OMNIFIGURE: UNIFYING REPRESENTATION FOR DATA VISUALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reusing a visualization example requires understanding whether its analysis fits the new data. Similar-looking tables may support different analyses, while changes in storage layout can obscure useful examples. To address both difficulties, retrieval needs a shared representation of what an analysis requires and what the new data support, independent of their storage layout. We therefore develop OmniFigure, combining OmniGrammar with a library of visualization examples. OmniGrammar matches case requirements to the observation structure of new data; retrieved figures and programs guide models and coding agents. To evaluate this reuse, we introduce OmniFigureBench, covering retrieval under representation changes and relation loss, together with intent-aligned generation. Experiments show improved recovery of applicable examples and rejection of incompatible ones, with generation gains for most systems on MatPlotBench, PlotCraft, and OmniFigureBench. Our code and data are available at `https://anonymous.4open.science/r/OmniFigure-7CEA/` and `https://anonymous.4open.science/r/OmniFigure-Bench-EF4E/`.

1 INTRODUCTION

Visualization systems increasingly turn a user's goal and data into plotting code. Execution and visual feedback can help revise that code (Yang et al., 2024; Ni et al., 2026), but complex tasks remain challenging (Zhang et al., 2026b). Human designers often turn to existing figures for design ideas and implementation guidance (Bako et al., 2023). A useful example brings together a figure, the data behind it, and the code that produced it. Reusing that experience with new data requires understanding the analysis the example represents.

Automated systems also draw on prior work. Retrieved queries can serve as prototypes for new visualizations (Song et al., 2022), and selected examples can guide a language model through in-context learning (Li et al., 2025). RAGvis stores analytical operations from exploratory data analysis (EDA) notebooks in a knowledge graph and aligns them with new data (Helali et al., 2025). ReSpark adapts the analytical objectives and data transformations of an earlier report (Tian et al., 2025). These approaches reuse different kinds of prior material. Our question is how to describe a visualization case and a new task in a common form that reveals when the case can help.

Consider a reference figure that connects each subject's measurements across two visits. Applying it to new data creates two possible errors:

Unsupported reuse: same fields, different subjects. A within-subject line represents a person's change across visits (Weissgerber et al., 2016). A new dataset can still contain `subject`, `visit`, and `response`, while the two visits include different people. Reusing the paired plot would claim individual changes that were never observed.

Missed reuse: different tables, the same subjects. The same repeated measurements can be stored in a wide table with one column per visit or a long table with one row per subject–visit pair (Wickham, 2014). The layout and code change, but the observations and their pairing remain. A comparison tied to the original columns could overlook a useful example.

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.

Both cases turn on which observations belong to the same subject (Figure 1). A common representation must preserve that relationship when comparing analyses, while allowing it to be expressed in either table layout.

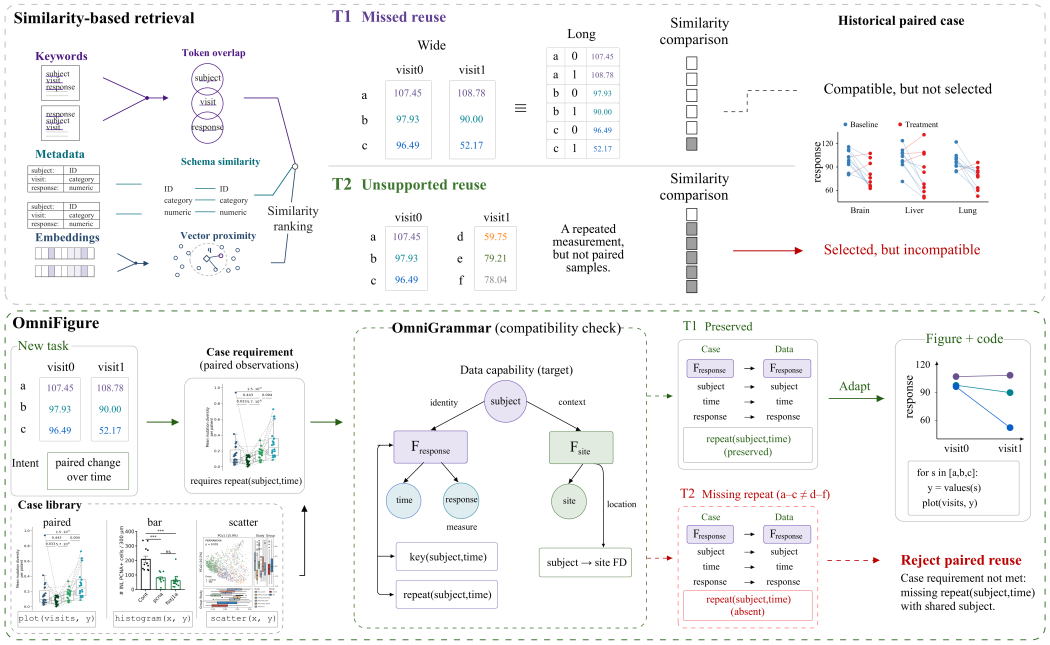

Figure 1: Visualization reuse with OmniFigure. Top: similarity-based retrieval can miss equivalent layouts or select incompatible cases. Bottom: OmniGrammar matches case requirements to data structure, preserving valid reuse and rejecting unsupported pairing.

OmniFigure describes stored visualization cases and new tasks using **OmniGrammar**, a shared schema built around a type hierarchy. OmniGrammar can mark repeated observations separately from independent ones, a distinction needed by the paired-line example above. For a stored case, annotation draws on the data and code, a graphics-object summary, and curator notes; for a new task, on the data and the user's request. OmniFigure compares the resulting schemas through the type hierarchy and gives retrieved figures and programs to the generator as references.

OmniFigureBench evaluates this reuse through representation consistency (T1), relation sensitivity (T2), and intent-aligned generation (T3). Curators check each source case for reproducibility and visual complexity.

Our contributions are:

- We develop OmniFigure based on OmniGrammar, a representation that enables analyses to transfer across datasets by matching their requirements to new data. OmniFigure retrieves compatible figures and programs to guide models and coding agents.
- We introduce OmniFigureBench with executable visualization cases to evaluate these retrieval decisions and their effect on generation from a stated intent.
- We establish that observation structure is central to transferring visualization experience across datasets. Matching this structure improves both example retrieval and downstream generation across models and agents.

2 RELATED WORK

Visualization Experience Retrieval and Reuse. Prior work shows that designers actively search for existing examples to obtain inspiration. The number, diversity, and similarity of examples can also affect the resulting designs (Bako et al., 2023; 2025). Several systems have explored reuse

of historical experience at different granularities. Retrieve-Then-Adapt retrieves adaptable designs from existing infographics (Qian et al., 2021); RGVisNet retrieves historical visualization queries as generation prototypes (Song et al., 2022); Prompt4Vis mines examples to select in-context demonstrations for LLMs (Li et al., 2025); RAGvis retrieves analytical operations from a knowledge graph built from historical EDA notebooks (Helali et al., 2025); and ReSpark uses analysis goals and transformation processes from existing data reports to guide new report generation (Tian et al., 2025). StructGraphics, Mystique, and VAID further make previous designs easier to represent and reuse through reusable graphical structures, chart-layout decomposition, and visual-analytics design indexing, respectively (Tsandilas, 2021; Chen et al., 2024; Ying et al., 2024). OmniFigure builds on this line of work by representing library cases and new data with schemas used for example retrieval.

Structural Semantics for Data and Visualization. Data management has long studied how to identify structural correspondences beyond field names and instance similarity. Schema matching research systematically considers schema-level, instance-level, structure-level, and constraint-based evidence (Rahm & Bernstein, 2001). Cupid jointly uses linguistic information, data types, constraints, and schema structure (Madhavan et al., 2001), while SANTOS shows that semantic relationships between columns provide structural signals beyond independent column similarity (Khatiwada et al., 2023). OmniGrammar extends this structural view to visualization by modeling the data relations required to instantiate a plotting program.

LLM-Based and Agentic Visualization Generation. Automatic visualization generation has progressed from single-pass program synthesis toward models and agents capable of execution, verification, and iterative refinement. PlotCoder studies visualization code generation from natural language and program context (Chen et al., 2021). MatPlotAgent introduces code execution, iterative debugging, and visual feedback (Yang et al., 2024). nvAgent and CoDA further use multi-agent collaboration for data understanding, planning, code generation, and output validation (Ouyang et al., 2025; Chen et al., 2026). VisCoder2 extends execution–rendering–self-correction to multiple visualization languages (Ni et al., 2026), while PlotCraft targets more complex and interactive visualization tasks with multi-round refinement (Zhang et al., 2026b). OmniFigure supplies these systems with examples retrieved through annotated data schemas. OmniFigureBench examines retrieval under data changes and the use of retrieved examples in downstream generation.

3 OMNIFIGURE

Matching field names and types cannot tell whether measurements from two visits belong to the same subjects. Conversely, reshaping those measurements from wide to long form leaves their pairing intact. OmniFigure represents the observations and relations required by a historical analysis in OmniGrammar. It describes the new data in the same language and checks whether they support those requirements. Compatible figures and programs then become references for generation.

3.1 OMNIGRAMMAR REPRESENTATION

OmniGrammar represents data as Variables connected through Facts. Variables distinguish Dimensions, such as patient identifiers, from Measures, such as treatment times. Each also records a semantic concept and a value kind: a patient identifier has concept `bio.patient` and kind `category`. A Fact specifies which Variables are observed together.

Consider a patient timeline, where treatment intervals and scan responses are drawn along each patient's row (Appendix Figure 4). A treatment Fact links the patient, treatment, interval endpoints, scan time, and response within each record. Each occurrence of a Variable in a Fact is an incidence. Roles distinguish the start and end incidences even though both refer to timepoints.

Constraints record requirements such as unique keys and repeated observations. Derivations specify how one set of Facts produces another. In the same example, taking the latest treatment end time for each patient produces a patient-summary Fact, used to draw the background line. A key on this summary specifies one record per patient. The treatment and summary Facts share the patient Variable, connecting observations at the two grains.

Bindings specify how physical columns and other storage layouts supply these logical records. After a column rename or reshape, the recovered structure can still contain the same observations and

relations. When the bindings and annotations preserve this structure, the case remains compatible. T1 tests this representation consistency (Section 4). Appendix A.1 gives the full representation and element definitions.

3.2 Library and Query Annotation

A historical case consists of data D_i, a plotting program p_i, and a figure F_i:

$$c_i = (D_i, p_i, F_i). \tag{1}$$

The program shows which fields and transformations contribute to the figure. OmniFigure annotates the case using the data and code, a summary of the graphics object, and curator notes when available. The resulting requirement pattern C_i retains the Facts needed by the analysis; optional Facts describe additional structure that can be used when available.

A new task provides an analysis request x and data D, but no completed program or figure. The request specifies which variables and relationships the figure should emphasize; query annotation records them and their supporting structure from the data and supplied metadata. T3 evaluates whether the generated figure follows this focus. For model-annotated cases and queries, the two routes are

$$\begin{aligned} S_i &= A_{\mathrm{lib}}(D_i, p_i, Z_i, N_i), \\ S_q &= A_{\mathrm{query}}(x, D), \end{aligned} \tag{2}$$

where Z_i is the graphics-object summary and N_i contains curator notes. The case pattern is formed from S_i; the query annotation describes the structure available for matching.

