navigate|...",
      "field": "<dataset column>",
      "importance": 1-5,
      "operation_semantics": "<what analytical question does this enable>",
      "expected_behavior": "<what should happen when the user interacts>"
    }
  ],

  "constraints": {
    "field_types": {
      "<field>": "quantitative|temporal|nominal|ordinal|text|
      boolean|id|geojson"
    },
    "encoding_constraints": {
      "x_type": "quantitative|temporal|nominal|ordinal|null",
      "y_type": "quantitative|temporal|nominal|ordinal|null",
      "color_type": "quantitative|nominal|ordinal|null"
    }
  }
}


Rules:
- Stay semantic; do not use UI or widget terminology.
- Use only dataset column names for fields.
- Use temporal fields only for time_navigation.
- Ensure required_encodings respect field_types.
```

Figure 11: **Stage 1(b) Prompt: SVIR Construction.** The system prompt used to translate extracted analytical intent into a structured semantic specification that constrains visualization design and interaction behavior before interface synthesis.

Stage 2 Prompt: Candidate Interface Generation

```
You are designing alternative interactive visualization interfaces.

Your job is to propose multiple interface designs that faithfully implement the semantic
requirements already defined in SVIR.
Do NOT invent new requirements.

Return only valid JSON in the following format:


{
  "candidates": [
    {
      "id": "C1",
      "chart": {
        "mark": "<one of: point|bar|line|area|rect|tick|circle|square|rule|..>",
        "encodings": {
          "x": "<field|null>",
          "y": "<field|null>",
          "color": "<field|null>",
          "facet": "<field|null>"
        },
        "description": "<short explanation of the design>"
      },
      "controls": [
        {
          "control_type": "range|time|category|boolean|threshold|
          search|sort|aggregate|compare|focus|
          drilldown|rollup|highlight|navigate}...",
          "field": "<dataset column>",
          "importance": 1-5,
          "notes": "<short>"
        }
      ]
    }
  ]
}


Requirements:
- Generate exactly four candidates.
- Each candidate must implement all required_encodings from SVIR.visualization_intent.
- Each candidate must implement all SVIR.exploration_intents.

Controls:
- Use only the interactions specified in SVIR.exploration_intents and Stage-1.
- Do not introduce new interaction types.
- Map each exploration_intent to exactly one control.
- For any exploration_intent with importance at least 4, the control must appear in every
candidate.

Diversity:
- All four candidates must be structurally different.
- No two candidates may share the same chart and control configuration.

Safety and validity:
- Use only dataset columns for fields.
- Ensure encodings respect field types from SVIR.constraints.
- Do not produce redundant controls.
- Do not include UI or widget names; controls are abstract.

The goal is to explore multiple high-quality interface designs while strictly respecting the
intent and interaction requirements.
```

Figure 12: **Stage 2 Prompt: Candidate Interface Generation.** The system prompt used to generate multiple structurally diverse candidate interfaces that satisfy the semantic intent and interaction constraints defined in the previous stages.

Stage 3 Prompt: Candidate Critique and Evaluation

```
Evaluate each candidate using the following criteria:

1) widget_match:  1 if the candidate includes all interaction_requirements from Stage-1 and
all exploration_intents from SVIR, else 0
2) intent_coverage (1-5):  weighted by objectives priority
3) viz_fit (1-5):  how well chart structure matches visualization_intent.visual_family from
SVIR
4) feasibility (1-5):  field type compatibility, schema validity, encoding correctness

Return only valid JSON in the following format:


{
  "evaluations": [
    {
      "id": "C1",
      "widget_match": 0 or 1,
      "intent_coverage": 1-5,
      "viz_fit": 1-5,
      "feasibility": 1-5,
      "issues": ["<short>", "..."]
    }
  ]
}


Rules:
- Reject candidates with missing required encodings or invalid field types.
```

Figure 13: **Stage 3 Prompt: Candidate Critique and Evaluation.** The system prompt used to assess candidate interfaces with respect to intent fulfillment, interaction coverage, visualization fit, and execution feasibility under strict rejection constraints.

Stage 4 Prompt: HTML Generation

```
You are a data visualization system.

Your task is to generate a complete, self-contained HTML file that implements the FINAL_PLAN
as a correct and interactive Vega-Lite visualization.

