

VisCodeBench: a new benchmark dataset for Text-to-Vis reflecting real-world practice

Hy Nguyen^{1*}, Thomas James Elton¹, Ella Wilson¹, Andrew Reeson², Cécile Paris²,
Josiah Poon¹, Jonathan K. Kummerfeld^{1†}
The University of Sydney¹ CSIRO's Data61²
nngu0448@uni.sydney.edu.au* jonathan.kummerfeld@sydney.edu.au[†]

Abstract

Generating visualisations from natural language requests (text-to-vis) enables users to explore data more efficiently. While LLMs can complete simple visualisation requests, their performance on challenging ones is not well covered by existing benchmarks. We introduce VisCodeBench, an end-to-end text-to-vis benchmark with 2,672 samples from three different domains: educational, scientific, and general audience scenarios with varying levels of complexity. We benchmark three visualisation systems—Codex, Claude Code, and LIDA—paired with a range of open and commercial LLMs across seven metrics covering execution, chart content, and visual quality. While basic execution is near-solved for strong systems, no system comes close to reproducing the reference data, with the best displayed data similarity reaching only 0.81 and falling to 0.69 on our hardest subset for this metric. We further find that frontier models separate on chart correctness and efficiency, with ClaudeCode-Opus-4.7 improving chart correctness by around 5 points over Codex-GPT-5 while running roughly 20 seconds faster. Separately, we find that system design matters as much as the underlying model: despite both running on GPT-5, Codex clearly outperforms LIDA, showing that the surrounding prompting, execution, and refinement pipeline can shape outcomes as much as the model itself. Finally, latency remains a practical obstacle, as even the strongest systems take 39-78 seconds to produce results. VisCodeBench is available at <https://github.com/SAIPLL/VisCodeBench> and accessed via a leaderboard; the full dataset is kept private to prevent LLM training use.

1 Introduction

Text-to-Vis is the task of taking a human language request for visualisation of some data and generating code to produce the corresponding visualisation. A range of benchmarks exist for this task, including nvBench (Luo et al., 2021), nvBench2.0 (Luo et al., 2025), ChartX (Xia et al., 2025), Text2Chart31 (Pesaran Zadeh et al., 2024), MatPlotBench (Yang et al., 2024), and Text2Vis (Rahman et al., 2025). However, most of these datasets are synthesised or automatically constructed and do not match real-world use, as shown in our earlier work (Nguyen et al., 2024). Often, the visualisations lack meaningful real-world context, or the generated code follows fixed templates.

We introduce VisCodeBench, a new benchmark dataset for evaluating text-to-vis systems in real-world scenarios. VisCodeBench is based on three sources of data. (1) GitHub-T2V contains 843 samples based on assignment notebooks used in online courses shared on GitHub. (2) arXiv-T2V is comprised of 805 samples from academic papers, collected by contacting authors and requesting the chart, code, underlying data, and a natural language request describing the visualisation. (3) OWID-T2V includes 1,024 samples from <https://ourworldindata.org>, a publicly accessible platform with data-driven analyses of global issues. After collecting the raw data, expert annotators, each with over four years of coding experience and extensive experience creating academic visualisations, verified and

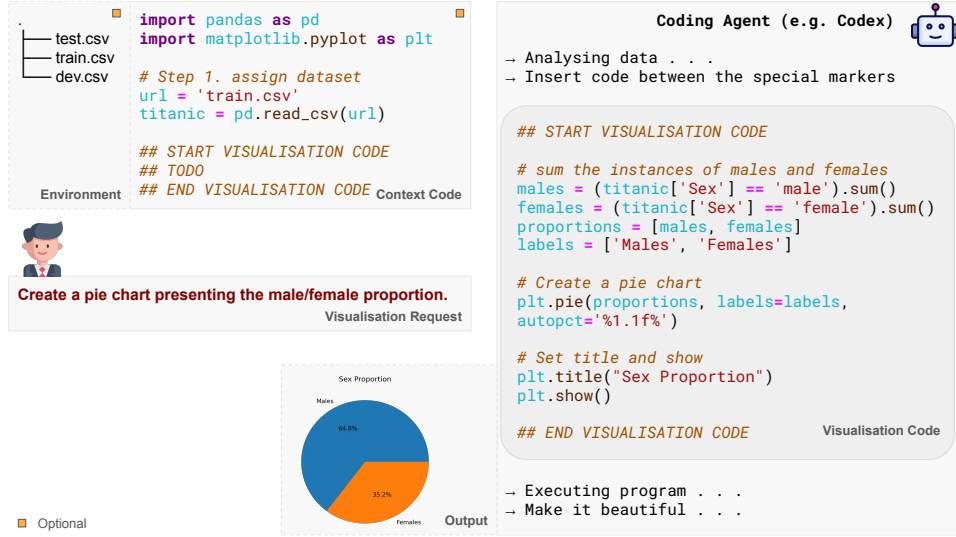


Figure 1: A Text-to-Vis system. Left: input data files, code context, and user request. Right: system output.

refined the samples. A range of statistics show that the final datasets are more difficult and realistic than existing benchmarks.

For evaluation, we propose two new metrics in addition to five existing ones. (1) Displayed Data Similarity assesses the similarity of data points displayed in generated and reference charts (Davila et al., 2019), providing a more general, LLM-independent, and flexible alternative to exact-match evaluations used in existing work (Chen et al., 2025; Yang et al., 2024); (2) Accessibility Score measures contrast and spacing of texts displayed on the visualisation images (Miranda-Pena et al., 2024).

We benchmark three visualisation systems—Codex, Claude Code, and LIDA—paired with open (GPT-OSS-20B, Gemma-4-31B, Qwen-3.6-35B) and commercial (GPT-5, Opus-4.7) LLMs. Our results show that basic execution is near-solved for strong systems, with only the small local model failing to reliably produce valid charts. Content fidelity is the harder challenge, as no system reproduces the reference data well. The best displayed data similarity comes from Claude Code with Opus-4.7, but it reaches only 0.81 and falls to 0.69 on OWID-T2V, the hardest subset for displayed data similarity. VisCodeBench separates frontier models: they differ on chart correctness score and efficiency, with ClaudeCode-Opus-4.7 improving chart correctness by around 5 points over Codex-GPT-5 while running roughly 20 seconds faster, making Opus-4.7 the strongest model overall. We further show that a system’s design can weigh as heavily as its underlying model: Codex and LIDA share the same GPT-5 backbone, yet on OWID-T2V, LIDA completes only 79% of tasks compared to Codex’s 100%, with correctness falling to 54.37% versus 73.67%. Finally, latency remains a barrier to interactive use across every system we test. Together, these findings highlight the challenges of realistic text-to-vis scenarios and demonstrate the value of VisCodeBench in exposing them.¹

2 Related Work

2.1 Development of Text-to-Vis Benchmarks

Early Text-to-Vis benchmarks were manually created. For example, for the NLV Corpus (Srinivasan et al., 2021) and ChartDialogs (Shao & Nakashole, 2020), crowd workers

¹The evaluation code, representative samples, leaderboard, and are available at [VisCodeBench GitHub](#), and the full benchmark leaderboard is linked from that page.

were given real-world data tables and charts. They then wrote commands that described how a system should create a chart, change its layout, or modify its visual style. This method is closely aligned with real-world use because the charts, data, and requests originate directly from people. However, it has clear limits: only a few chart types are used, the data tables are small, and the natural language requests are in a single turn or short.

To address these limits, researchers moved to semi-automated and later fully automated approaches. Semi-automated datasets—such as nvBench (Luo et al., 2021), Dial-nvBench (Song et al., 2024), VisEval (Chen et al., 2025), MatplotBench (Yang et al., 2024), Text2Vis (Rahman et al., 2025), and nvBench2.0 (Luo et al., 2025)—begin with a real-world data table, from which a synthesiser generates a chart and a corresponding natural language request; then human annotators verify and refine the results. Fully automated approaches take this further by using LLMs to generate the data table, natural language request, code, and visualisation from pre-designed prompts, e.g., for Chartllama (Han et al., 2023), ChartX (Xia et al., 2025), and Text2Chart31 (Pesaran Zadeh et al., 2024). Although these methods allow rapid dataset scaling and control over diversity and balance, they tend to be less realistic: the data is often synthetic, the charts convey little meaningful real-world insight, and the outputs lack the customisation typical of genuine use. Also, the requests and visualisations often follow strict templates made by dataset designers, making them less like real user requests.

In our work, we source charts from data created by real users, avoiding the issues with the synthetic worker-written data raised above. To ensure quality, we then use human effort with AI assistance to refine code and requests.

2.2 Automated Evaluation for Text-to-Vis

Execution-based Evaluation Methods evaluate visualisations by executing the generated programs and comparing the resulting charts with reference chart outputs in terms of chart type and displayed data. To compare the displayed data, prior work commonly uses the extracted x- and y-coordinates, obtained either by parsing the rendered output (e.g., from SVG files) or by logging values directly during execution (Maddigan & Susnjak, 2023; Li et al., 2024; Chen et al., 2025). While this approach directly verifies whether a visualisation encodes the intended data, it introduces several challenges. First, it requires substantial post-processing, such as normalising numerical values and enforcing consistent naming for temporal units. Second, the comparison is often overly strict: a single misplaced or incorrectly scaled data point can cause the entire output to be marked as incorrect.

LLM-based Evaluation Methods employ LLMs (e.g., GPT-4o) to assess the readability and quality of generated chart images, or their correctness against reference charts (Dibia, 2023; Chen et al., 2025; Rahman et al., 2025; Yang et al., 2024). This approach is easy to implement and correlates well with human judgment. However, it depends on proprietary models and incurs substantial cost when scaled to large datasets.

To address these limitations, we adopt Displayed Data Similarity, a more flexible metric that measures the similarity between the two sets of displayed data points. Inspired by CHART-Infographics (Davila et al., 2019), a competition focused on extracting raw data from pixel-image charts, our metric offers a more tolerant, informative, and LLM-independent measure of performance.