3.3 Data–Program Compatibility and Retrieval

Matching asks whether the new data can supply all the required structure of a case. For a fixed request x, let R_i collect the mandatory Facts of case c_i and their derivation dependencies, and write $\widehat{C}_i = C_i[R_i]$. Let $A(I_D)$ denote the available structure described by the query annotation, including materialized outputs of declared deterministic derivations. Compatibility is

$$\mathrm{Compat}(D, c_i) = \mathbb{1}\left[\exists \phi : \widehat{C}_i \hookrightarrow A(I_D)\right], \tag{3}$$

where ϕ is a single injective mapping of the required Facts, Variables, and incidences. It preserves their connections, shared references, and roles, and satisfies the required constraints and derivations. The new data may contain additional Facts that the case does not use.

The matcher first finds candidate Variable correspondences from value kinds and semantic concepts. The concept hierarchy allows a data Variable annotated as patient to satisfy a broader subject requirement. The matcher then searches for a consistent mapping across the requirement. In the timeline example, the patient Variable shared by the treatment and summary Facts must map to the same Variable in the new data. Two local matches that assign it to different Variables cannot form a global witness. When a required relation is lost, the case is rejected unless another global mapping satisfies all its requirements. T2 tests this relation sensitivity. Appendix A.2 specifies the checks and Appendix B gives the search procedure.

Only compatible cases enter the ranking. Exact matches precede semantic-parent matches. Within a tier, cases are ranked by their minimum required semantic distance; ties prefer greater joint optional coverage among witnesses attaining that distance, then a fixed case identifier. The first k cases form the reference set $\mathcal{E}_k(D)$. Appendix B defines the distance, joint optional coverage, and deterministic tie-breaking.

Appendix C gives the formal statements and proofs of representation consistency and relation sensitivity. The code repository provides the implementation and Lean proofs for the finite reference matcher.

3.4 Generation from Retrieved Examples

Each retrieved case provides a figure as a design reference and a program showing how it was produced. A visualization model or agent receives these examples with the new request and data:

$$\hat{p} = G_\theta\big(x, D, \mathcal{E}_k(D)\big). \tag{4}$$

The generator adapts the examples to the new task. Section 5.1 evaluates the resulting figures, and Appendix Figure 9 illustrates the process for a forest plot.

4 OMNIFIGUREBENCH

OmniFigureBench evaluates visualization reuse through representation consistency (T1), relation sensitivity (T2), and intention alignment (T3). Appendix Table 6 summarizes the tasks.

4.1 SOURCE CASES

The collection spans R, Python, and declarative visualization ecosystems. Figure 2 summarizes source and domain coverage across the reference library and benchmark; Appendix D describes the benchmark records and task inputs. Cases must include complete data and code that run end to end; curators check the figures and assign visual complexity. Data and task definitions are provided in the OmniFigureBench repository.

For the retrieval tasks, write the input as $z = (x, D)$, where x is the analysis request and D includes the data and any supplied metadata. Let $y_i(z) \in \{0, 1\}$ indicate whether this input supports the analysis represented by case c_i, allowing the code to be adapted to the new data. The label definition does not depend on the retrieval method. OmniGrammar schemas are the representations used by our method, defined in Section 3.

4.2 T1: REPRESENTATION CONSISTENCY

T1 changes the representation of a source case while preserving the information needed for its analysis. For a source input z_i with $y_i(z_i) = 1$, an admissible transformation satisfies

$$z_i^+ = T_{\text{repr}}(z_i), \qquad y_i(z_i^+) = y_i(z_i) = 1. \tag{5}$$

The analysis request stays fixed; any changes to identifiers or metadata must preserve their correspondence to the observations. Consistently relabeling an individual across linked tables, for example, leaves that individual's repeated measurements paired.

The current protocol tests source recovery from anonymized data: a hit records whether c_i is retrieved for z_i^+. Other returned cases may also be useful, so this metric measures source recovery rather than exhaustive relevance. The broader task design includes wide-to-long reshaping, lossless decomposition into linked tables, and mixed record representations.

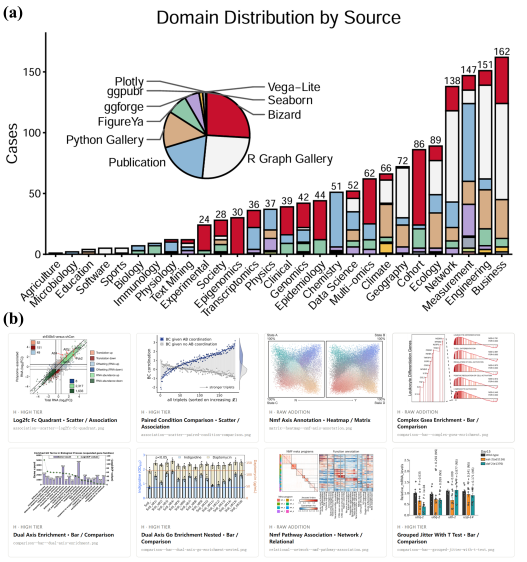

Figure 2: Visualization resources for OmniFigure and OmniFigureBench. (a) Case counts by domain, colored by source; the pie chart summarizes source contributions. (b) Library examples include paired comparisons, annotated heatmaps, and composite layouts.

4.3 T2: RELATION SENSITIVITY

T2 removes a relationship needed by the source analysis while retaining similar surface features. Starting from the same valid source input, the intervention must satisfy

$$z_i^- = T_{\text{rel}}(z_i), \qquad y_i(z_i^-) = 0. \tag{6}$$

For an analysis of individual change, replacing repeated measurements of the same people with measurements from separate cohorts removes the required pairing. The column names and primitive types can remain unchanged.

The task design also covers loss of joint observations and observation-level information.

Summarization invalidates a distribution analysis only when the required information is lost. Verifying Equation (6) requires a judgment about the source analysis, independent of OmniGrammar.

4.4 T3: Intention Alignment

T3 asks whether a model or agent can fulfill a specified visualization intent. The generator receives the task's data and request, with retrieved figures and code in retrieval-augmented conditions. Evaluation distinguishes code execution, fulfillment of the requested analysis, and visual quality.

4.5 Reference Labels and Evaluation Protocol

The source case provides the retrieval target for T1. T2 needs an additional judgment about the analysis: which relationship does it require, and does the intervention remove that relationship? The reported inclusion checks establish reproducibility and visual complexity. They do not establish the labels in Equation (6). We therefore specify T2 as an evaluation design with independent labeling separated from OmniGrammar-based retrieval decisions.

T1 and T2 are interpreted jointly so that rejection after a relation change is assessed alongside recovery on the corresponding compatible input.

Source-recovery tests retain the source case in the candidate pool. Section 5 specifies the shared candidate pools and retrieval budgets, and the exclusion of evaluation solutions from generation references.

5 EXPERIMENT

5.1 Visualization Generation

Benchmarks and systems. We evaluate generation on **MatPlotBench**, which contains 100 scientific plotting tasks with explicit visual specifications (Yang et al., 2024); **PlotCraft**, using 491 visualization goals in single- and multi-turn settings (982 instances) (Zhang et al., 2026b); and the 1,010 intent-driven tasks in **OmniFigureBench-T3**.

We evaluate two general models (GPT-5.6 Sol (OpenAI, 2026) and DeepSeek V4.1 Flash (DeepSeek, 2026)), three visualization models (Text2Chart31 (Pesaran Zadeh et al., 2024), VisCoder2 (Ni et al., 2026), and PlotCraftor (Zhang et al., 2026b)), and four agents (MatPlotAgent (Yang et al., 2024), Text2Vis (Rahman et al., 2025), nvAgent (Ouyang et al., 2025), and CoDA (Chen et al., 2026)). The model baselines share a ReAct framework (Yao et al., 2023) implemented with LangChain (LangChain, 2026). Its plotting tool executes generated code in a sandbox and returns feedback for revision (Appendix D). Agent systems use GPT-5.6 Sol as the common backbone.

For each generator G, we compare G with OmniFigure+G under identical inference and execution settings; only the augmented run receives retrieved figures and programs. Reference collections (Figure 2) exclude external benchmark cases. MatPlotBench uses Matplotlib and OriginLab gallery examples (Yang et al., 2024); PlotCraft uses newly authored tasks and solutions on Kaggle data (Zhang et al., 2026b). For OmniFigureBench, the evaluation case and duplicate or equivalent source solutions are excluded from retrieval. We use Gemini 2.5 Pro (Google, 2025) for evaluation. All models use their default thinking level or default dynamic thinking budget.

Evaluation. An output is valid only if execution succeeds and produces a readable, non-empty visualization. Valid outputs are scored on 11 VLM dimensions from RealChart2Code (Zhang et al., 2026a): chart type, layout, text, axes, color, style, data pattern, component completeness, clarity, compositional balance, and typography. Each dimension is scored 0, 1, or 2; invalid outputs receive zero. For T3, the rubric evaluates the requested analysis and data use without requiring the reference layout. With dimension scores s_{ij} and validity $V_i \in \{0, 1\}$,

$$Q_i = \frac{V_i}{11} \sum_{j=1}^{11} s_{ij}, \qquad \text{VLM} = \frac{1}{N} \sum_{i=1}^{N} Q_i, \qquad \text{Pass} = \frac{1}{N} \sum_{i=1}^{N} V_i. \tag{7}$$

Table 1: Generation results across three benchmarks; paired rows compare each generator with its OmniFigure-augmented version. Pass reports valid outputs (%); VLM averages all tasks, counting failures as zero. Expert averages five independent ratings on the scored subset. VLM and Expert range from 0 to 2; bold marks the higher value within each pair.

Method	MatPlotBench			PlotCraft			OmniFigureBench		
	Pass (%)	VLM	Expert	Pass (%)	VLM	Expert	Pass (%)	VLM	Expert
Closed-Source Models									
GPT-5.6 Sol	88.00	1.30	1.52	72.40	0.78	0.84	78.12	1.03	1.12
+ OmniFigure	**92.00**	**1.37**	**1.56**	**77.19**	**0.89**	**1.12**	**88.12**	**1.16**	**1.64**
Open-Source Models									
DeepSeek V4.1 Flash	82.00	1.14	1.28	59.88	0.70	1.00	68.71	0.85	1.13
+ OmniFigure	**86.00**	**1.20**	**1.62**	**61.41**	**0.73**	**1.21**	**75.45**	**0.93**	**1.38**
Visualization Models									
Text2Chart31	**79.00**	**1.00**	**1.30**	**33.71**	**0.31**	**0.97**	**46.73**	**0.54**	1.19
+ OmniFigure	73.00	0.91	1.20	32.38	0.28	0.78	45.64	0.53	**1.41**
VisCoder2	82.00	1.06	1.36	44.60	0.47	**1.10**	54.85	0.65	**1.28**
+ OmniFigure	**85.00**	**1.12**	**1.78**	**45.93**	**0.51**	0.85	**57.82**	**0.68**	1.12
PlotCraftor	80.00	1.08	1.32	70.77	0.84	0.91	65.35	0.83	1.35
+ OmniFigure	**84.00**	**1.16**	**1.49**	**73.73**	**0.93**	**1.41**	**71.98**	**0.90**	**1.52**
Visualization Agents									
MatPlotAgent	95.00	1.43	1.00	74.85	0.78	1.04	76.34	1.01	1.22
+ OmniFigure	**98.00**	**1.51**	**1.54**	**82.38**	**1.04**	**1.09**	**86.63**	**1.13**	**1.45**
Text2Vis	89.00	1.22	**1.52**	72.30	0.75	1.03	73.27	0.94	1.35
+ OmniFigure	**94.00**	**1.32**	1.16	**78.82**	**0.95**	**1.64**	**83.47**	**1.07**	**1.58**
nvAgent	**86.00**	**1.08**	0.86	71.59	0.57	0.53	67.13	0.73	1.24
+ OmniFigure	82.00	1.00	**1.24**	**72.71**	**0.62**	**0.99**	**72.67**	**0.80**	**1.39**
CoDA	98.00	1.56	1.30	75.56	0.79	1.05	76.14	1.00	1.24
+ OmniFigure	**99.00**	**1.62**	**1.72**	**84.52**	**1.10**	**1.49**	**88.02**	**1.16**	**1.42**

Five domain experts holding doctoral degrees independently score generated figures; Appendix E.1 describes the rubric and compares their ratings with VLM scores.