Core constraints:
- Use correct data types:  temporal only for true dates, quantitative for numeric, nominal or
ordinal for categorical.
- Implement the FINAL_PLAN exactly:  chart encodings, mark type, and all specified controls.
- The dataset must be loaded using the provided dataset path only.
- The system must be robust under interaction and handle rendering errors.
- Output only the final HTML document with no explanations.
...

Reference structure:
Follow the general structure of the provided example HTML.
...

Task context:
TASK TYPE: {QUERY_TYPE}
USER QUERY: {QUERY}
DATASET FILE: {DATASET_PATH}
COLUMNS: {COLUMN_LIST}
FINAL_PLAN (JSON): {FINAL_PLAN}
```

Figure 14: **Stage 4 Prompt: HTML Generation.** The system prompt used to synthesize a complete interactive HTML visualization that faithfully implements the final structured design plan.

Evaluation Scoring Rubric

```
Task 1 - Widget Match (Binary)
Decide whether the required widgets or reasonable functional equivalents appear in the
interface and meaningfully support the analytical goal stated in the query.
Score:
- 1:  All required widgets (or clearly equivalent interactive mechanisms) are present and
correctly support the task.
- 0:  One or more required widgets are missing, ineffective, or unrelated to the analytical
goal.

Task 2 - Data Validity (Binary)
Decide whether the visualization displays valid plotted data.
Score:
- 1:  The chart shows actual data marks (e.g., bars, points, lines) and responds correctly to
interactions.
- 0:  The chart is empty, broken, shows only axes, or displays error placeholders.

Task 3 - Readability (1-5)
Evaluate how easily a human can understand and use the interface.
Score:
- 1:  Unreadable; cluttered, confusing, mislabeled, or unusable.
- 2:  Major readability problems; difficult to follow with frequent confusion.
- 3:  Acceptable; understandable with some effort.
- 4:  Clear and well-organized; easy to follow.
- 5:  Exceptionally clear; intuitive layout and labeling.

Task 4 - Visual Quality (1-5)
Evaluate the aesthetic quality and professional presentation of the interface.
Score:
- 1:  Very poor; messy, misaligned, or visually broken.
- 2:  Weak design; noticeable visual issues.
- 3:  Adequate; visually acceptable.
- 4:  High quality; clean and polished.
- 5:  Excellent; publication-ready professional design.

Task 5 - Correctness (1-5)
Evaluate whether the visualization and interactions correctly address the user's query.
Score:
- 1:  Does not answer the query at all.
- 2:  Barely related; mostly incorrect or misleading.
- 3:  Partially answers the query.
- 4:  Mostly correct and useful.
- 5:  Fully answers the query accurately and robustly.
```

Figure 15: **Evaluation Rubric.** Scoring criteria to assess interaction completeness, data validity, readability, visual quality, and analytical correctness of generated visualization interfaces.

Source	**Count (%)**
Our World in Data (OWID)	710 (73.50%)
Tableau Public	107 (11.08%)
OECD	94 (9.73%)
Synthetic	55 (5.69%)
Total	**966 (100.00%)**

Table 6: Distribution of data tables from different sources in VIS-GEN.

Model	**Count (%)**
GPT-4o	900 (33.33%)
GPT-5.2	600 (22.22%)
Claude Sonnet 4.5	600 (22.22%)
Gemini 2.5 Pro	600 (22.22%)
Total	**2,700 (100.00%)**

Table 7: Distribution of LLM-generated candidate queries used for dataset expansion.

Setting	Widget	Intent	VizFit	Feas.	Final Pass
Default	0.30	0.30	0.20	0.20	37.6
Widget-heavy	0.40	0.25	0.20	0.15	39.6
Intent-heavy	0.25	0.40	0.20	0.15	36.4
Viz-heavy	0.25	0.25	0.35	0.15	35.2
Feasibility-heavy	0.25	0.25	0.15	0.35	35.6
Uniform	0.25	0.25	0.25	0.25	37.2

Table 8: Sensitivity analysis of candidate selection weights in Stage 3, based on 250 stratified samples.

Metric	**Pearson (r)**	**Spearman (ρ)**
Executability	1.00	1.00
Widget Match	0.92	0.93
Data Validity	0.95	0.94
Readability	0.86	0.88
Visual Quality	0.91	0.91
Chart Correctness	0.85	0.87

Table 9: Correlation between human ratings and LLM-judge scores on 2000 stratified samples.