3 Dataset Construction

Our benchmark includes 2,672 samples, grouped into three subsets: **GitHub-T2V**, **arXiv-T2V**, and **OWID-T2V**. After collection, the raw data went through an annotation process involving both LLM-assisted and human interventions, followed by expert manual review. The dataset is formatted as illustrated in Figure 1 to facilitate easy benchmarking of coding agents, such as Codex and Claude Code.

3.1 GitHub-T2V

GitHub-T2V contains 843 text-to-vis samples collected from an educational context, where students write visualisation code as part of assignments in data science, data analysis, and machine learning courses. Each sample includes the underlying data and dependent files, the relevant context code, a natural language visualisation request, the corresponding visualisation code, and the resulting chart.

The construction process followed a five-step pipeline. First, we scanned The Stack ², a large-scale pre-training dataset for code LLMs, to identify GitHub URLs of MIT-licensed Python notebooks whose filenames contain “solution”, indicating homework solutions or teaching materials. Second, we randomly sampled 5,000 notebooks and retrieved their associated environment files, including imported modules and underlying data files, from the original GitHub repositories. Third, we automatically executed the notebooks using a custom script, retaining only those that ran successfully within a 60-second time limit and discarding notebooks with runtime errors. Fourth, we used GPT-4o to automatically detect natural-language visualisation requests, which typically appear in markdown cells immediately preceding code cells that generate charts. Finally, we manually verified the detected natural language visualisation requests and constructed samples in our format.

3.2 arXiv-T2V

arXiv-T2V is comprised of 805 text-to-vis samples collected from scientific research settings, where researchers write code to visualise experimental data using graphs and charts. The raw data for this subset were collected via an online survey approved by our organisations’ ethics committees.³ We invited approximately 30,000 authors of CC BY-SA 4.0–licensed papers published on arXiv in 2024 using the email addresses listed in their papers, and further promoted the study through social media and academic mailing lists. Participants were asked to submit chart images from their papers, along with the corresponding code, underlying data, and a newly written natural language request describing how they would ask another person to create the visualisation. In total, we received 1,151 valid submissions from 848 participants, corresponding to a response rate of 2.69%. Additional details are provided in Appendix B.

We filtered down to suitable visualisations and modified the code to suit our benchmark. We manually de-identified and selected 805 submissions that target Python-based visualisation code and involve substantial data processing and manipulation. We then determined the placement of the natural language request within the code: when the underlying data are provided as files, the request is inserted at the beginning of the Python code file; when the data are embedded directly in the code, the request is placed right after the data definition.

Although we asked participants to write a natural language request, many of the submitted requests were overly abstract or appeared to be generated by large language models. To address this issue, we created new natural language requests for the charts. We used GPT-5 to generate initial natural language requests of approximately 120 words, ensuring that the requests were concrete and sufficiently detailed. Native English-speaking experts with over four years of programming experience in Python-based academic data visualisation revised the initial requests. During this stage, the experts were provided only with the raw data (data files or code-embedded data), the initial requests, and the resulting visualisation images—they were not given access to the visualisation code. They were advised to explore and visualise the data by writing and executing their own code until they developed a clear understanding of the data and the visualisations. The experts then revised the initial requests, phrasing them as they would when prompting a coding agent to generate the visualisations from the given data.

Annotation Quality To assess annotation quality, an expert with over ten years of programming experience and extensive involvement in academic data visualisation manually rated 60 samples. The expert had full access to each annotated sample, and was asked to

²<https://huggingface.co/datasets/bigcode/the-stack>

³For details of ethics approval, see the Ethics Statement.

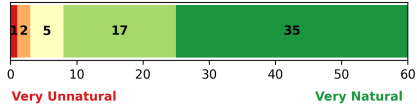


Figure 2: Annotation quality assessment of arXiv-T2V

rate the naturalness of the request on a five-point Likert scale (1 = very unnatural, 5 = very natural). As shown in Figure 2, the score distribution shows that most samples received high ratings: 52 out of 60 requests received scores of 4 or 5, with an average score of 4.34, indicating a generally high level of naturalness.

The expert was also asked to provide reasons for any rating below 5. The most common issue was the use of excessively long paragraphs that combine multiple categories of information—such as data loading, data processing, the core visualisation request, and chart styling instructions. Other issues include the use of specialised mathematical symbols that are difficult to type on a standard keyboard, as well as over-specification, such as repeating items already present in the context or providing unnecessary details about numerical formats (e.g., decimals and percentages).

3.3 OWID-T2V

OWID-T2V consists of 1,024 text-to-vis samples collected from <https://ourworldindata.org/>, a publicly accessible research platform where data scientists and researchers curate datasets, develop visualisations, and write analytic articles on major global issues. Each sample includes a underlying data file, a natural language visualisation request, and the corresponding chart image. Since the original image was produced by custom javascript, we do not have code for these samples. We instead provide the reference displayed data points and the original chart images for evaluation.

We built the data by collecting analytical articles from <https://ourworldindata.org/>. For each article, we extracted the associated charts and related materials, including: (1) chart images; (2) the analytical discussion written by the article authors, in which the charts are used as supporting evidence; (3) chart- and dataset-level metadata; (4) the full CSV dataset used to generate each chart; and (5) the displayed CSV data shown in the chart after filtering and processing by the website.

To generate initial natural language requests, we used GPT-5. The model was guided to write a 50-word request, following the wording and grammatical structure suggested by the chart and its associated materials, with particular emphasis on the analytical discussion in which the chart is referenced (e.g., “The bar chart illustrates global births in 2025”). Expert annotators then revised these drafts to produce the final human-authored requests, as described in section 3.2.

Finally, we reconstructed the reference displayed data points in x-, y-, and z-coordinates from the published CSV data. The reference data points were created by: (1) using Codex with GPT-5 to replicate each chart image based on the published CSV data and metadata; (2) manually comparing the replicated charts with the original images and repeating this process until a visually similar result was obtained, discarding samples that fail to converge after three attempts; and (3) extracting the reference data points using custom scripts, which are described in Appendix D.

4 Dataset Analysis

4.1 Statistics

Libraries Used Figure 3 (left) shows the ten most commonly used Python libraries in GitHub-T2V and arXiv-T2V subsets. Among them, matplotlib, numpy, pandas, seaborn, sklearn, and scipy stand out as the most prevalent, reflecting their essential roles in data

manipulation, transformation, and visualisation. Libraries such as `random`, `os`, `json`, and `pickle` also appear frequently, primarily for tasks like loading data, controlling randomness, and handling I/O. The popularity of `statsmodels` on GitHub-T2V versus `torch` on arXiv reflects the distinct characteristics of these two datasets.

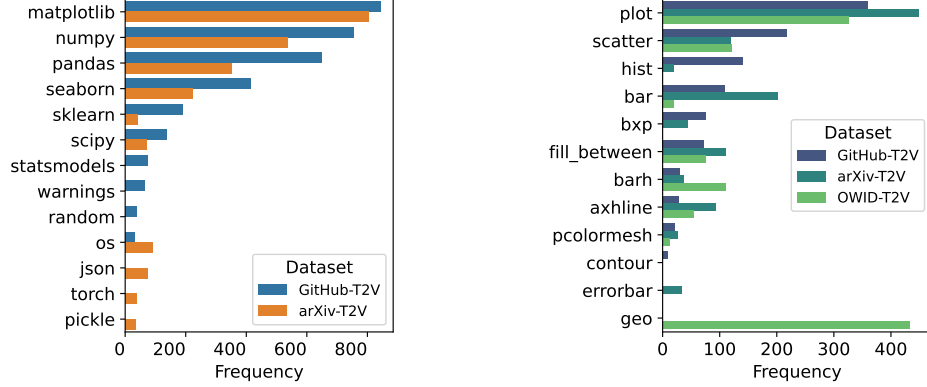


Figure 3: Top 10 libraries and plot types used in VisCodeBench.

Plot type Figure 3 (right) illustrates the distribution of plot types across our data subsets. GitHub-T2V and arXiv-T2V exhibit very similar chart-type distributions: line charts are the most popular, followed by scatter plots, histograms, bar charts, box plots, and area charts. The most striking outlier is the `geomap` category, which appears almost exclusively in OWID-T2V, reflecting the dataset’s emphasis on geographical data.

Analysis of other dimensions Table 1 (lower) presents statistics across various dimensions. Overall, the arXiv-T2V dataset exhibits longer outputs, a greater variety of plot types, and more subplots, indicating a stronger need for customisation in both style and layout of the resulting visualisations.

4.2 Comparison with real-world code

Following the approach from our earlier work [Nguyen et al. \(2024\)](#), we compare code from our dataset subsets with real-world visualisation code collected from the Internet, focusing specifically on Matplotlib. In terms of chart type distribution, GitHub-T2V and arXiv-T2V exhibit strong alignment with real-world data, with line graphs being the most common chart type, followed by scatter and bar charts. For usage patterns of 62 common attributes, GitHub-T2V and arXiv-T2V show strong alignment with real-world data, with Spearman’s rank correlation coefficients exceeding 0.91. Regarding complexity, while GitHub-T2V exhibits the same level of complexity as real-world data, arXiv-T2V is more complex, with an average of 9.20 distinct functions and 17.32 parameters per program. Further details of these comparisons are provided in [Appendix C](#).

4.3 Comparison with related datasets

We focus comparisons on seven public end-to-end benchmark datasets: NLV Corpus ([Srinivasan et al., 2021](#)), nvBench ([Luo et al., 2021](#)), nvBench2.0 ([Luo et al., 2025](#)), ChartX ([Xia et al., 2025](#)), Text2Chart31 ([Pesaran Zadeh et al., 2024](#)), MatplotBench ([Yang et al., 2024](#)), and Text2Vis ([Rahman et al., 2025](#)), as summarised in Table 1. Other existing datasets are excluded because they are variants of these benchmarks or do not support end-to-end text-to-vis evaluation.