5.2 Retrieval Analysis

Protocol and baselines. Retrieval uses the 1,010-case OmniFigureBench library. T1 evaluates 272 representation-preserving cases under the anonymized-input protocol in Section 4. T2 contains 253 control–intervention episodes from 243 source cases. Control and intervention inputs are queried independently; methods receive the raw benchmark inputs, field semantics, and value types, but no source identity, transformation provenance, gold label, or gold OmniGrammar annotation. Multiple interventions from one source are averaged before source-case macro-averaging.

We compare BM25 (Robertson & Zaragoza, 2009), Qwen3-Embedding (Zhang et al., 2025), an LLM Retriever, adapters of CoDA (Chen et al., 2026), RAGvis (Helali et al., 2025), RGVisNet (Song et al., 2022), Prompt4Vis (Li et al., 2025), and ReSpark (Tian et al., 2025), and the relationship-aware SANTOS baseline (Khatiwada et al., 2023). All methods rank the same candidate pool with the same Top-K cutoffs.

Metrics. Let r_i^0 and r_i^+ be the T1 source ranks before and after transformation, and let r_e^c and r_e^- be the T2 source ranks for the compatible control and relation-breaking intervention. With source

Table 2: Retrieval under representation changes (T1) and relation-breaking interventions (T2). H@K and Ret@K measure source recovery and retention; R@K measures rejection after intervention. Pair@K requires recovery on the control and rejection on the intervention. All values are percentages; higher is better.

Method	Representation Consistency (%)				Relation Sensitivity (%)			
	H@1	H@5	H@10	Ret@10	R@1	R@5	R@10	Pair@10
BM25	11.40	23.90	33.46	66.91	71.60	40.95	23.46	18.11
Qwen3-Embedding	19.85	36.03	50.37	77.84	74.07	31.48	14.40	11.73
LLM Retriever	28.68	50.37	65.44	82.41	68.93	38.89	29.84	25.10
CoDA-style Retriever	24.63	44.49	58.46	81.54	83.74	61.52	45.06	35.60
RAGvis Retriever	22.43	38.97	54.04	76.17	97.94	82.30	61.73	31.89
RGVisNet Retriever	15.44	29.41	42.65	70.30	99.59	97.74	95.88	10.49
Prompt4Vis Retriever	24.63	43.75	57.72	78.89	92.59	83.74	81.48	36.21
ReSpark Retriever	26.84	47.43	61.40	82.67	77.16	45.68	26.54	21.40
SANTOS Retriever	35.29	61.76	75.00	85.71	85.80	63.37	43.62	41.98
OmniGrammar	**49.63**	**77.57**	**89.34**	**93.46**	**99.18**	**97.94**	**87.65**	**85.39**

cases $c \in \mathcal{C}_2$ and episodes $e \in E_c$,

$$\begin{aligned} \mathrm{H@}K &= \frac{1}{N_1}\sum_i \mathbf{1}[r_i^+ \le K], \qquad \mathrm{Ret@}K = \Pr(r_i^+ \le K \mid r_i^0 \le K), \\ \mathrm{R@}K &= \mathbb{E}_c\mathbb{E}_{e\in E_c}\mathbf{1}[r_e^- > K], \quad \mathrm{Pair@}K = \mathbb{E}_c\mathbb{E}_{e\in E_c}\mathbf{1}[r_e^c \le K < r_e^-]. \end{aligned} \tag{8}$$

We report H@1/5/10 and Ret@10 for representation consistency, and R@1/5/10 and Pair@10 for relation sensitivity. Pair@10 requires the source case to be retrieved for the compatible control and rejected after the relation-breaking intervention.

5.3 Main Results

OmniFigure improves visualization generation. Table 1 shows higher pass rates and VLM scores for most generators. For CoDA on PlotCraft, OmniFigure raises Pass from 75.56% to 84.52%, VLM from 0.79 to 1.10, and Expert from 1.05 to 1.49.

VisCoder2 improves Pass and VLM on all three benchmarks, while Text2Chart31 declines on both. Expert scores vary more across generators. Human and VLM scores are positively associated within each benchmark ($\rho = 0.63$–0.74; Appendix E.1). Pooled execution-rate gains are 3.71 percentage points on PlotCraft and 7.02 on OmniFigureBench; Appendix E.2 separates execution gains from quality changes among valid outputs.

OmniGrammar preserves applicability rather than surface similarity. On T1, OmniGrammar reaches 89.34 H@10 and 93.46 Ret@10, compared with 75.00 and 85.71 for SANTOS (Table 2). On T2, Pair@10 rises from 41.98 to 85.39, a gain of 43.41 percentage points over the strongest baseline. RGVisNet obtains 95.88 R@10 but only 10.49 Pair@10, illustrating why recovery and rejection are evaluated jointly. OmniGrammar retains source cases across representation changes while removing them when a required relation is lost.

5.4 Sensitivity and Ablation Analysis

Reference-budget sensitivity. We vary $K \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12\}$ with the retriever, library, generator, and evaluation protocol fixed. Larger budgets extend the same ranked reference list; $K = 0$ uses no references.

VLM rises from 1.035 at $K = 0$ to 1.162 at $K = 6$, then declines at larger budgets (Figure 3). We use $K = 6$ in the main experiments. Low-complexity tasks gain little from additional references, while high-complexity tasks continue to improve beyond $K = 6$.

Component ablation. At $K = 6$, we compare LLM Retriever and OmniGrammar under two libraries with the same generator and protocol. Base Library covers basic chart types and their composition methods; Curated is the full library. ReAct + GPT-5.6 Sol without retrieval is the baseline.

Replacing LLM Retriever with OmniGrammar raises VLM from 1.093 to 1.127 on Base Library and from 1.142 to 1.162 on Curated Library (Table 3). The full configuration raises Pass from 78.12% to 88.12%: retrieved cases help even when basic chart types and composition methods are already available.

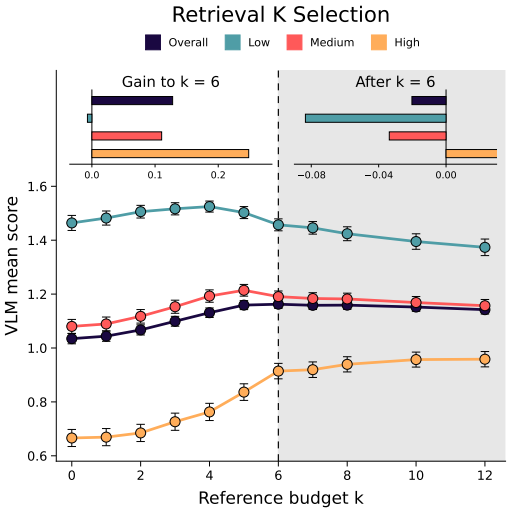

Figure 3: Effect of the reference budget K on OmniFigureBench. Points show mean VLM scores overall and by task complexity; error bars indicate standard errors across tasks. Insets show changes from $K = 0$ to 6 and from $K = 6$ to 12. The retriever, library, and generator are fixed; $K = 0$ uses no references.

Representation ablation. Joint observation structure accounts for a substantial part of OmniGrammar's retrieval gains. With annotations held fixed, SANTOS-style pairwise semantics collapse compatible controls and relation-breaking interventions in 22.84% of T2 pairs, compared with 0.82% for Full OmniGrammar. Removing Fact membership lowers Pair@10 from 85.19% to 46.91%; removing shared identity raises PCR to 7.61% and lowers PairCorrect from 95.06% to 86.83%. The two coordinates preserve which variables are observed together and which observations refer to the same entity. Both are needed in the patient timeline: treatment records must retain their structure and connect to the same patient's summary. These distinctions explain why pairwise semantics alone do not determine whether an analysis transfers. Appendix Tables 8–10 report the full comparisons.

Table 3: Ablation of the retriever and reference library on OmniFigureBench with ReAct + GPT-5.6 Sol. Retrieval conditions use $K = 6$ references; Base covers basic chart types and composition methods, and Curated is the full library. Pass is the valid-output rate (%); VLM reports mean $\pm$ standard deviation on the 0–2 scale, and Δ is the gain over the no-retrieval baseline.

Condition	**Pass ↑**	**VLM ↑**	**Δ vs. Baseline**
ReAct + GPT-5.6 Sol	78.12%	1.035 ± 0.607	—
ReAct + LLM Retriever + Base Library	84.46%	1.093 ± 0.525	+0.059
ReAct + LLM Retriever + Curated Library	86.53%	1.142 ± 0.506	+0.108
ReAct + OmniGrammar + Base Library	87.82%	1.127 ± 0.484	+0.092
ReAct + OmniFigure	**88.12%**	$\mathbf{1.162 \pm 0.490}$	**+0.128**

6 Conclusion

OmniFigure makes visualization experience reusable by matching the structure an analysis requires to the structure available in new data. Compatible figures and programs guide generation. Experiments show improved recovery of applicable examples and rejection of unsupported reuse, with higher generation scores for most systems. Controlled comparisons show how joint observations and shared variable identity resolve reuse decisions that pairwise semantics leave ambiguous. A reference should transfer because the new data support its analysis; layout and code can then be adapted.

AI USE STATEMENT

Generative AI tools assisted literature search, manuscript drafting and editing, analysis code, and preliminary dataset annotation. Final benchmark annotations are human-determined. Experimental model use is described in Section 5. The authors take responsibility for the final manuscript, code, and dataset.

ETHICS STATEMENT

The benchmark retains source attribution for collected visualization cases. Reuse of source data, code, and figures is subject to the original usage and redistribution terms. Human evaluation concerns the quality of generated visualizations, and reported results do not identify individual raters. Generated figures require checking against the input data before scientific use.

REPRODUCIBILITY STATEMENT

The anonymous code and benchmark repositories linked in Sections 4 and 3 provide the implementation and task resources. Appendices A–C specify the matching assumptions, algorithms, and proofs, including the scope of Lean verification. Appendix D describes task inputs; Appendices E.1 and E.4 detail the human evaluation, statistical procedures, and controlled ablations.

REFERENCES

Hannah K. Bako, Xinyi Liu, Leilani Battle, and Zhicheng Liu. Understanding how designers find and use data visualization examples. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1048–1058, 2023. doi: 10.1109/TVCG.2022.3209490.

Hannah K. Bako, Xinyi Liu, Grace Ko, Hyemi Song, Leilani Battle, and Zhicheng Liu. Unveiling how examples shape visualization design outcomes. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):1137–1147, 2025. doi: 10.1109/TVCG.2024.3456407.

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.

Xinyun Chen, Linyuan Gong, Alvin Cheung, and Dawn Song. Plotcoder: Hierarchical decoding for synthesizing visualization code in programmatic context. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 2169–2181. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.169.

Zichen Chen, Jiefeng Chen, Sercan Ö. Arık, Misha Sra, Tomas Pfister, and Jinsung Yoon. CoDA: Agentic systems for collaborative data visualization. In *International Conference on Learning Representations (ICLR)*, 2026.

DeepSeek. DeepSeek-V4.1-Flash: Smarter, faster, more efficient, 2026. URL `https://api-docs.deepseek.com/news/news260910/`. Official release announcement, September 10, 2026.

Google. Gemini 2.5 Pro, 2025. URL `https://ai.google.dev/gemini-api/docs/models/gemini-2.5-pro`. Official model documentation. Accessed September 26, 2026.

Mossad Helali, Yutai Luo, Tae Jun Ham, Jim Plotts, Ashwin Chaugule, Jichuan Chang, Parthasarathy Ranganathan, and Essam Mansour. Reliable and cost-effective exploratory data analysis via graph-guided rag. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 16536–16553. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.836.

Aamod Khatiwada, Grace Fan, Roee Shraga, Zixuan Chen, Wolfgang Gatterbauer, Renée J. Miller, and Mirek Riedewald. SANTOS: Relationship-based semantic table union search. *Proceedings of the ACM on Management of Data*, 1(1):9:1–9:25, 2023. doi: 10.1145/3588689.

LangChain. Agents, 2026. URL `https://docs.langchain.com/oss/python/langchain/agents`. Official documentation. Accessed September 26, 2026.

Shuaimin Li, Xuanang Chen, Yuanfeng Song, Yunze Song, Chen Jason Zhang, Fei Hao, and Lei Chen. prompt4vis: prompting large language models with example mining for tabular data visualization. *The VLDB Journal*, 34(4), 2025. doi: 10.1007/s00778-025-00912-0.

Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB)*, pp. 49–58, 2001.

Kenneth O. McGraw and S. P. Wong. Forming inferences about some intraclass correlation coefficients. *Psychological Methods*, 1(1):30–46, 1996. doi: 10.1037/1082-989X.1.1.30.

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 *International Conference on Learning Representations (ICLR)*, 2026.

OpenAI. GPT-5.6 Sol, 2026. URL `https://developers.openai.com/api/docs/models/gpt-5.6-sol`. Official model documentation. Accessed September 26, 2026.

Geliang Ouyang, Jingyao Chen, Zhihe Nie, Yi Gui, Yao Wan, Hongyu Zhang, and Dongping Chen. nvagent: Automated data visualization from natural language via collaborative agent workflow. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19534–19567. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.960.

Fatemeh Pesaran Zadeh, Juyeon Kim, Jin-Hwa Kim, and Gunhee Kim. Text2Chart31: Instruction tuning for chart generation with automatic feedback. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 11459–11480. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.640.

Chunyao Qian, Shizhao Sun, Weiwei Cui, Jian-Guang Lou, Haidong Zhang, and Dongmei Zhang. Retrieve-then-adapt: Example-based automatic generation for proportion-related infographics. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):443–452, 2021. doi: 10.1109/TVCG.2020.3030448.

Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001. doi: 10.1007/s007780100057.

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.

Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019.

Tetsuya Sakai and Noriko Kando. On information retrieval metrics designed for evaluation with incomplete relevance assessments. *Information Retrieval*, 11(5):447–470, 2008. doi: 10.1007/s10791-008-9059-7.

Yuanfeng Song, Xuefang Zhao, Raymond Chi-Wing Wong, and Di Jiang. 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*, pp. 1646–1655. Association for Computing Machinery, 2022. doi: 10.1145/3534678.3539330.

Yuan Tian, Chuhan Zhang, Xiaotong Wang, Sitong Pan, Weiwei Cui, Haidong Zhang, Dazhen Deng, and Yingcai Wu. Respark: Leveraging previous data reports as references to generate new reports with llms. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery, 2025. doi: 10.1145/3746059.3747644.

Theophanis Tsandilas. Structgraphics: Flexible visualization design through data-agnostic and reusable graphical structures. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):315–325, 2021. doi: 10.1109/TVCG.2020.3030476.

Tracey L. Weissgerber, Vesna D. Garovic, Marko Savic, Stacey J. Winham, and Natasa M. Milic. From static to interactive: Transforming data visualization to improve transparency. *PLOS Biology*, 14(6):e1002484, 2016. doi: 10.1371/journal.pbio.1002484.

Hadley Wickham. Tidy data. *Journal of Statistical Software*, 59(10):1–23, 2014. doi: 10.18637/jss.v059.i10.

Zhiyu Yang, Zihan Zhou, Shuo Wang, Xin Cong, Xu Han, Yukun Yan, Zhenghao Liu, Zhixing Tan, Pengyuan Liu, Dong Yu, Zhiyuan Liu, Xiaodong Shi, and Maosong Sun. Matplotagent: Method and evaluation for llm-based agentic scientific data visualization. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11789–11804. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.701.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Lu Ying, Aoyu Wu, Haotian Li, Zikun Deng, Ji Lan, Jiang Wu, Yong Wang, Huamin Qu, Dazhen Deng, and Yingcai Wu. Vaid: Indexing view designs in visual analytics system. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, 2024. doi: 10.1145/3613904.3642237.

Jiajun Zhang, Yuying Li, Zhixun Li, Xingyu Guo, Jingzhuo Wu, Leqi Zheng, Yiran Yang, Jianke Zhang, Qingbin Li, Shannan Yan, Zhetong Li, Changguo Jia, Junfei Wu, Zilei Wang, Qiang Liu, and Liang Wang. Realchart2code: Advancing chart-to-code generation with real data and multi-task evaluation, 2026a.

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. In *International Conference on Machine Learning (ICML)*, 2026b.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

A OMNIGRAMMAR FORMALIZATION

A.1 LOGICAL STRUCTURE

Let X denote the physical inputs and Γ the semantic and binding information used to interpret them. A binding materializes X into a logical instance I, which is encoded as an OmniGrammar structure H; deterministic derivations then extend the facts available from the data:

$$(X, \Gamma) \xrightarrow{\beta} I \xrightarrow{\alpha} H \xrightarrow{\mathrm{Cl}_\Omega} A(I). \tag{9}$$

Here Ω is a finite collection of explicitly specified deterministic derivations; $A(I)$ augments H with their available materialized outputs and provenance. The matcher consumes this augmented structure. The logical structure is

$$H = (\mathcal{D}, \mathcal{M}, \mathcal{F}, \mathcal{C}, \mathcal{R}), \tag{10}$$

where $\mathcal{D}$ and $\mathcal{M}$ contain Dimensions and Measures, $\mathcal{F}$ Facts, $\mathcal{C}$ constraints, and $\mathcal{R}$ derivations. Table 4 summarizes the information used by matching. Section A.4 applies the representation to the patient timeline in Figure 4.

Table 4: OmniGrammar elements used in matching. Variable attributes constrain correspondences; the remaining elements encode observation structure and analytical requirements.

Element	**Information**	**Role in matching**
Variable	Dimension/Measure, semantic concept, value kind, and value metadata	Restricts admissible correspondences between variables
Fact incidence	Fact membership of a variable occurrence	Preserves which variables are jointly observed
Shared reference	Incidences that refer to the same variable	Preserves identity across Facts
Role	source/target, parent/child, estimate/lower/upper, etc.	Distinguishes positions within a relation
Constraint	key, repeat, functional dependency, cardinality, and topology	Records structural conditions required by an analysis
Derivation	operation, origin, input Facts, and output Facts	Records how required Facts are obtained

A Variable is a Dimension or Measure with a semantic concept and a value kind. Value kinds include number, category, ordinal, datetime, and boolean; units, domains, orders, or meaningful centers are included when they constrain interpretation. A Fact groups variable incidences at one observation grain. Each incidence refers to a Variable, so the same reference can occur in several Facts and make identity explicit across relations. An incidence may also carry a role, allowing the same entity type to occupy distinct positions such as source and target or parent and child.

Constraints record relations that affect applicability but are not determined by Fact membership alone. OmniGrammar includes keys, repeated-entity constraints, functional dependencies, cross-Fact cardinalities, and graph properties such as direction, acyclicity, and symmetry. Derivations connect input and output Facts and record the operation and its origin. Deterministic derivations contribute to $A(I)$; results that depend on stochastic procedures, external data, or human curation retain those provenance boundaries.

Bindings connect physical storage to this logical structure. They cover direct columns as well as wide tables, matrices, trees, and related Facts distributed across multiple files. Different bindings can therefore materialize the same H. Changes in column layout or table organization need not change the structure used for retrieval, while changes to Fact membership, shared identity, roles, constraints, or derivations remain visible to the matcher.

Variables also carry semantic concepts organized in a hierarchy $\mathcal{T}$. Its ancestor relation defines

$$\tau \preceq \tau' \quad \Longleftrightarrow \quad \tau' \text{ is an ancestor of } \tau \text{ or equals } \tau. \tag{11}$$

A more specific data concept can therefore satisfy a coarser case requirement. This relaxation affects variable correspondence only; Fact incidence, shared references, roles, constraints, and derivations are still preserved by the same mapping.

A.2 Compatibility Specification

Let A be an available logical structure and C a case requirement. Write $V_H = \mathcal{D}_H \sqcup \mathcal{M}_H$ for the Variables of a structure H, F_H for its Facts, and S_H for its incidences. Every incidence $s \in S_H$ has an owning Fact $f_H(s)$, a Variable reference $v_H(s)$, and an optional role. Incidences are identified within their owning Fact and Variable class. Matching uses finite, well-formed structures with declared semantic concepts, constraints, and derivations.

A witness consists of three injections,

$$\phi = (\phi_F, \phi_V, \phi_S), \qquad \phi_F : F_C \hookrightarrow F_A, \quad \phi_V : V_C \hookrightarrow V_A, \quad \phi_S : S_C \hookrightarrow S_A. \tag{12}$$

The injections preserve distinct required Facts, Variables, and occurrences. They must also agree on the connections between these objects:

$$\begin{aligned} f_A(\phi_S(s)) &= \phi_F(f_C(s)), \\ v_A(\phi_S(s)) &= \phi_V(v_C(s)). \end{aligned} \tag{13}$$

The Variable class is preserved, and every specified requirement role equals the role of its mapped incidence. An unspecified role places no additional restriction.