Compared to existing datasets, ours show significant advantages in input/output length, the number of distinct libraries used per sample, the presence of multiple subplots per chart, and the flexibility of data format. These characteristics indicate that our benchmark dataset places higher demands on models, requiring stronger contextual understanding, the ability

	Annotation note	Test Samples	Data Format	Plot Types	Distinct libs*	Sub- plots*	Input Length*	Output Length*	Diversity		Complexity	
									Input	Output	#func*	#args*
NLV Corpus	<i>D, R, C, I</i>	893	Fixed	4	1.0	1.55	57.7	376	0.64	0.01	7.11	12.56
nvBench	<i>D, R, C, I</i>	3990	Fixed	4	1.0	1.00	131	709	0.13	0.01	4.59	10.02
nvBench2.0	<i>D, R, C, I</i>	751	Fixed	6	1.0	1.00	3532	458	0.07	0.03	4.94	5.54
ChartX	<i>D, R, C, I</i>	1152	Fixed	16	1.98	1.18	1030	1159	0.20	0.09	10.19	16.03
Text2Chart31	<i>D, R, C, I</i>	1423	Fixed	31	2.46	1.08	1069	767	0.38	0.18	8.64	11.09
MatPlotBench	<i>D, R, I</i>	100	Fixed	13	-	1.55	1532	-	0.62	-	-	-
Text2Vis	<i>D, R, C, I</i>	1985	Fixed	20	1.71	1.15	1153	1219	0.18	0.16	9.02	14.2
GitHub-T2V	<i>D, R, C, I</i>	843	Free	18	4.08	1.96	6745	227	0.13	0.31	10.34	15.61
arXiv-T2V	<i>D, R, C, I</i>	805	Free	21	3.15	2.54	1557	2847	0.46	0.20	10.54	20.37
OWID-T2V	<i>D, R, DD, I</i>	1024	Fixed	6	-	1.15	209	-	0.55	-	-	-

Table 1: Comparison with related benchmarks. Metrics with a * are averaged over samples. D, R, C, and I indicate underlying data, natural language request, code, and chart image, respectively. Annotation notes displayed in italic blue text denote components that were created by *humans*. Note, OWID-T2V does not have code, which is why Displayed Data (DD) is provided for evaluation and some values are blank. Data Format denotes whether the underlying data has a **fixed** structure (e.g., CSV) or varies **freely** in schema and format.

to generate longer and more complex code, and the capacity to produce charts with multiple subplots and customisations. In comparison to the NLV Corpus, a fully human-curated dataset, our dataset is comparable in test set size but supports a wider range of plot types.

We also measured content diversity for both inputs and outputs, tokenised with the GPT-2 tokenizer (Radford et al., 2019). arXiv-T2V achieves high diversity scores for both inputs and outputs, despite processing longer input sequences and generating substantially longer code outputs than the other datasets. This result is noteworthy because diversity measured by distinct n-grams typically decreases with increasing sequence length due to repetition (Li et al., 2016).

We further evaluated the complexity of visualisation programs using the metric we proposed in prior work Nguyen et al. (2024). In this analysis, we considered visualisation functions from Matplotlib, Pandas, and Seaborn, rather than limiting the evaluation to Matplotlib alone, as computed in section 4.2. As shown in the last two columns of Table 1, our arXiv-T2V exhibits high complexity, with an average of approximately 10.54 distinct visualisation functions and 20.37 visualisation parameters per program.

5 Evaluation Metrics

We implemented seven evaluation metrics, which differ in their evaluation targets and level of granularity. Together, they provide a comprehensive assessment of system performance in terms of both data presented and presentation style.

Task Completion Rate (TCR) We compute $TCR = \frac{C}{T}$. Where C is how many samples the generated code runs successfully on and produces at least one visualisation image, and T is the total number of samples.

Prediction Time This is the time spent, measuring the total time required to complete the visualisation task.

Chart Type Accuracy This measures whether the predicted and reference chart types match. In our benchmark, we implement this metric to consider multiple chart types in both generated and reference charts. Let $P = \{p_1, \dots, p_m\}$ and $G = \{g_1, \dots, g_n\}$ denote the set of chart types in the generated and reference visualisations. A prediction is considered correct if $|P \cap G| = |G|$.

Displayed Data Similarity After extracting displayed data points using a monkey-patching technique (see Appendix D for details), we adopt the evaluation metric from Task 6b of the ICDAR 2019 Competition on Harvesting Raw Tables from Infographics (Davila et al., 2019), with several modifications to better suit our setting. (1) When comparing data

series from two lines, we replace linear interpolation, which causes errors for area charts containing cycles, with arc-length-parameterised interpolation to establish stable pointwise correspondence along the curves (Farin, 2014). (2) We extend the original data-series-level similarity computation to figures with multiple subplots: subplot scores are computed from their constituent data series, and the overall figure score is obtained by aggregating the subplot scores.

Chart Correctness We adopt the chart correctness evaluation method proposed by Yang et al. (2024), which employs an LLM as a judge to assign a correctness score on a 100-point scale to each pair of predicted and reference chart images.

Readability Score We adopt the readability evaluation method proposed by Chen et al. (2025), which employs an LLM as a judge to assign a readability score on a five-point scale to each predicted chart image.

Accessibility Scores We follow Miranda-Pena et al. (2024) with two accessibility scores: text colour contrast and text spacing. These metrics evaluate the readability of chart text by measuring the contrast between text and background colours, as well as character spacing.

6 Benchmarking Visualisation Agents

In this work, we used our dataset to benchmark three visualisation systems: two general-purpose LLM-powered coding agents, Codex (Co)⁴ and Claude Code (Cl)⁵, and one dedicated text-to-vis pipeline that constructs the visualisation through separate components, LIDA (Dibia, 2023). For the underlying LLMs, we selected models that reflect common deployment scenarios. The commercial models GPT-5 and Opus-4.7 process data externally with fewer hardware constraints, whereas the open models GPT-OSS-20B, Gemma-4-31B, and Qwen-3.6-35B run locally for privacy-sensitive or resource-limited deployments. Additional details about the experimental setup are provided in Appendix F. The overall performance of the systems is summarised in Table 2.

6.1 Execution Performance

Basic execution is largely solved, except for the small local model. As shown in Table 2, task completion and chart-type accuracy are near-ceiling for every strong system, completing 99–100% of samples and chart-type accuracy between 0.90 and 0.97. The exception is Co-GPT-OSS-20B, a small local model, which completes only 76%, 67%, and 60% of GitHub-T2V, arXiv-T2V, and OWID-T2V, respectively. This means task completion remains discriminating for small local models, which still fail to reliably produce valid visualisations.

Displayed data similarity remains far from solved. Displayed data similarity and correctness are the metrics that separate systems, and here every system falls well short of the ceiling. The best score across all data subsets is 0.81 (Cl-Opus-4.7 on arXiv-T2V), and all systems degrade sharply on OWID-T2V, where the top score is only 0.69.

Frontier models separate on correctness and efficiency. Although Co-GPT-5 and Cl-Opus-4.7 complete all tasks, Cl-Opus-4.7 achieves consistently higher correctness, improving by around 5 points across the three subsets, while also running 20-43% faster. Cl-Opus-4.7 matches or surpasses Co-GPT-5 in quality while achieving substantially lower latency.

System design can matter as much as the underlying model. Although GPT-5 powers both Codex and LIDA, their performance diverges sharply on harder subsets. On OWID-T2V, LIDA-GPT-5 completes only 79% of tasks compared with Co-GPT-5’s 100%, with lower displayed data similarity (0.52 vs. 0.68) and correctness (54.37 vs. 73.67). This shows that the surrounding pipeline—prompt construction, code execution, and iterative refinement—can shape outcomes as much as the model itself, exposing a dimension that LLM-only benchmarks miss.

⁴<https://github.com/openai/codex>

⁵<https://github.com/anthropics/claude-code>

Subset	Model	Task Compl.	Pred. Time	Chart Type	Data Sim.	Correct- ness	Reada- bility	Text Contrast	Text Spacing
GitHub-T2V	Co-GPT-OSS-20B	0.76	66s	0.64	0.49	51.71	4.80	0.90	-0.42
	CL-Qwen3.6-35B	0.96	93s	0.86	0.60	68.06	4.81	0.88	-2.12
	CL-Gemma4-31B	0.99	90s	0.90	0.70	75.41	4.83	0.88	-0.33
	LIDA-GPT-5	0.98	113s	0.90	0.66	71.74	4.71	0.89	-0.58
	Co-GPT-5	1.00	68s	0.90	0.61	69.92	4.85	0.88	-0.98
	CL-Opus-4.7	1.00	39s	0.92	0.68	74.68	4.91	0.88	-2.20
arXiv-T2V	Co-GPT-OSS-20B	0.67	55s	0.62	0.48	42.20	4.54	0.89	-0.96
	CL-Qwen3.6-35B	0.97	231s	0.92	0.73	68.07	4.69	0.88	-3.51
	CL-Gemma4-31B	0.99	124s	0.94	0.78	70.80	4.72	0.86	-1.35
	LIDA-GPT-5	0.94	124s	0.88	0.74	68.03	4.55	0.88	-1.43
	Co-GPT-5	1.00	75s	0.93	0.80	72.36	4.72	0.88	-1.98
	CL-Opus-4.7	1.00	55s	0.93	0.81	77.03	4.83	0.89	-3.02
OWID-T2V	Co-GPT-OSS-20B	0.60	22s	0.56	0.37	38.74	4.66	0.92	-1.18
	CL-Qwen3.6-35B	0.90	228s	0.89	0.54	64.74	4.67	0.87	-3.29
	CL-Gemma4-31B	0.99	132s	0.95	0.62	72.29	4.85	0.88	-1.64
	LIDA-GPT-5	0.79	132s	0.77	0.52	54.37	4.60	0.92	-2.17
	Co-GPT-5	1.00	78s	0.97	0.68	73.67	4.74	0.90	-2.07
	CL-Opus-4.7	1.00	63s	0.97	0.69	78.80	4.76	0.90	-3.58

Table 2: Performance of text-to-vis systems. "Co", "CL", and "LIDA" denote Codex, Claude Code, and LIDA, respectively. Readability and accessibility (text contrast, text spacing) are computed only over successfully generated chart images; the number of images therefore varies across systems according to their task completion rate.