Variable compatibility. At the exact tier, semantic concepts agree. At the semantic-parent tier, the available concept equals or descends from the required concept in $\mathcal{T}$. The value kind agrees unless the requirement specifies `any`. Required value refinements are checked as well: units must be compatible, declared ordinal orders must agree, the available domain must lie within the required domain, and a specified center must agree. Supported unit conversions are applied before comparing numeric domains and centers. Let $B_t \subseteq V_C \times V_A$ collect these admissible correspondences at tier t. A witness requires $(v, \phi_V(v)) \in B_t$ for every required Variable.

Constraints and derivations. Table 5 gives the constraint checks after applying ϕ. Each required constraint must have corresponding evidence on the mapped Facts. A required derivation must match an available occurrence with the same operation and origin, with input and output Fact sets equal to the images of its required input and output sets. Sharing an operation name alone does not establish this correspondence.

Table 5: Evidence required to satisfy each constraint under a common compatibility witness ϕ. All Facts and incidences are compared after mapping them to the new data.

Required constraint	**Available evidence**
Key	A declared key contained in the mapped required key. Uniqueness on a subset implies uniqueness on the larger tuple.
Functional dependency	A declaration with the same mapped determinant and dependent incidence sets.
Repeated entity	The same mapped entity and repetition coordinate, with a minimum count at least as large as required.
Cross-Fact cardinality	The same mapped Fact pair, incidence pairs, cardinality kind, and unmatched-record policy.
Topology	The required directed, acyclic, or symmetric declaration on the mapped Fact.

We write $\mathrm{Sat}_t(A, C, \phi)$ when the injections satisfy all these conditions. Eligibility at tier t is

$$\mathrm{Eligible}_t(A, C) \iff \exists \phi \ \mathrm{Sat}_t(A, C, \phi). \tag{14}$$

The matching predicate checks declared structure. Annotation establishes the logical structure to which this predicate is applied.

A.3 Required Facts and Derivation Dependencies

A requirement pattern specifies mandatory Facts R_0 and optional Facts O. A required derived Fact also brings its dependencies into the match. For a derivation occurrence r, let $\mathrm{in}(r)$ and $\mathrm{out}(r)$ be its input and output Fact sets. Starting from R_0, define

$$R_{j+1} = R_j \cup \bigcup_{\substack{r \in \mathcal{R}_C \\ \mathrm{out}(r) \cap R_j \neq \varnothing}} \big(\mathrm{in}(r) \cup \mathrm{out}(r)\big), \qquad R = \mathrm{ReqCl}_C(R_0). \tag{15}$$

Here R is the fixed point of this iteration. Including every output of an activated occurrence preserves the complete input–output tuple checked by the derivation predicate. If a pattern omits an explicit required set, all its Facts are required.

The restriction $C[R]$ retains the selected Facts, their incidences and Variables, constraints whose referenced Facts are all selected, and derivations with selected outputs. ReqCl operates on the requirement side. The available structure $A(I_D)$ in Equation (9) contains the data Facts and their available deterministic derivations. Matching takes this structure as input; dependency completion specifies which case Facts must be checked against it.

A.4 A Patient-Treatment Example

Figure 4 illustrates the specification with three treatment records for one patient. Physical columns bind to logical incidences in a treatment Fact. The patient Variable has concept `bio.patient` and categorical values; start and end are timepoints whose incidence roles distinguish the interval boundaries. The Fact retains each treatment's endpoints and scan response as a joint observation.

The program draws a background line to the patient's latest treatment end time. For records $\mathcal{R}_p$ of patient p, this gives

$$\mathrm{last_time}(p) = \max_{r \in \mathcal{R}_p} \mathrm{end}(r). \tag{16}$$

The displayed endpoints yield 42.68 months. Expressing this grouped maximum as a derivation produces a patient-summary Fact with key(patient). Both Facts share the patient Variable, so a compatibility witness must map their patient incidences consistently despite the different observation grains. Renaming columns preserves this possibility when the bindings recover the same structure.

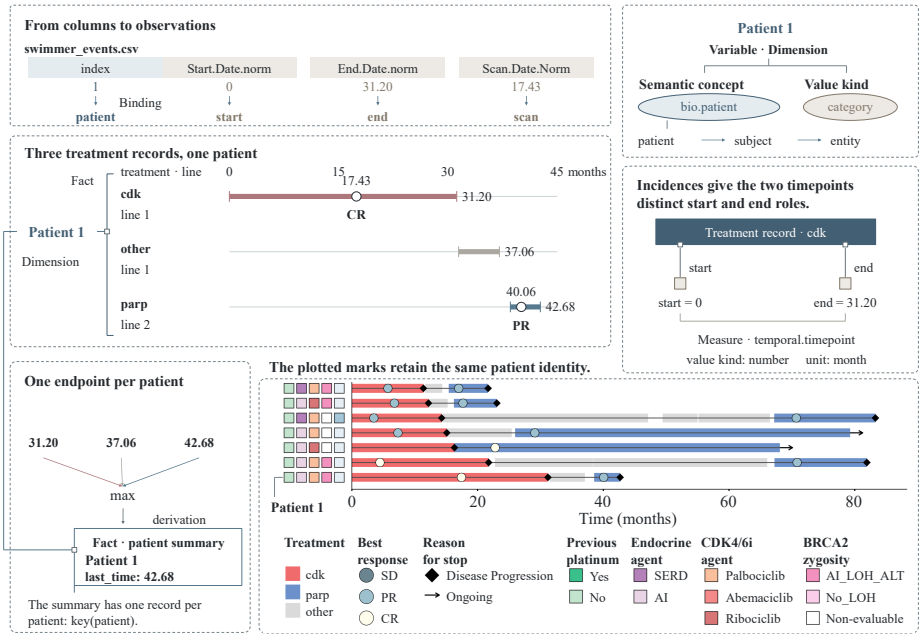

Figure 4: OmniGrammar for a patient timeline. Column bindings recover joint treatment observations, with distinct start and end roles. A grouped maximum yields the patient endpoint; shared identity links the treatment and summary Facts.

B Matching and Retrieval

The retrieval procedure separates eligibility from ordering. Algorithm 1 admits only cases with a joint witness. Algorithm 2 selects the matching tier, required distance, and optional coverage used in Equation (18).

The compatible library is

$$\mathrm{Ret}(D) = \left\{ c_i \in \mathcal{V} : \widehat{C}_i \hookrightarrow A(I_D) \right\}. \tag{17}$$

Ineligible cases are removed before ordering. Among compatible cases, OmniFigure orders matches by specificity. The matcher first searches for an exact witness and then allows semantic-parent correspondence, where a more specific data concept satisfies an ancestor requirement. For case c_i, let $t_i = 0$ for exact matching and $t_i = 1$ for semantic-parent matching, let d_i be the minimum required distance at that tier, summed over distinct Variables within each required Fact (Equation 20), and let o_i be the minimum number of unmatched optional Facts among joint witnesses attaining that tier and distance. The retrieval key is

$$\rho(D, c_i) = \big(t_i, d_i, o_i, \mathrm{id}(c_i)\big). \tag{18}$$

Cases are sorted lexicographically by ρ. Thus exact matches precede semantic-parent matches; within the same tier, smaller semantic distance precedes larger distance; remaining ties prefer greater optional coverage and then a fixed case identifier. Request text is used during query annotation and does not enter this ordering. For a reference budget k,

$$\mathcal{E}_k(D) = \mathrm{TopK}_\rho\big(\mathrm{Ret}(D)\big). \tag{19}$$

Required distance. Let $\delta(a, b)$ be the shortest number of ontology-parent edges from available concept a to required concept b. For a witness ϕ, the required distance is

$$d_R(\phi) = \sum_{f \in R} \sum_{v \in V_C(f)} \delta\big(\tau_A(\phi_V(v)), \tau_C(v)\big), \tag{20}$$

where $V_C(f)$ is the set of distinct Variables referenced by Fact f. Repeated incidences of a Variable within one Fact contribute once. A Variable shared by several required Facts contributes in each of those Facts. The sum includes the prerequisite Facts in R.

Algorithm 1: OmniFigure retrieval

Input: available query structure A, case patterns $\{C_i\}_{i=1}^{n}$, budget k.
Output: up to k compatible cases and their witnesses.

1. Initialize an empty list L.
2. For each case C_i, call $\mathrm{JointMatch}(A, C_i)$.
3. If the call returns (ϕ_i, t_i, d_i, o_i), append the case and witness to L with key $(t_i, d_i, o_i, \mathrm{id}(c_i))$.
4. Sort L lexicographically by its keys and return its first $\min(k, |L|)$ entries.

Algorithm 2: Joint compatibility matching

Input: available structure A, pattern C with required roots R_0 and optional Facts O.
Output: a witness and (t, d, o), or failure.

1. Set $R \leftarrow \mathrm{ReqCl}_C(R_0)$.
2. In order $t = 0$ (exact), then $t = 1$ (semantic-parent), search for witnesses satisfying $\mathrm{Sat}_t(A, C[R], \phi)$. Stop at the first tier with a witness; return failure if both searches fail.
3. At this tier, choose a witness $\phi^\star$ of minimum required distance and set $d^\star \leftarrow d_R(\phi^\star)$, $q \leftarrow |O \cap R|$.
4. Enumerate nonempty subsets $U \subseteq O \setminus R$ in decreasing cardinality. For each, set $S \leftarrow \mathrm{ReqCl}_C(R \cup U)$ and search for a witness of $C[S]$ at tier t, minimizing d_R over the original required set R.
5. At the first feasible extension with $d_R(\phi) = d^\star$, set $\phi^\star \leftarrow \phi$, $q \leftarrow |O \cap S|$, and stop the subset search.
6. Return $(\phi^\star, t, d^\star, |O| - q)$.

Joint optional coverage. Optional Facts improve ranking only when one witness covers them together with the mandatory structure at the best required tier and distance. The extension search can change the original witness while preserving these two quantities. Optional Facts already included as prerequisites count as covered. Searching subsets in decreasing size finds maximum joint coverage: if the closure of a feasible subset covers additional optional Facts, explicitly adding those Facts leaves the same closed set and yields a feasible larger subset.

Witness search. The implementation first constructs candidate Fact correspondences from local Variable and role compatibility. It visits constrained Facts first and intersects the admissible images of shared Variables as Fact assignments accumulate. Fact and Variable images are kept distinct. Shared or constraint-sensitive Variables are searched jointly; remaining local assignments use minimum-cost matching. Incidence assignments and the resulting constraints and derivation occurrences complete the check. Lower bounds on required distance prune assignments that cannot improve the best witness.

For a reference search with f, v, s required Facts, Variables, and incidences, and F, V, S available objects, the number of candidate injection triples is at most

$$(F)_f\,(V)_v\,(S)_s, \qquad (n)_m = \frac{n!}{(n-m)!} \quad (m \le n). \tag{21}$$

The count is zero when a required carrier is larger than its available counterpart. The optimized search restricts these candidates before checking complete witnesses. Optional selection has at most $2^{|O \setminus R|}$ subsets, including the empty subset covered by the initial required match.

C PROPERTIES OF OMNIGRAMMAR MATCHING

C.1 CORRECTNESS OF THE REFERENCE MATCHER

Let $\mathrm{Match}_t(A, C)$ enumerate the finite injection triples in Equation (12) and accept when a triple satisfies Sat_t. The following statement applies to the required structure after dependency completion and to a fixed admissible correspondence relation B_t.