Subset	Model	Task Compl.	Pred. Time	Chart Type	Data Sim.	Correct- ness	Reada- bility	Text Contrast	Text Spacing
arXiv-T2V-easy (456 samples)	Co-GPT-OSS-20B	0.66	23s	0.62	0.53	45.75	4.60	0.89	-0.82
	CL-Qwen3.6-35B	0.98	215s	0.93	0.80	73.84	4.73	0.88	-3.31
	CL-Gemma4-31B	0.99	102s	0.95	0.84	75.34	4.74	0.86	-0.95
	LIDA-GPT-5	0.97	137s	0.93	0.84	75.02	4.55	0.88	-0.97
	Co-GPT-5	1.00	66s	0.95	0.85	76.28	4.72	0.89	-1.30
	CL-Opus-4.7	1.00	38s	0.93	0.86	79.67	4.84	0.89	-2.61
arXiv-T2V-hard (349 samples)	Co-GPT-OSS-20B	0.69	97s	0.61	0.43	37.55	4.46	0.90	-1.13
	CL-Qwen3.6-35B	0.97	251s	0.90	0.63	60.51	4.65	0.88	-3.76
	CL-Gemma4-31B	0.99	153s	0.92	0.69	64.87	4.69	0.87	-1.86
	LIDA-GPT-5	0.91	107s	0.81	0.61	58.86	4.55	0.87	-2.03
	Co-GPT-5	1.00	87s	0.91	0.72	67.21	4.71	0.87	-2.84
	CL-Opus-4.7	1.00	76s	0.90	0.73	73.58	4.83	0.89	-3.50

Table 3: Performance on easy and hard splits of arXiv-T2V.

Open models rival frontier systems on easy charts but fail on hard ones. CL-Gemma4-31B is the strongest system on GitHub-T2V (correctness 75.41; displayed data similarity 0.70). It stays close on arXiv-T2V, but underperforms on OWID-T2V; CL-Qwen3.6-35B follows the same trajectory. These results shows our subsets vary in difficulty, with OWID-T2V the most discriminative for displayed data similarity.

Latency remains a practical challenge. Even the fastest quality systems take tens of seconds per sample, and several are considerably slower: CL-Qwen3.6-35B requires up to 231s on arXiv-T2V, and the multi-stage LIDA-GPT-5 exceeds 110s on every subset. From a user-experience perspective, this latency is a meaningful obstacle to interactive data exploration, and it is a dimension on which systems with comparable output quality still differ substantially.

6.2 Readability & Accessibility Evaluation

Across all systems, readability scores exceed 4.5 out of 5, indicating consistently readable charts, with Opus-4.7 scoring highest. Text contrast is likewise high for every system, at roughly 0.86–0.92 on a $[0, 1]$ scale. Text spacing, however, is negative for all systems, from -3.58 to -0.33 on a $(-\infty, \infty)$ scale, where scores above zero denote adequate spacing (Miranda-Pena et al., 2024). This shows that generated charts tend to pack text more tightly than recommended. Since this metric sums the per-word-pair spacing over the whole image, systems that render more text (e.g., the Claude Code pairings) accumulate more negative scores, and values worsen on the denser arXiv-T2V and OWID-T2V subsets.

6.3 Performance on Easy and Hard Splits of arXiv-T2V

We further evaluate all systems on the easy and hard difficulty splits of arXiv-T2V. The easy split contains samples where the input data is already structured and ready for direct visualisation, while the hard split requires additional preprocessing steps such as data transformation, simulation, or reformatting prior to plotting. The assignment of samples to the easy and hard splits was decided by expert annotators. As shown in Table 3, all systems exhibit degraded performance on the hard split, reflected by consistently longer prediction time and lower scores in chart type accuracy, displayed data similarity, chart correctness, readability, and text spacing.

6.4 Other Results

Human Request vs. Machine-Generated Request We also constructed machine-generated request variants. As shown in Table 9 in Appendix G, readability and accessibility metrics remain largely unaffected. Notably, Co-GPT-5, the experimental model, exhibits significant improvements under machine-generated requests for GitHub-T2V and arXiv-T2V. Because machine-generated requests tend to be more explicit, they can inflate performance on those subsets; human-authored requests offer a more realistic and challenging evaluation setting.

Correlation Analysis We further conduct correlation analyses to examine the behaviour of the visualisation systems, with details in Appendix H. Overall, we observe no significant correlations between displayed data similarity and factors such as the length of the natural language request, chart type, data source, or the length of data manipulation steps. Similarly, readability scores show no meaningful correlation with the number of displayed data points.

7 Conclusion

This paper introduces VisCodeBench, a comprehensive benchmark for evaluating text-to-vis systems from an end-to-end perspective. VisCodeBench is comprised of three subsets, GitHub-T2V, arXiv-T2V, and OWID-T2V, which cover educational, scientific, and public-interest visualisation scenarios with varying levels of complexity. Each sample is designed to reflect realistic data visualisation workflows, including data loading, transformation, and chart generation with respect to human natural language requests.

We benchmarked three visualisation systems—Codex, Claude Code, and LIDA—paired with a range of open and commercial LLMs, corresponding to different deployment constraints. We evaluated them using seven metrics: task completion rate, execution time, chart type accuracy, displayed data similarity, chart correctness, readability, and accessibility. Our results show that strong systems now largely solve basic execution, with only the small local model struggling to reliably produce valid charts. Content fidelity, however, remains unsolved, as no system reproduces the reference data well. VisCodeBench separates frontier models: they differ on chart correctness score and efficiency, with ClaudeCode-Opus-4.7 gaining about 5 points in chart correctness over Codex-GPT-5 and running around 20 seconds faster, making Opus-4.7 the strongest model overall. Our results also show that a system’s design can matter as much as its underlying model, and that latency remains a practical obstacle to interactive use. VisCodeBench will serve as a valuable resource for advancing research on reliable and practical text-to-vis systems.

Ethics Statement

This research was conducted under the approved ethics application "2024/HE000844" by the Research Ethics Committee at The University of Sydney. In conducting the study, we obtained informed consent from all participants to publish the collected data under the CC BY-SA 4.0 license. For external data sources such as GitHub and Our World in Data (OWID), all data used in this study clearly indicated permissive copyright and licensing terms. Lastly, we note that users of our datasets may obtain different visualisations or results when applying their own methods, due to variations in data processing and visualisation techniques.

Acknowledgments

This project is partially supported by the Australian Research Council through a Discovery Early Career Researcher Award, the Commonwealth Scientific and Industrial Research Organisation (CSIRO), and an unrestricted gift from Google. We also acknowledge all participants of our online survey. Their contributions will be made available at <https://github.com/SAIPLL/VisCodeBench>. We extend our gratitude to the anonymous reviewers for their constructive feedback and valuable advice on our submissions.

References

- Nan Chen, Yuge Zhang, Jiahang Xu, Kan Ren, and Yuqing Yang. Viseval: A benchmark for data visualization in the era of large language models. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):1301–1311, 2025. doi: 10.1109/TVCG.2024.3456320.
- Kenny Davila, Bhargava Urala Kota, Srirangaraj Setlur, Venu Govindaraju, Christopher Tensmeyer, Sumit Shekhar, and Ritwick Chaudhry. Icdar 2019 competition on harvesting raw tables from infographics (chart-infographics). In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1594–1599, 2019. doi: 10.1109/ICDAR.2019.00203.
- Victor Dibia. LIDA: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models. In Danushka Bollegala, Ruihong Huang, and Alan Ritter (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 113–126, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-demo.11. URL <https://aclanthology.org/2023.acl-demo.11/>.
- G. Farin. *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide*. Computer science and scientific computing. Academic Press, 2014. ISBN 9781483296999. URL <https://books.google.com.au/books?id=CM3SBQAQBAJ>.
- Yucheng Han, Chi Zhang, Xin Chen, Xu Yang, Zhibin Wang, Gang Yu, Bin Fu, and Hanwang Zhang. Chartllama: A multimodal llm for chart understanding and generation. *arXiv preprint arXiv:2311.16483*, 2023.
- Paul A. Harris, Robert Taylor, Robert Thielke, Jonathon Payne, Nathaniel Gonzalez, and Jose G. Conde. Research electronic data capture (redcap)—a metadata-driven methodology and workflow process for providing translational research informatics support. *Journal of Biomedical Informatics*, 42(2):377–381, 2009. ISSN 1532-0464. doi: <https://doi.org/10.1016/j.jbi.2008.08.010>. URL <https://www.sciencedirect.com/science/article/pii/S1532046408001226>.
- Paul A. Harris, Robert Taylor, Brenda L. Minor, Veida Elliott, Michelle Fernandez, Lindsay O’Neal, Laura McLeod, Giovanni Delacqua, Francesco Delacqua, Jacqueline Kirby, and Stephany N. Duda. The redcap consortium: Building an international community of software platform partners. *Journal of Biomedical Informatics*, 95:103208, 2019. ISSN 1532-0464. doi: <https://doi.org/10.1016/j.jbi.2019.103208>. URL <https://www.sciencedirect.com/science/article/pii/S1532046419301261>.