Proposition 1 (Matching correctness) *For finite well-formed structures A and C,*

$$\mathrm{Match}_t(A, C) = 1 \quad \Longleftrightarrow \quad \mathrm{Eligible}_t(A, C). \tag{22}$$

Proof. If the matcher accepts, an enumerated triple passes every condition of Sat_t. This triple is a witness of eligibility. Conversely, suppose a witness exists. Each of its three components is an injection between finite carriers. Enumerating all injections therefore includes each component, and their Cartesian product includes the complete witness. Its check succeeds, so the matcher accepts. Both the enumeration and each check are finite, establishing termination. □

Shared-reference preservation. For required incidences s_1, s_2, Equation (13) and injectivity of ϕ_V give

$$v_C(s_1) = v_C(s_2) \quad \Longleftrightarrow \quad v_A(\phi_S(s_1)) = v_A(\phi_S(s_2)). \tag{23}$$

The forward implication follows because ϕ_V is a function; the reverse follows because it is injective. Thus a reference shared across Facts remains shared, while distinct required references remain distinct.

Why local matches can disagree. Consider a requirement with two Facts, each containing a patient incidence that refers to the same Variable p. Suppose the corresponding available Facts contain patient Variables a and b, with $a \neq b$, and no other Fact candidates exist. Each Fact admits a local semantic match. Together, however, they require both $\phi_V(p) = a$ and $\phi_V(p) = b$. No global witness exists. This is the structural obstruction captured by shared identity.

C.2 INVARIANCE TO LOGICAL REPRESENTATION

An isomorphism $\eta : A \cong A'$ consists of bijections on Facts, Variables, and incidences that preserve their connections, Variable metadata, roles, constraint declarations, and derivation occurrences. Semantic concepts and the ontology are fixed. Consequently, $B_t(v, a)$ holds exactly when $B'_t(v, \eta_V(a))$ holds.

Proposition 2 (Representation consistency) *If $A \cong A'$ in this sense, then for every case requirement C,*

$$\mathrm{Eligible}_t(A, C) \quad \Longleftrightarrow \quad \mathrm{Eligible}_t(A', C). \tag{24}$$

Proof. Let ϕ witness eligibility in A. Define $\phi' = (\eta_F \circ \phi_F, \eta_V \circ \phi_V, \eta_S \circ \phi_S)$. A bijection composed with an injection is injective. Preservation of Fact ownership and Variable references by η gives Equation (13) for ϕ'. Variable compatibility and roles are preserved by assumption. Each available constraint or derivation witnessing a required declaration is carried by η to the corresponding declaration in A', so every remaining condition holds. Hence ϕ' is a witness in A'. Composing a witness in A' with η^{-1} proves the reverse implication. □

If a transformation T changes only physical representation and the corresponding bindings and annotations yield $A(I_{T(D)}) \cong A(I_D)$, then

$$\mathrm{Ret}(T(D)) = \mathrm{Ret}(D). \tag{25}$$

The witness correspondence also preserves ontology distances and the optional subsets that admit a joint extension. For a fixed library and case identifiers, the retrieval keys and ordered Top-k results are therefore unchanged.

Relation sensitivity. If a transformation removes information required by a case, the case becomes ineligible exactly when no alternative witness remains:

$$\mathrm{Compat}(D', c_i) = 0 \quad \Longleftrightarrow \quad \nexists \phi : \widehat{C}_i \hookrightarrow A(I_{D'}). \tag{26}$$

C.3 Dependency Closure and Retrieval Order

Least dependency closure. The iteration in Equation (15) is monotone and ranges over subsets of the finite Fact set. Each strict update adds at least one Fact, so it reaches a fixed point after at most $|F_C| - |R_0|$ strict updates. The result contains R_0 and includes all inputs and co-outputs of every activated derivation. To show minimality, let Q be any set with these properties. Initially $R_0 \subseteq Q$. If $R_j \subseteq Q$, every derivation activated by R_j is also activated by Q, whose closure property puts all newly added Facts in Q. Induction gives $R_j \subseteq Q$ at every step and hence $\mathrm{ReqCl}_C(R_0) \subseteq Q$.

Eligibility before ranking. Every entry admitted by Algorithm 1 has a witness. Sorting and taking a prefix preserve this property. With fixed match metadata and unique case identifiers, the lexicographic key defines a deterministic order, independent of the order in which cases are visited. Increasing k extends the existing prefix and never introduces an ineligible case. These properties concern the specified compatibility relation and ordering; generation subsequently adapts the selected programs as in Equation (4).

The Lean reference model verifies finite witness search for a supplied Variable-compatibility relation, shared-reference preservation, least requirement closure, and ordering over supplied match metadata. Its derivation model uses nonempty input and output Fact sets. Proposition 1 corresponds to its soundness and completeness statements on this domain. Proposition 2 follows from the witness-transport argument above. Algorithm 2 specifies the implementation's tier selection and joint optional search.

D OmniBenchmark

Table 6 summarizes the three tasks and their expected behavior.

Table 6: OmniFigureBench tasks and expected responses. T1 tests reuse after representation changes, T2 tests rejection after relation loss, and T3 tests generation under a stated intent.

Task	Task setting	Expected response
T1: Representation	Layout or identifiers change; observations and their relations are preserved.	Continue to retrieve the source example despite the change in representation.
T2: Relations	A relation needed by the original analysis is removed.	Exclude the source example when the new data no longer support its analysis.
T3: Intent	A short task prompt specifies what the figure should emphasize.	Produce a data-faithful figure that reflects the specified intent without reproducing a fixed layout.

Benchmark records. Each OmniFigureBench case records a fixed upstream revision, input data, plotting code or specification, a reference image, and bindings from physical fields to logical Facts. Figure 5 outlines its domains and task settings.

Retrieval inputs. T1 query bundles pair anonymized tables with metadata describing value kinds, nullability, variable identity, and available structural constraints. Columns and identifier values use local names; occurrences of the same logical Variable retain a shared reference. Candidate contracts describe table and field requirements. Relevance annotations are stored separately as known-positive and known-negative candidate IDs; unjudged candidates remain unlabeled. The controlled representation comparison in Appendix E.4 uses this contract library.

Relation interventions. Each T2 episode contains two inputs with the same request, field semantics, and file names. The intervention changes the observations or their linkage. Cohort splits

separate design and measurement records; marginalization replaces joint observations with separate variable-value frequencies; summarization replaces observation-level values with aggregate statistics. Other episodes break identity across occasions or remove within-group treatment contrasts. For a same-flower scatterplot, separate marginal distributions retain both variables but no longer identify which length and width measurements belong to the same flower.

Generation prompts. Each T3 task selects the data files needed for its request and records a reference case for evaluation. Prompts state the comparisons to emphasize and may suggest an encoding. For example, a genomic task asks the generator to compare gene order, interval length, spacing, and orientation across genomes, with aligned tracks and directional intervals as possible design choices. The prompt leaves the arrangement of these elements to the generator.

Generation interface. The reference implementation uses LangChain's `bind_tools` and `invoke` interfaces (**?**). `run_python(code)` executes plotting code in a sandbox and returns execution feedback; `read_reference(number, view)` provides retrieved code or images for OmniFigure. A compact prompt asks the agent to answer the request using the supplied data, preserve the required observation relationships, and inspect the rendered figure before submission.

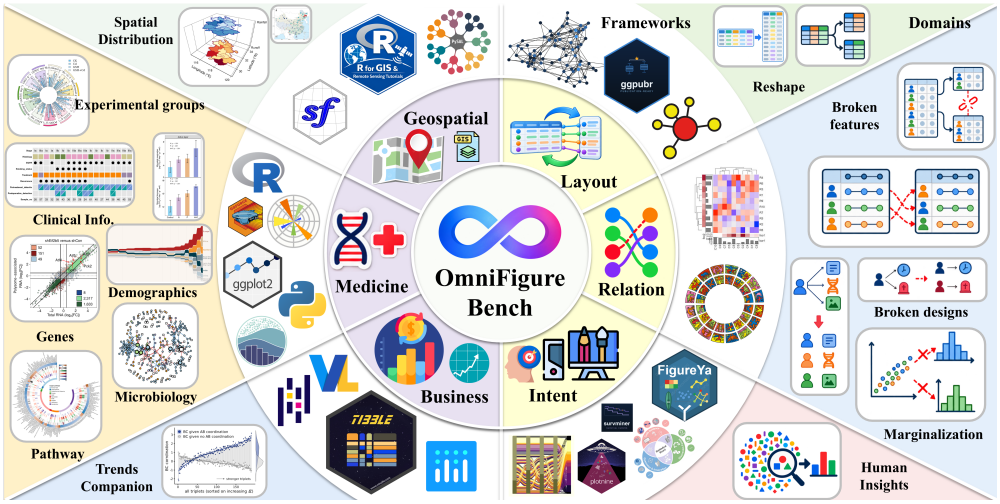

Figure 5: Coverage of OmniFigureBench. Inner sectors organize application domains and evaluation axes; surrounding examples show visualization frameworks, data reshaping, relation changes, and analytical intents.

E Additional Results and Ablation

E.1 Human–VLM Agreement

We compare human and VLM scores on 156 outputs across the three benchmarks. Five domain experts holding doctoral degrees independently score each output on Structure, Layout, Text/Axis, Color/Style, and Clarity, using a 0–2 scale. The human score averages these dimensions and then averages the five raters; the VLM score averages its 11 evaluation dimensions. Each benchmark subset covers all 18 method conditions, comprising nine generators with and without OmniFigure. All rated outputs pass execution. The PlotCraft subset contains 32 single-turn and 27 multi-turn outputs from 58 underlying tasks.

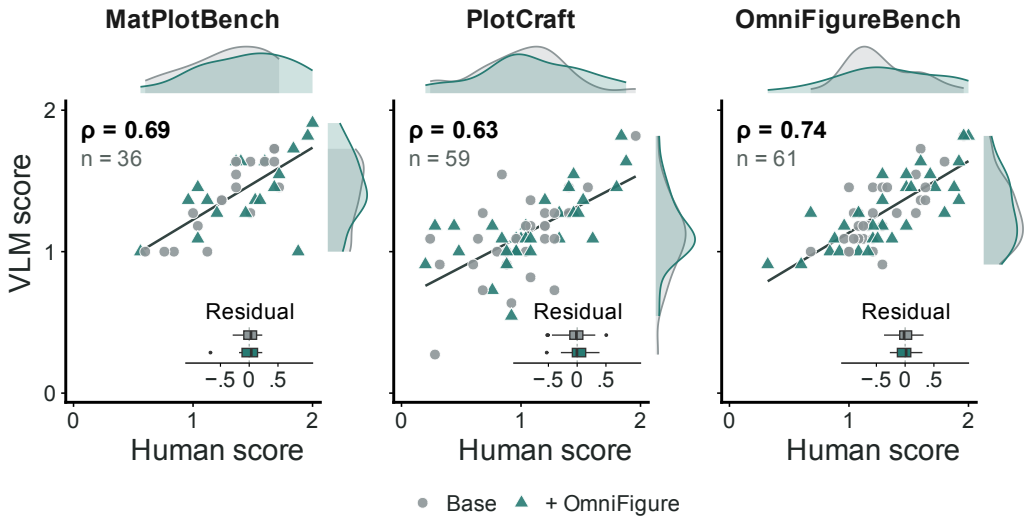

Figure 6: Human–VLM score agreement within each benchmark. Each point is one output; human scores average five experts, and both scores range from 0 to 2. Lines fit all outputs within a benchmark; ρ denotes Spearman correlation. Marginal densities and inset residual boxplots show the score distributions and deviations from the fit for each method condition.

Table 7: Human rating reliability and association with VLM scores. ICC(A,5) measures absolute agreement of the five-expert mean; Spearman ρ measures its association with VLM scores. Counts distinguish rated outputs from underlying tasks and identify the base and OmniFigure conditions. Brackets give 95% task-cluster bootstrap intervals.

Benchmark	**Outputs (tasks)**	**Base / OmniFigure**	**ICC(A,5) [95% CI]**	**Spearman ρ [95% CI]**
MatPlotBench	36 (36)	18 / 18	0.95 [0.91, 0.96]	0.69 [0.38, 0.88]
PlotCraft	59 (58)	29 / 30	0.96 [0.94, 0.98]	0.63 [0.43, 0.78]
OmniFigureBench	61 (61)	30 / 31	0.96 [0.94, 0.97]	0.74 [0.58, 0.84]

Figure 6 and Table 7 show positive rank association within every benchmark, with ρ ranging from 0.63 to 0.74. Absolute agreement of the five-rater mean, measured by the two-way ICC(A,5) (McGraw & Wong, 1996), ranges from 0.95 to 0.96. The VLM scores therefore track human judgments across the rated outputs, while the scatter and residuals show variation at the individual-output level. Both statistics use 4,000 bootstrap draws over underlying tasks, keeping outputs from the same task together; PlotCraft turn variants share a task identity.

E.2 Quality and Execution Gains

We pair outputs by benchmark, task, turn setting, and generator. Figure 7 separates two changes after adding OmniFigure: the mean VLM score difference among pairs that pass execution in both conditions, and the execution-rate difference over all pairs. There are 900 pairs for MatPlotBench, 8,838 for PlotCraft, and 9,090 for OmniFigureBench. Intervals use 4,000 task-cluster bootstrap draws, retaining all generator and turn variants within a sampled task. The pooled row combines generators within each benchmark.



Figure 7: Generation gains from adding OmniFigure, paired by task and generator. Left: VLM score changes among pairs that execute successfully in both conditions. Right: execution-rate changes over all pairs, in percentage points (pp). Points and bars show mean differences and 95% task-cluster bootstrap intervals; positive values favor OmniFigure. Pooled rows combine generators within each benchmark.

PlotCraft shows the largest conditional score gains, particularly for CoDA, MatPlotAgent, and Text2Vis. OmniFigureBench shows larger execution gains: the pooled increase is 7.02 percentage points, compared with 3.71 on PlotCraft and 1.56 on MatPlotBench. Text2Chart31 has negative point estimates for both measures on all three benchmarks, and several MatPlotBench intervals cross zero.

E.3 Generation with Retrieved Examples

Figure 8 illustrates layouts for a complex heatmap task with two measurements at each row–column intersection and a categorical row annotation. Shared row and column orders let readers compare the measurements directly. The examples include split cells, paired marks, and adjacent matrices; the class annotation remains aligned with the same rows.

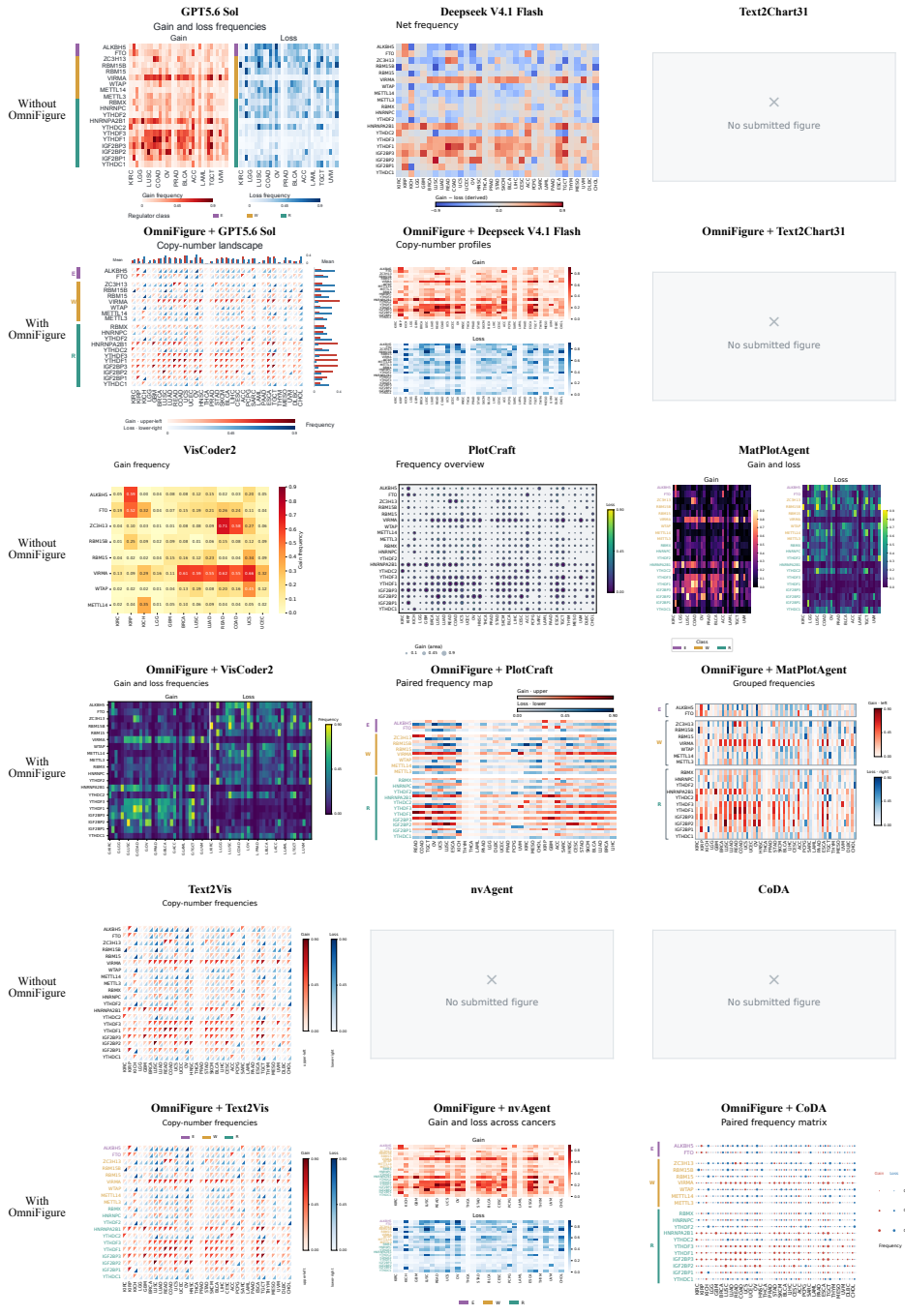

Figure 8: Complex-heatmap generation with and without OmniFigure. Each vertical pair uses the same generator and task, with the baseline above and OmniFigure below. The task requires two measurements per row–column pair and a categorical row annotation; outputs illustrate alternative ways to align these elements. Gray panels indicate no submitted figure.

E.4 OMNIGRAMMAR REPRESENTATION ABLATION

Controlled Representations. We compare unary column semantics, pairwise relations, and Full OmniGrammar using the same reviewed annotations. We then remove one coordinate from Full OmniGrammar at a time, keeping the other coordinates and evaluation inputs fixed.

For an annotated structure H, write

$$H_{\mathrm{col}} = \pi_{\mathrm{col}}(H), \qquad H_{\mathrm{pair}} = \pi_{\mathrm{pair}}(H), \qquad H_{\mathrm{full}} = H. \tag{27}$$

Column Semantic retains the Dimension/Measure class, semantic concept, value kind, and available value refinements of each Variable. SANTOS-style Pairwise also retains binary semantic relationships between Variables. Full OmniGrammar retains Fact membership and the shared references that connect incidences across Facts, together with incidence roles, constraint evidence, and derivation occurrences. Its compatibility decision requires these correspondences to agree within one global witness.

Each projection is applied to both query and candidate annotations. The first two representations use a common controlled matcher; Full OmniGrammar uses the compatibility predicate in Appendix A.2. The pairwise condition is constructed from the reviewed annotations, so its results measure the information supplied by binary relations under this protocol. Table 10 records the annotation and matching conditions.

Evaluation Sets and Metrics. The T1 study ranks a fixed pool of 272 candidates for 272 queries, with 568 known-positive query–candidate associations. Its query set contains 32 general-, 133 medium-, and 107 high-complexity cases. T2 contains 253 control–intervention episodes from 243 source cases: 105 cohort-split, 81 marginalization, 57 summarization, nine occasion-specific-identity, and one treatment-confounding episode. The candidate pools, judged labels, and retrieval cutoff are held fixed across representation conditions.

T1 retrieval. Let P_q and N_q be the known-positive and known-negative candidate sets for query q, and let L_q^{10} be the first ten entries of its raw ranking. For $\mathcal{Q}$ the set of queries, we compute

$$\mathrm{Recall@10} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{|P_q \cap L_q^{10}|}{|P_q|}. \tag{28}$$

This averages recall over queries, giving each query equal weight despite differing numbers of known positives. Candidates outside $P_q \cup N_q$ remain unjudged. Following condensed-list evaluation (Sakai & Kando, 2008), we remove these candidates to obtain $\widetilde{L}_q$. If $\widetilde{r}_q$ is the first positive rank in $\widetilde{L}_q$, then

$$\mathrm{MRR}_{\mathrm{cond}} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{1}{\widetilde{r}_q}, \qquad \frac{1}{+\infty} = 0. \tag{29}$$

Recall@10 uses the raw ranking; MRR uses the condensed ranking. The main retrieval comparison searches the larger 1,010-case library and reports source-case hit and retention metrics, so its values describe a separate setting.

T2 structural collisions. For episode e, let S_e^c and S_e^- denote the compatible control and intervention annotations. Pair collision rate measures whether a projection makes these inputs isomorphic. With E_c the episodes from source case c,

$$\mathrm{PCR}_\pi = \frac{1}{|\mathcal{C}_2|} \sum_{c \in \mathcal{C}_2} \frac{1}{|E_c|} \sum_{e \in E_c} \mathbb{1}\left[\pi(S_e^c) \cong \pi(S_e^-)\right]. \tag{30}$$

A collision identifies a distinction erased by the representation before a retrieval decision is made. The isomorphism comparison preserves the labels retained by the projection and ignores arbitrary object identifiers.