- Taylor Lewis and Karl Hess. The effect of alternative e-mail contact timing strategies on response rates in a self-administered web survey. *Field Methods*, 29(4):351–364, 2017. doi: 10.1177/1525822X17715865. URL <https://doi.org/10.1177/1525822X17715865>.
- Guozheng Li, Xinyu Wang, Gerile Aodeng, Shunyu Zheng, Yu Zhang, Chuangxin Ou, Song Wang, and Chi Harold Liu. Visualization generation with large language models: An evaluation. *arXiv preprint arXiv:2401.11255*, 2024.
- Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. In Kevin Knight, Ani Nenkova, and Owen Rambow (eds.), *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 110–119, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1014. URL <https://aclanthology.org/N16-1014/>.
- Tianqi Luo, Chuhan Huang, Leixian Shen, Boyan Li, Shuyu Shen, Wei Zeng, Nan Tang, and Yuyu Luo. nvbench 2.0: Resolving ambiguity in text-to-visualization through stepwise reasoning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. doi: 10.52202/085713-4172. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/b5f985cdc1796defe83a6b2f84fe2ab6-Paper-Datasets_and_Benchmarks_Track.pdf.
- Yuyu Luo, Nan Tang, Guoliang Li, Chengliang Chai, Wenbo Li, and Xuedi Qin. Synthesizing natural language to visualization (nl2vis) benchmarks from nl2sql benchmarks. In *Proceedings of the 2021 International Conference on Management of Data, SIGMOD '21*, pp. 1235–1247, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383431. doi: 10.1145/3448016.3457261. URL <https://doi.org/10.1145/3448016.3457261>.
- Paula Maddigan and Teo Susnjak. Chat2vis: Generating data visualizations via natural language using chatgpt, codex and gpt-3 large language models. *IEEE Access*, 11:45181–45193, 2023. doi: 10.1109/ACCESS.2023.3274199.
- Clarissa Miranda-Pena, Andrew Reeson, Cécile Paris, Josiah Poon, and Jonathan K. Kummerfeld. Do LLMs generate creative and visually accessible data visualisations? In Tim Baldwin, Sergio José Rodríguez Méndez, and Nicholas Kuo (eds.), *Proceedings of the 22nd Annual Workshop of the Australasian Language Technology Association*, pp. 12–29, Canberra, Australia, December 2024. Association for Computational Linguistics. URL <https://aclanthology.org/2024.alta-1.2/>.
- Hy Nguyen, Xuefei He, Andrew Reeson, Cecile Paris, Josiah Poon, and Jonathan K. Kummerfeld. Do text-to-vis benchmarks test real use of visualisations? In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7433–7441, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.423. URL <https://aclanthology.org/2024.emnlp-main.423/>.
- Alexander Novikov, Ngan Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.
- Fatemeh Pesaran Zadeh, Juyeon Kim, Jin-Hwa Kim, and Gunhee Kim. Text2Chart31: Instruction tuning for chart generation with automatic feedback. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 11459–11480, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.640. URL <https://aclanthology.org/2024.emnlp-main.640/>.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. 2019.

- Mizanur Rahman, Md Tahmid Rahman Laskar, Shafiq Joty, and Enamul Hoque. Text2Vis: A challenging and diverse benchmark for generating multimodal visualizations from text. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 31849–31874, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1622. URL <https://aclanthology.org/2025.emnlp-main.1622/>.
- Yutong Shao and Ndapa Nakashole. ChartDialogs: Plotting from Natural Language Instructions. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 3559–3574, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.328. URL <https://aclanthology.org/2020.acl-main.328/>.
- Yuanfeng Song, Xuefang Zhao, and Raymond Chi-Wing Wong. Marrying dialogue systems with data visualization: Interactive data visualization generation from natural language conversations. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2733–2744, 2024.
- Arjun Srinivasan, Nikhila Nyapathy, Bongshin Lee, Steven M. Drucker, and John Stasko. Collecting and characterizing natural language utterances for specifying data visualizations. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI ’21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380966. doi: 10.1145/3411764.3445400. URL <https://doi.org/10.1145/3411764.3445400>.
- Renqiu Xia, Hancheng Ye, Xiangchao Yan, Qi Liu, Hongbin Zhou, Zijun Chen, Botian Shi, Junchi Yan, and Bo Zhang. Chartx & chartvln: A versatile benchmark and foundation model for complicated chart reasoning. *IEEE Transactions on Image Processing*, 2025.
- Zhiyu Yang, Zihan Zhou, Shuo Wang, Xin Cong, Xu Han, Yukun Yan, Zhenghao Liu, Zhixing Tan, Pengyuan Liu, Dong Yu, Zhiyuan Liu, Xiaodong Shi, and Maosong Sun. MatPlotAgent: Method and evaluation for LLM-based agentic scientific data visualization. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11789–11804, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.701. URL <https://aclanthology.org/2024.findings-acl.701/>.

A Limitations

Limited evaluated models: Our benchmarking is limited to seven text-to-vis systems, with two proprietary LLMs: GPT-5 and Opus-4.7. This is primarily due to practical cost constraints. A single run on three data splits costs 175 USD for Co-GPT-5 and 780 USD for CL-Opus-4.7.

Static, single-turn evaluation: VisCodeBench evaluates a single pass—data and request in, one chart out—and does not capture the iterative, multi-agent workflows used by many modern text-to-vis systems. We believe this setup remains meaningful and complementary: a strong system should still produce accurate results quickly in a single turn, yet even Co-GPT-5 and CL-Opus-4.7 still struggle with this setting.

Reliance on LLM-assisted requests : For arXiv-T2V and OWID-T2V, initial requests are drafted by GPT-5 and revised by experts, which may weaken the claim that they reflect natural human phrasing. We also release arXiv-T2V-Author, which includes original author-written requests as supplementary data.

Task	Precision	Recall	F1-score
Email extraction	1.0000	0.9676	0.9835
Plot classification	0.9846	0.9846	0.9846

Table 4: Performance of GPT-4o-mini in the email extraction task (232 repositories) and plot classification task (1,819 images).

# Papers for experiment	300
# Papers with TeX repository	232
# Papers with first author’s email	177
# Papers with + first author’s email + at least a plot image	134

Table 5: The number of papers through each filtering step.

B Online Survey for arXiv-T2V

B.1 Target Demographic

Our target participants are authors of papers from academic repositories (e.g., arXiv and the ACL Anthology) that are licensed under CC BY-SA 4.0, as well as researchers who can be reached through social media platforms and academic mailing lists. To identify the most relevant contributors, we applied several filters. These include selecting papers that contain visual plots, having email addresses listed in the manuscript, and maintaining a balance between easily made charts and more challenging ones. Using these selections, we identified approximately 35k target participants from 120k papers published on arXiv in 2024.

B.2 Identifying Contactable Papers

We employed a four-step process to identify contactable papers, including email information and plot images. First, we downloaded the TeX repository and metadata of published computer science papers from the arXiv system in 2024⁶. Second, we used several heuristics and GPT-4o-mini⁷ to extract the first author’s email address from the main TeX content, as we believe that the first authors are the primary contributors. Third, for papers containing the contact address, we followed the {figure} sections in the main TeX to locate attached images in the repository and classified them as plots or non-plots using GPT-4o-mini. Finally, papers with both an extracted email address and at least a plot image would be considered contactable papers.

Table 4 describes the performance of GPT-4o-mini in the email extraction task on 232 TeX repositories, and the plot classification task on 1,819 figure images, with ground truths made by our experts. Overall, the LLMs’ performance in both tasks is sufficient for our purpose of identifying contactable papers. Additionally, we instructed the LLM to extract only certain cases and leave uncertain ones blank for the email extraction task, resulting in 100% precision to avoid contacting the wrong individuals.

Table 5 reports the number of papers remaining after each filtering step. Starting from 300 arXiv papers, 232 have an associated TeX repository; from these, we extract the first author’s email for 177, and 134 of those additionally contain at least one plot image. A paper is considered contactable only if it has both an extractable email and at least one plot, so roughly 134 of every 300 papers (44.67%) are contactable for data submission.

⁶<https://arxiv.org/list/cs/recent>

⁷<https://platform.openai.com/docs/models/gpt-4o-mini>

B.3 Platform and Distribution

We used a dedicated email address to communicate with participants throughout the study. Microsoft Outlook was used to manually respond to inquiries, while an SMTP server configured with AWS SES enabled the automated distribution of mass invitation emails. Each invitation included a brief overview of the project, information about the benefits of participation, and a link to a REDCap survey that contained a consent form and a data submission form.

We conducted the survey in two rounds. In the first round, we sent invitation emails to approximately 5,678 people who were the first authors of 2024 arXiv papers in the Computation and Language (cs.CL) category. In the second round, we contacted around 25,842 people who were the first authors of papers in other computer science categories, such as cs.CV (Computer Vision and Pattern Recognition), cs.LG (Machine Learning), and cs.AI (Artificial Intelligence). To further increase participation, we also promoted the project via social media and academic mailing lists within and beyond our university.

Each round allowed participants around 34 days to respond, with the option to withdraw their submitted data at any time before the dataset's release. To encourage participation, two reminder emails were sent: the first two weeks after the initial invitation, and the second three to five days before the survey deadline. To maximise response rates, reminders were scheduled for Tuesday, Wednesday, or Thursday mornings (Lewis & Hess, 2017). As a result, these reminders significantly boosted the submission rates in two rounds of the survey, as shown in Figure 4.

Regarding rewards for each round of the survey, the first 50 participants with successful submissions, providing correct and relevant information, received a A\$20 Amazon gift card. Additionally, each successful submission entered the participant into a lucky draw, where ten winners each received a A\$100 Amazon gift card.

Study data were collected and managed using REDCap electronic data capture tools hosted at The University of Sydney (Harris et al., 2009; 2019). REDCap (Research Electronic Data Capture) is a secure, web-based software platform designed to support data capture for research studies, providing 1) an intuitive interface for validated data capture; 2) audit trails for tracking data manipulation and export procedures; 3) automated export procedures for seamless data downloads to common statistical packages; and 4) procedures for data integration and interoperability with external sources.

B.4 Responses

Figure 4 shows the number of submissions collected via the email channel, where invitation emails were sent directly to participants. We received submissions from 304 participants in the first round and 544 in the second, corresponding to response rates of approximately 5.3% and 2.1%, respectively⁸. In contrast, the social media channel, where invitations were posted on social media platforms and academic mailing lists, produced only four submissions. In total, we collected 1,151 valid submissions, each of which includes underlying data files, a natural language request for visualisation, code files, and a resulting visualisation image. Note that a single participant may contribute multiple submissions, with each one corresponding to a different visualisation in their papers.