T2 compatibility and retrieval. Let $a_e^c, a_e^- \in \{0,1\}$ be acceptance decisions for the source analysis, and r_e^c, r_e^- its ranks for the two inputs. The episode scores are

$$\begin{aligned} \mathrm{PairCorrect}_e &= \mathbb{1}\left[a_e^c = 1 \wedge a_e^- = 0\right], \\ \mathrm{Pair@10}_e &= \mathbb{1}\left[r_e^c \le 10 < r_e^-\right]. \end{aligned} \tag{31}$$

For either score m_e, the reported value is

$$\overline{m} = \frac{1}{243} \sum_{c \in \mathcal{C}_2} \frac{1}{|E_c|} \sum_{e \in E_c} m_e. \tag{32}$$

Thus every source analysis receives equal weight, including sources with several interventions. PairCorrect evaluates the compatibility decision; Pair@10 evaluates whether the source survives selection for the control and leaves the retrieved set after intervention.

Table 8: Retrieval with progressively richer representations of the same reviewed annotations. T1 reports raw Recall@10 and condensed MRR. T2 measures structural collisions (PCR), correct control–intervention decisions (PairCorrect), and paired top-10 retrieval (Pair@10). Lower PCR and higher values of the remaining metrics are better.

Representation	T1 Recall@10	T1 MRR	T2 PCR ↓	T2 PairCorrect ↑	T2 Pair@10 ↑
Column Semantic	74.52%	0.48	62.55%	25.10%	18.52%
SANTOS-style Pairwise	82.86%	0.59	22.84%	65.23%	54.94%
Full OmniGrammar	**91.43%**	**0.75**	**0.82%**	**95.06%**	**85.19%**

From Column Semantics to Joint Structure. Binary relations recover a substantial part of the information lost by independent column descriptions (Table 8). They raise T1 Recall@10 from 74.52% to 82.86% and reduce T2 PCR from 62.55% to 22.84%. Full OmniGrammar further improves Recall@10 to 91.43% and reduces PCR to 0.82%. The accompanying increase in PairCorrect from 65.23% to 95.06% shows that the retained distinctions also improve compatibility decisions under this protocol.

The remaining gap is consistent with the shared-reference obstruction in Appendix C: locally admissible correspondences can fail to agree across the full requirement. Fact membership and cross-Fact identity constrain these correspondences jointly. The coordinate ablations below separate their contributions from the other structural conditions.

Removing Individual Coordinates. For coordinate j, let $\pi_{-j}(H)$ erase that coordinate while retaining the others. We apply π_{-j} to both sides of each comparison and rerun the same evaluation. The five conditions remove: shared reference identity; membership of occurrences in Facts; incidence roles; constraint evidence; or the connection between a derivation occurrence and its input/output Facts. Variables retain their unary semantic attributes in all five conditions. The source analyses and their applicability labels stay fixed.

Fact membership has the largest aggregate effect (Table 9). Its removal raises PCR to 32.10% and lowers Pair@10 to 46.91%, exposing the importance of preserving which values belong to one observation. Removing shared identity raises PCR to 7.61% and lowers PairCorrect to 86.83%, separating identity consistency from observation membership. Constraint and derivation ablations also reduce Pair@10, to 80.66% and 74.28%, respectively.

The evaluated T2 contracts contain explicit shared-identity and multi-Fact requirements; explicit role-bearing and derivation-occurrence contracts are absent. The role and derivation rows therefore describe aggregate sensitivity to those coordinates in this corpus.

Controlled and Native Pairwise Comparisons. The controlled comparison derives all three representations from the same reviewed annotations (Table 10). Native SANTOS uses its own annotation and ranking pipeline in the main experiment.

Table 9: One-coordinate ablations of OmniGrammar. Each row removes the named coordinate from query and candidate representations while retaining the remaining information, candidate pools, and evaluation protocol. Full OmniGrammar provides the reference condition. All values are percentages; lower PCR and higher retrieval and decision scores are better.

Representation	T1 Recall@10	T2 PCR ↓	T2 PairCorrect ↑	T2 Pair@10 ↑
− Shared Identity	86.69%	7.61%	86.83%	82.92%
− Fact Membership	81.88%	32.10%	55.56%	46.91%
− Incidence Role	89.94%	1.65%	91.98%	84.98%
− Constraint Evidence	85.02%	2.47%	88.89%	80.66%
− Derivation Occurrence	88.45%	7.00%	84.36%	74.28%
Full OmniGrammar	**91.43%**	**0.82%**	**95.06%**	**85.19%**

Table 10: Controlled comparison of column semantics, pairwise relations, and Full OmniGrammar. The upper block specifies how each representation and matcher are obtained; the lower block reports retrieval and compatibility results. All conditions derive from the same reviewed annotations and use the full T1/T2 evaluation sets. Lower PCR and higher remaining scores are better.

Representation	Annotation source	Matcher / scoring	Scope
Column Semantic	Reviewed OmniGrammar → unary projection	Common controlled matcher	Full T1/T2
SANTOS-style Pairwise	Reviewed OmniGrammar → pairwise semantic graph	Common controlled matcher	Full T1/T2
Full OmniGrammar	Reviewed OmniGrammar annotation	Full OmniGrammar matcher / contract semantics	Full T1/T2

Representation	T1 Recall@10	T2 PCR ↓	T2 PairCorrect ↑	T2 Pair@10 ↑
Column Semantic	74.52%	62.55%	25.10%	18.52%
SANTOS-style Pairwise	82.86%	22.84%	65.23%	54.94%
Full OmniGrammar	**91.43%**	**0.82%**	**95.06%**	**85.19%**

Ablation by Structural Condition. Table 11 groups T2 episodes by the requirements present in their Full OmniGrammar annotations. For shared identity, multiple required Facts, and explicit constraints, we compare Full OmniGrammar with removal of Shared Identity, Fact Membership, and Constraint Evidence, respectively. Subsets stay fixed across the comparison. Scores are averaged within source case and then across sources in each subset; 95% intervals use 4,000 source-cluster bootstrap draws. Feature subsets can overlap, and a source may contribute different episodes to both the present and absent subsets.

Removing constraint evidence has its largest effect in the ten source cases with explicit constraints: PairCorrect falls by 60.0 percentage points and Pair@10 by 50.0. Fact membership affects both subsets substantially, with Pair@10 drops of 38.7 and 38.2 points. Shared-identity drops are larger in the subset requiring shared identity, although its intervals are wide. Thus the aggregate ablations include both concentrated losses on specific requirements and losses spread across the corpus.

Table 11: Effects of coordinate removal when the corresponding structural requirement is present or absent. Multiple required Facts are tested by removing Fact membership. Drops are Full OmniGrammar minus the ablated score, in percentage points (pp); positive values indicate a loss. PC denotes PairCorrect and P@10 denotes Pair@10. Brackets give 95% source-cluster bootstrap intervals.

Feature	Subset	Episodes / sources	Full PC (%)	PC drop (pp) [95% CI]	Full P@10 (%)	P@10 drop (pp) [95% CI]
Shared identity	Present	33 / 33	90.9	15.2 [−3.0, 33.3]	78.8	9.1 [−12.1, 30.3]
	Absent	220 / 220	95.5	7.7 [2.7, 12.7]	85.9	1.8 [−4.6, 8.2]
Multiple required Facts	Present	31 / 31	87.1	45.2 [25.8, 64.5]	71.0	38.7 [16.1, 61.3]
	Absent	222 / 212	96.2	38.7 [31.6, 46.2]	87.3	38.2 [30.4, 46.0]
Explicit constraints	Present	10 / 10	90.0	60.0 [30.0, 90.0]	80.0	50.0 [10.0, 90.0]
	Absent	243 / 243	95.1	4.9 [0.8, 9.5]	85.2	3.3 [−2.9, 9.9]

E.5 Case Study

Figure 9 illustrates how a coding agent can use OmniFigure to turn cell-level records into a forest plot relating cell proportions to two clinical outcomes. The final layout shares a row order across outcomes, places numerical estimates beside their intervals, and groups cell types through side annotations.

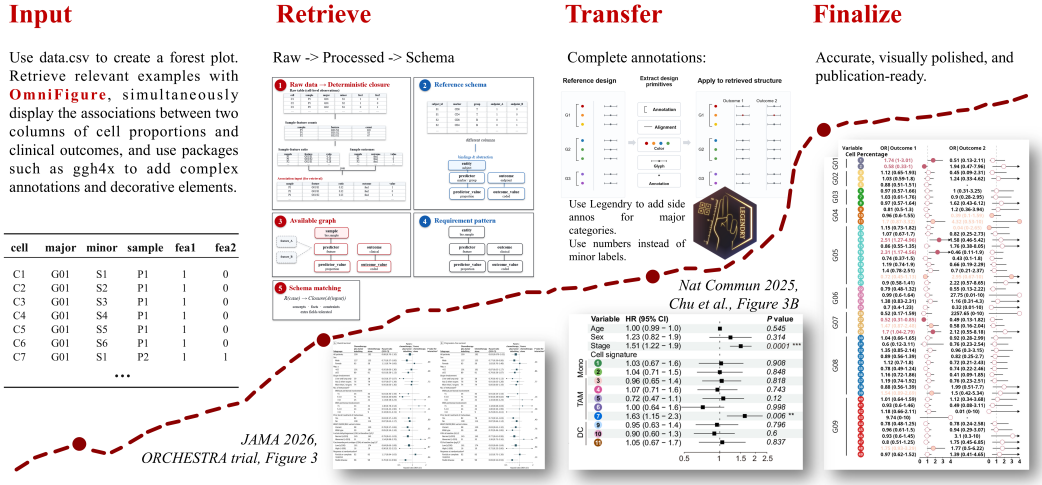

Figure 9: OmniFigure-assisted forest-plot generation in a coding agent. From left to right, the agent aggregates cell records into sample-level proportions, retrieves structurally compatible references, and adapts their layout and annotations. The final figure aligns cell types across two outcomes, with group annotations and numerical estimates beside the plotted intervals.

Request and Data Preparation. An illustrative interaction begins with: "Use OmniFigure to create a forest plot from `data.csv`. Show cell-proportion associations with both outcomes side by side, with annotations for major groups and minor cell types." The agent checks sample identifiers and outcome consistency within each sample. It counts cells of type k in sample s, computes $p_{sk} = n_{sk} / \sum_j n_{sj}$, and joins these proportions to outcomes by sample. The sample is the analysis unit; individual cells contribute to its proportions.

Retrieval. The agent invokes OmniFigure on the prepared data and request. The query annotation records the sample, predictor, outcome, proportion, and coded outcome value, together with the derivation from the cell-level table. A reference can use different names, such as `subject_id`, `marker`, and `endpoint`, while requiring the same observation structure. After OmniGrammar matches those requirements, the agent opens the retrieved figures and programs to inspect their layouts.

Design Transfer. Some library cases include design evidence: brief notes that a model prepares in advance from the figure and plotting code to explain useful design choices. For this task, the notes can suggest parallel outcome columns and compact group annotations. The agent adapts the layout to align cell types across outcomes and place estimates beside their intervals.

Rendering and Revision. The agent writes and executes the plotting program, then inspects the image. It checks that corresponding rows identify the same cell type, intervals remain attached to their estimates, and group labels span the intended rows. Revisions address label collisions, outcome-panel spacing, and the width of numerical annotations while keeping the analytical table fixed. The resulting layout lets readers follow each cell type across both outcomes.