B.5 Recruitment of Expert Annotators

We recruited two expert annotators from an NLP course at our university after the course had concluded. The recruitment process included a brief demographic questionnaire and a qualification task, which attracted 32 applicants. From these, we selected two candidates with over four years of coding experience and strong, well-motivated interest in the project. One annotator was a Master's student with experience in both academic and industry settings, while the other was a PhD student with several years of teaching experience.

⁸We count only participants who completed data submission; responses without submitted data are excluded.

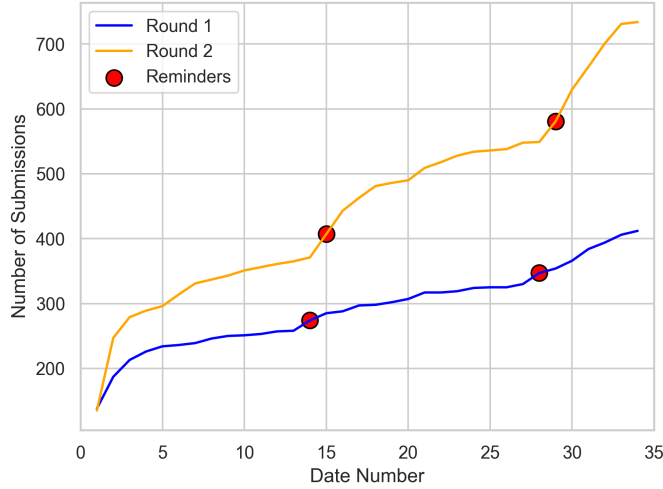


Figure 4: # Submissions by date

Annotators were paid at a rate of approximately A\$49.15 per hour. Prior to annotation, both experts carefully studied the annotation guidelines, spent one week familiarising themselves with the task, and actively asked clarification questions before proceeding with annotating the arXiv dataset.

C Comparison with real-world code

For this comparison, we focused exclusively on the Matplotlib library, building on the data used in our prior work (Nguyen et al., 2024), Matplotlib-nb. Matplotlib also dominates our datasets, appearing in 95% of the samples. This focus is further supported by the fact that both Pandas and Seaborn build on Matplotlib, resulting in similar rendering behaviour across these libraries.

Comparison of Chart type As shown in Figure 5, our dataset subsets exhibit the same trend as the real-world code. Line plots are the most frequently used chart type across all sources, accounting for approximately 40% to 50% of the compared data. Scatter plots are the second most common type, accounting for approximately 20% to 30%. Heatmaps, although present in real-world notebooks (6%), are rarely used in either benchmark subset. Other chart types, such as pie, contour, and stream plots, appear infrequently in all compared data.

Comparison of Attributes Figure 22 presents the normalised frequency of 62 visualisation attributes, with darker colours indicating higher usage. Overall, our dataset subsets show strong overlap in common aesthetic and layout attributes, including titles, axis limits, tick labels, colours, opacity, line styles, annotations, legends, and the use of multiple subplots. Furthermore, as shown in Figure 6, Spearman’s rank correlation coefficients indicate a high degree of similarity between our datasets and real-world visualisation code, with correlation scores exceeding 0.91.

Comparison of Complexity Table 6 reports the average number of distinct visualisation functions and parameters per code file. GitHub-T2V exhibits a similar level of complexity to real-world code, with averages of 6.39 functions and 9.18 parameters. Notably, arXiv-T2V demonstrates the highest complexity, averaging 9.20 functions and 17.32 parameters per program.

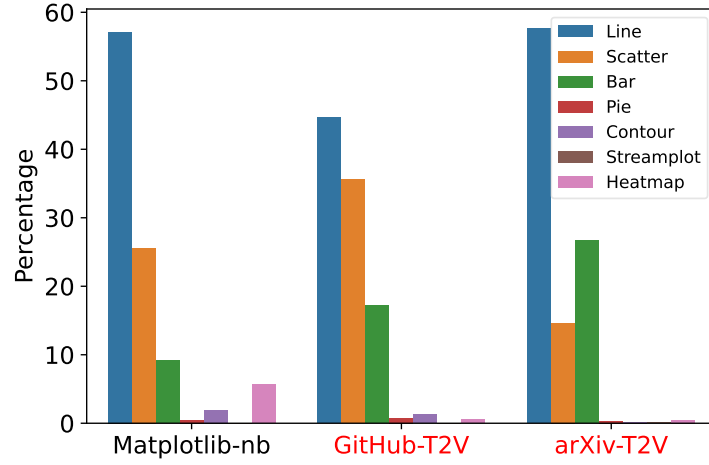


Figure 5: Plot type distribution over three datasets

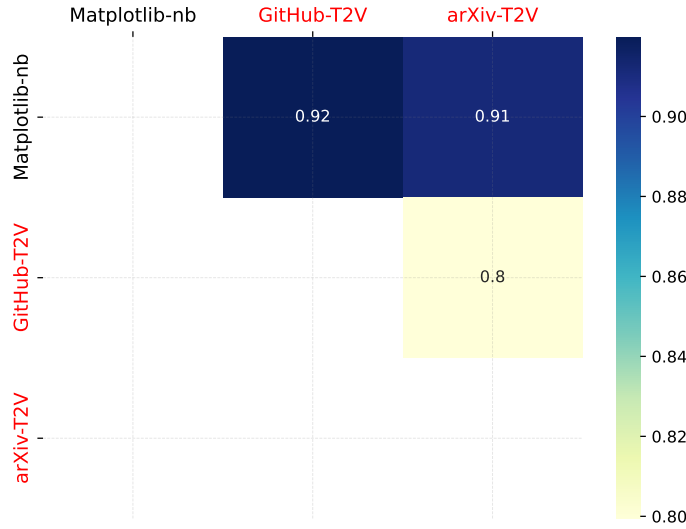


Figure 6: Spearman's rank correlation coefficient in terms of frequent attributes

	No. Funcs	No. Params
Matplotlib-nb	6.19	10.54
GitHub-T2V	6.39	9.18
arXiv-T2V	9.20	17.32

Table 6: Average number of functions and parameters.

D Extraction of Displayed Data Points

We follow a pipeline to extract displayed data points. First, we apply a monkey-patching technique to intercept plotting calls: each time a plotting function is invoked, our logging function is automatically triggered to extract and save the displayed data points (i.e., x and y coordinates) from the axes into separate files. Example of monkey-patch technique on `plt.plot()` function is shown in Figure 7. Table 7 explains the types of extracted coordinates for each chart type.

```

1  √ import matplotlib
2  import matplotlib.pyplot as plt
3
4  # Monkey-patch the plt.plot() method
5  # to log displayed data points
6  original_plot = matplotlib.axes.Axes.plot
7  √ def plot(self, *args, **kwargs):
8      lines = original_plot(self, *args, **kwargs)
9      # Log the displayed data points
10 √   for line in lines:
11       x_data = line.get_xdata()
12       y_data = line.get_ydata()
13       print(f"x: {x_data}, y: {y_data}")
14   return lines
15 matplotlib.axes.Axes.plot = plot
16
17 plt.plot([1, 2, 3], [4, 5, 6])
18
19 √ """
20 Output: Plot data - x: [1 2 3], y: [4 5 6]
21 """

```

Figure 7: Simple example of the monkey-patching technique

Next, we normalise the extracted data points to the range $[0, 1]$, based on the minimum and maximum values along the horizontal and vertical axes of the subplot. Finally, we group the normalised data points by chart type to form data series, which are then used as input for the evaluation script adopted from the CHART-Infographics competition. This grouping is applied to all chart types except line charts, as each line requires the interpolation of data points in the evaluation script.

E Human Study on Displayed Data Similarity

To validate that the Displayed Data Similarity metric aligns with human perception, we conducted a human study comparing the metric against expert judgments. We sampled 200 (predicted, reference) chart pairs and asked two expert annotators to independently rate, on a 0–9 scale, how closely the data displayed in the predicted chart matched the reference. Annotators viewed only the rendered chart images through a custom interface, mirroring how a person would visually assess similarity, while the automatic metric operates on the underlying extracted data points. We then measured the correlation between each annotator’s ratings and the automatic scores.

Table 8 reports the results. Both annotators show a positive and statistically significant correlation with the automatic metric ($n = 200$, $p < 0.001$), confirming that the metric captures the human notion of displayed-data similarity. The correlation is moderate rather than strong, which we attribute to a difference in what each side can perceive. The metric evaluates the full set of extracted data points and solves an optimal point-, series-, and subplot-level assignment, so it registers fine-grained discrepancies such as individual points in dense scatter plots, small deviations along long lines, and errors in secondary series or subplots. Human annotators, judging from rendered images, perceive only coarse and salient features (overall trend, shape, and the dominant series) and cannot resolve these point-level differences. This asymmetry also explains the negative residual mean: annotators systematically rate pairs higher than the metric, over-crediting charts whose small or hidden errors they cannot see.

Chart Type	Extracted Data Description
Pie Plot	Values representing pie segments
Line, Scatter, Stem, Step, Event, Area Plots	x, y coordinates of plotted data points
Bar Plot, Histogram	x, y coordinates of the center-top points of bars
Box Plot, Violin Plot	x, y coordinates of min, max, mean, and the first and third quartiles
Error Bar	x, y coordinates of the top, middle, and bottom of each error bar
2D Histogram, Hexbin, Heatmap	x, y coordinates of cells with corresponding values (z)
Contour Plot	x, y, z coordinates along contour lines
Geographic Map	x, y, z coordinates. x, y represent country borders while z represents the color magnitude

Table 7: Explanation of extracting x, y coordinates for each chart type

	Annotator 1	Annotator 2
Pearson r	0.558	0.415
Spearman ρ	0.557	0.392
Kendall τ	0.452	0.323
Residual mean	-0.045	-0.184

Table 8: Correlation between expert ratings and the automatic Displayed Data Similarity score ($n = 200$). Residual mean is the average of (human – automatic) scores after rescaling to a common range.

F Setting up Visualisation Systems

We benchmark three visualisation systems: two general-purpose coding agents, Codex CLI and Claude Code, and LIDA, a dedicated text-to-vis pipeline.

Coding agents. For each text-to-vis sample, we organised all required resources, including the underlying data and any dependent files, into a dedicated folder. We then created a code template within this folder that contains the context code and a pair of special markers (##START VISUALISATION CODE and ##END VISUALISATION CODE) indicating where the agent should insert the generated Python code. This design is inspired by AlphaEvolve, in which a coding agent reasons over the provided context and writes code within predefined regions (Novikov et al., 2025). After preparing the folder and template, we invoked the agent to complete the visualisation task with the natural-language visualisation request embedded in the prompt. Codex CLI and Claude Code share this setup, differing only in the agent binary and the API each uses (OpenAI for Codex, the Anthropic Messages API for Claude Code). The structure of the per-sample codebase, the Python template, and the prompt used for the coding agent are illustrated in Figure 1 (left), while an example of running the agent is described on the right of the figure. Further details of each agent’s configuration and prompts can be found in our repository.

LIDA. We adapt LIDA (Dibia, 2023) to our benchmark. Our implementation differs from the original in the kind of input it consumes: the original LIDA takes a single tabular file and lets the system decide what to visualise, whereas ours takes the same per-sample folder used for the coding agents—a scaffolded Python script with the two markers, a natural-language request, and any accompanying data files—and completes the missing visualisation code. The three-stage design of the original pipeline is preserved: (1) *data summarisation*, where the model probes the sample folder by writing and executing a short exploration script whose output serves as the summary, rather than profiling a single dataframe; (2) *goal*, which is read verbatim from the user’s request rather than invented by the model; and (3) *visualisation generation*, where the model completes the code between the two markers and the full script is executed to produce the chart images. The underlying idea is therefore identical to the original—visualisations are treated as executable code that must run, be inspectable, and be repairable—while the surrounding data handling is made flexible enough to accommodate the free-form, multi-file samples in our benchmark.

Model deployment. We accessed the commercial models (GPT-5, Opus-4.7) through their providers’ APIs; for example, GPT-5 was served via the OpenAI portal <https://api.openai.com/v1>. The open models (GPT-OSS-20B, Gemma-4-31B, Qwen-3.6-35B) were deployed locally on a machine whose GPU has sufficient memory to support the model’s maximum context length. For instance, GPT-OSS-20B was served with Ollama on a RunPod instance with an RTX 4090 GPU, supporting a maximum context length of 120k tokens.

G Human requests vs machine requests

In addition to GitHub-T2V, arXiv-T2V, and OWID-T2V, which include human-generated natural language requests, we created corresponding versions featuring machine-generated requests produced by GPT-5. The word limits for these versions were adjusted to align with their respective real-world contexts: 50 words for GitHub-T2V and OWID-T2V, and 120 words for arXiv-T2V. These variants enable controlled comparisons of coding-agent performance under human versus machine-generated requests within the same data source. Looking at the performance of the coding agents on readability and accessibility metrics, we observe no substantial differences between human-authored and machine-generated request settings.

In contrast, the content metrics exhibit notable gaps, as summarised in Table 9. Co-GPT-5 improves under machine-generated requests on GitHub-T2V and arXiv-T2V, most markedly on GitHub-T2V, with relative gains of about 6% in chart type accuracy, 43% in displayed data similarity, and 23% in correctness. OWID-T2V, by contrast, shows no improvement and a slight drop in displayed data similarity, likely because its analytically complex charts require intent that a short machine-generated request cannot capture as well as a carefully written human request. These results suggest that more explicit machine-generated requests can inflate performance on educational and scientific charts, whereas human-authored requests provide a more realistic and challenging evaluation.

H Correlation Analysis

This section examines how the displayed data similarity metric correlates with several factors, including the length of the natural language request, chart type, data source, and the number of data manipulation steps. It also examines the relationship between chart readability and the number of data points displayed. We mainly present results for Co-GPT-5 across three dataset splits: GitHub-T2V, arXiv-T2V, and OWID-T2V.

H.1 Analysis 1 - NL Request Length vs Displayed Data Similarity

Figure 8 presents a histogram of the length in words of the natural language requests pooled across the three subsets: GitHub-T2V, arXiv-T2V, and OWID-T2V. The distribution is

Subset	Task Compl.	Pred. Time	Chart Type	Disp. Data	Correctness	Readability	Text Contrast	Text Spacing
<i>GitHub-T2V</i>								
Human-Req	0.998	68s	0.897	0.609	69.92	4.847	0.882	-0.984
Machine-Req	0.991	52s	0.951	0.870	86.19	4.843	0.879	-1.331
<i>arXiv-T2V</i>								
Human-Req	1.000	75s	0.934	0.796	72.36	4.717	0.882	-1.979
Machine-Req	0.998	74s	0.948	0.814	77.40	4.683	0.882	-1.661
<i>OWID-T2V</i>								
Human-Req	0.998	78s	0.973	0.685	73.67	4.739	0.896	-2.072
Machine-Req	0.999	75s	0.973	0.659	73.10	4.746	0.895	-2.271

Table 9: Performance of Co-GPT-5 with human vs. machine-generated requests.

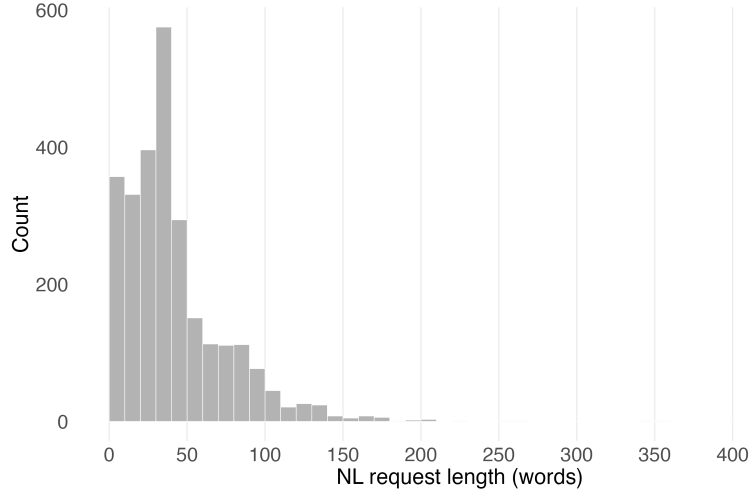


Figure 8: Histogram of human-generated natural language request length (in words) pooled across GitHub-T2V, arXiv-T2V, and OWID-T2V.

right-skewed, with most requests around 50 words in length. Figure 9 presents histograms of the length in words of these requests broken down by dataset. Separate panels show a) the bulk of requests (≤ 200 words), and b) the few long requests (> 200 words), which were present in the GitHub-T2V and arXiv-T2V.

The length of the bulk of requests is distributed differently across the datasets. The arXiv-T2V requests tend to be longer, with wider variance in the request length. GitHub-T2V requests are skewed to shorter request lengths, with most requests shorter than 60 words. The OWID-T2V requests are tightly clustered around 30-40 words.

Natural language request length was not meaningfully associated with similarity score for any of the runs investigated. For the arXiv-T2V subset, we found a non-significant Pearson coefficient, indicating no linear association between request length and similarity score ($r = -0.19$, $p = 3.76e-08$). The Spearman coefficient, while significant, was close to zero ($\rho = -0.225$, $p = 1.16e-10$), suggestive of only a weak monotonic association. This pattern was replicated across the other combinations of subsets and agents. There was thus no clear or consistent relationship between natural language request length and similarity score. Scatter plots for similarity score versus NL request length in words for each dataset-agent combination are presented in Figure 10.

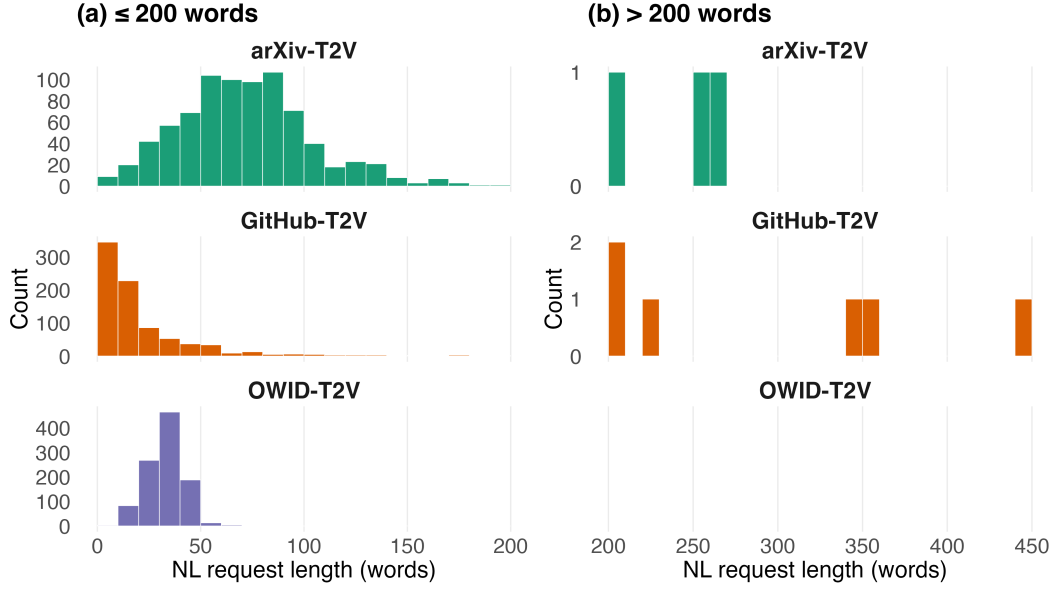


Figure 9: Histogram of human-generated natural language request length (in words) for each dataset: GitHub-T2V, arXiv-T2V, and OWID-T2V.

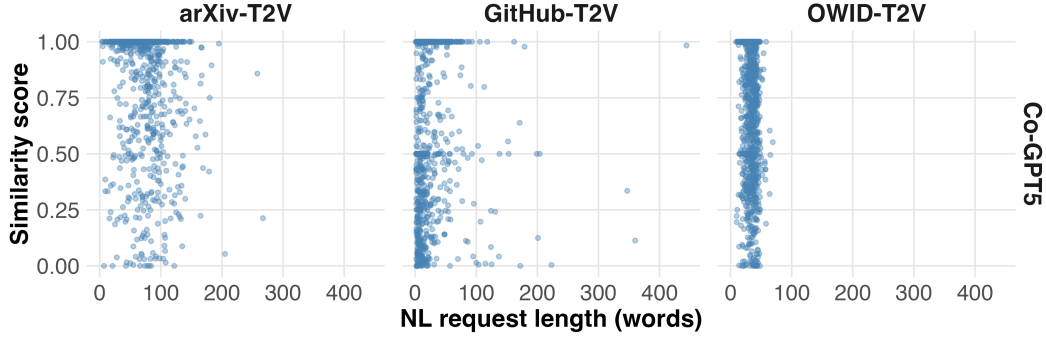


Figure 10: Scatter plots of similarity score versus NL request length for each dataset-agent combination (GitHub-T2V, arXiv-T2V, OWID-T2V; Co-GPT-5)

H.2 Analysis 2 - Data Source Type vs Displayed Data Similarity

In the arXiv-T2V and GitHub-T2V examples, the data for plotting is accessible from different sources. These sources are broadly separated into four categories:

- **File:** Where the plotting code reads in an external file (e.g., a .csv or .json file), which contains the data for plotting.
- **Manual:** Where the data for plotting is manually specified/embedded into the code file (e.g., a list or dictionary of values).
- **Simulation:** Where the data to plot is simulated using random numbers or some mathematical function is used to generate the data.
- **Other:** Where a plot falls into more than one of the above categories (e.g., a figure with two subplots, where one subplot falls under the 'File' category and the other falls under the 'Simulation' category), or none of the above categories (e.g., the data for plotting comes from a library's built-in datasets, such as Scikit-learn's Iris dataset).

You will be given a code snippet to generate a plot. Your task is to act as a classifier to determine what source of data is used for the underlying plot data. You can choose four options:

- "File" for when the original data to plot is stored in a file. For example, the data for plotting is predominantly stored in a file (e.g., the data is read in from a file `"data = open('file', 'r')"`). It should still be considered the "File" class where a file is loaded in, and then processing is done to the file's contents. It should also be considered the "File" class where most of the data is in the file, but some extra data is manually appended.

- "Simulation" for when the data to plot is simulated or randomly generated. This class is assigned where mathematical functions are applied to generate the data (e.g., we start with `torch.ones((1, 64, dim))` and then apply calculations). It could also involve random number generation.

- "Manual" for when the data to plot is provided directly in the file. For example, the data for plotting may be provided in lists (e.g. `x = [1,2,3,4,5]`, `y = [-1,-2,-3,-4,-5]`) or dictionaries. Where the data is manually specified, but simulation is done using that data, it is classified as the "Simulation" class.

- "Other" when you believe the other class labels are inappropriate. In this case, you must also provide a brief explanation of why it does not fit under the "Other" classifications. An example is where there are two subplots, and one fits the "File" class and the other the "Manual" class.

Only return the class label, and nothing else. If it is the "Other" label, write "Other" first, followed by your reason. Please think carefully before providing your answer.

Figure 11: GPT-5 prompt used for the data source classification in [H.2](#)

Data Source	arXiv-T2V	GitHub-T2V
File	56.27%	75.09%
Simulation	6.58%	18.62%
Manual	36.02%	1.30%
Other	1.12%	4.98%

Table 10: Proportion of data source classifications for different datasets.

The OWID-T2V data was not included in this analysis as its data always comes from a file.

To classify the data examples, we prompted GPT-5 (prompt in [Figure 11](#)) and validated the approach on a small set of manually labeled samples. We found that it correctly classified all examples. [Table 10](#) presents the proportion of different data sources for the examples in each data subset. "File" is clearly the most common data source, with "Manual" and "Other" contributing a far smaller proportion.

[Figure 12](#) shows the similarity scores for different data sources separated by data subsets and model type. There is a large spread across all data source measures. 'Manual' appears to have the least variability while having the greatest performance, yet has a long left-skew.

While 'File' has a higher median similarity score when compared to 'Simulation' and 'Other' in arXiv-T2V, this relationship does not hold with the GitHub-T2V data, where 'File', 'Simulation', and 'Other' seem to have similar medians and spreads.

H.3 Analysis 3 - Data Manipulation vs Displayed Data Similarity

In most arXiv-T2V, GitHub-T2V and OWID-T2V examples, the read-in data is manipulated before plotting. This includes, but is not limited to, data filtering, unit conversions, aggregation, merging different data frames, and applying dimension reduction (e.g., t-SNE). To quantify the number of data manipulation operations that have taken place for a given example, we prompted GPT-5 (the prompt in [Figure 13](#)) to count the number of data manipulation operations.

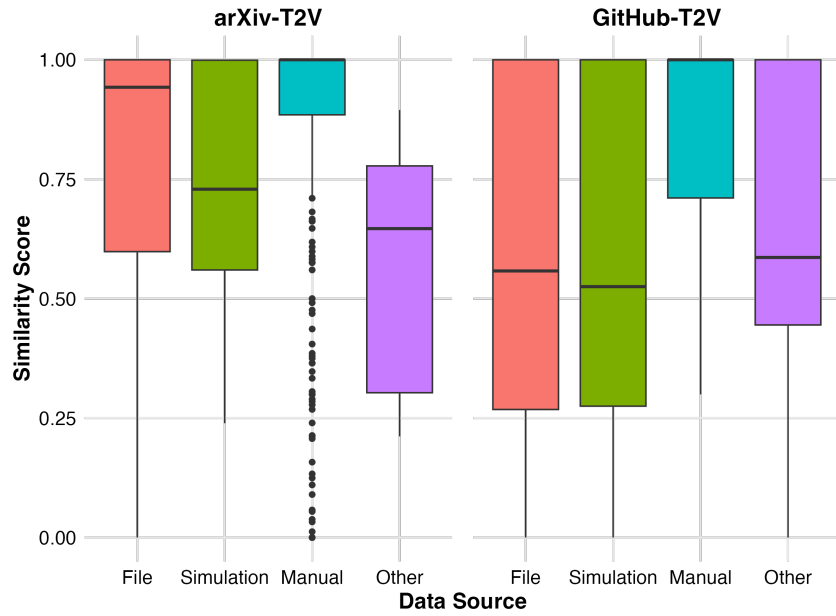


Figure 12: Similarity score vs different data sources for different data subsets separated by model configuration.

You will be given a code snippet that loads, processes, and plots data. Your task is to count the number of distinct data manipulation operations performed before plotting.

Trace the data processing steps and identify each unique data manipulation operation. Examples include:

- Filtering rows
- Type casting columns
- Unit conversions
- Renaming categories
- Data aggregation
- Merging datasets
- Appending values
- Dimensionality reduction (e.g., TSNE)
- Generating synthetic data

Do NOT count data loading (e.g., reading a CSV) as data manipulation. If the code only loads data and plots it with no intermediate processing, return 0.

If the language is not Python, return null.

Return only your answer in the required JSON structure:

```
{
  "num_steps": <integer or null>
}
```

Figure 13: GPT-5 prompt used to count the number of data manipulations for a given example in [H.3](#)

Figure 14 shows the number of data manipulation steps for each example as determined by GPT-5. Combining all datasets, the average number of data manipulation steps is 5.73 ($s = 6.72$).

Figure 15 shows scatter plots of data manipulation steps against the similarity score for the different data subsets and models. There appears to be no strong linear relationship between data manipulation steps and similarity score.

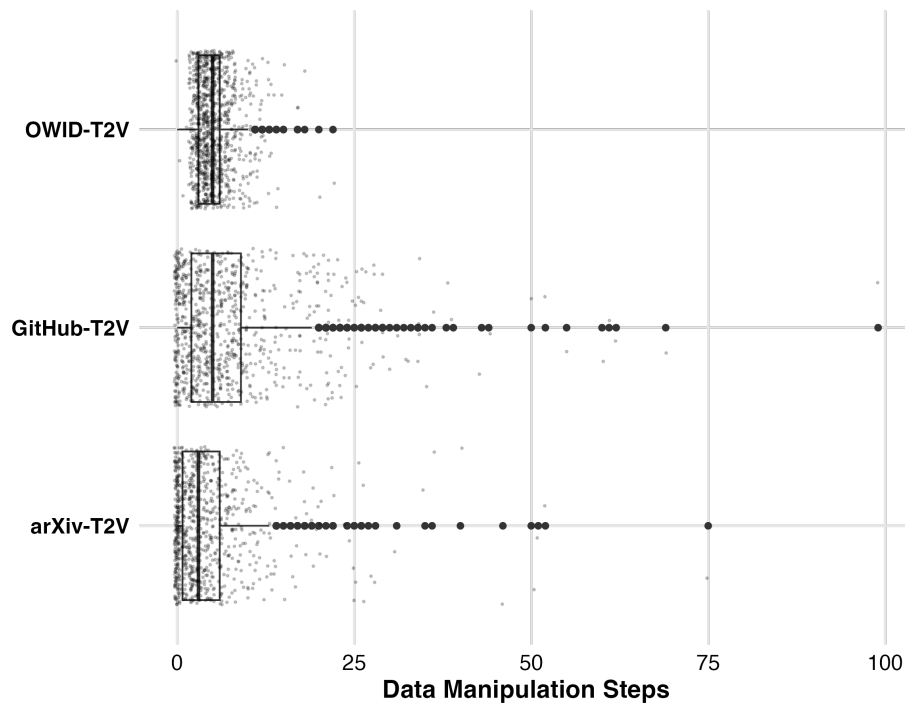


Figure 14: Number of data manipulation steps for arXiv-T2V, GitHub-T2V and OWID-T2V examples.

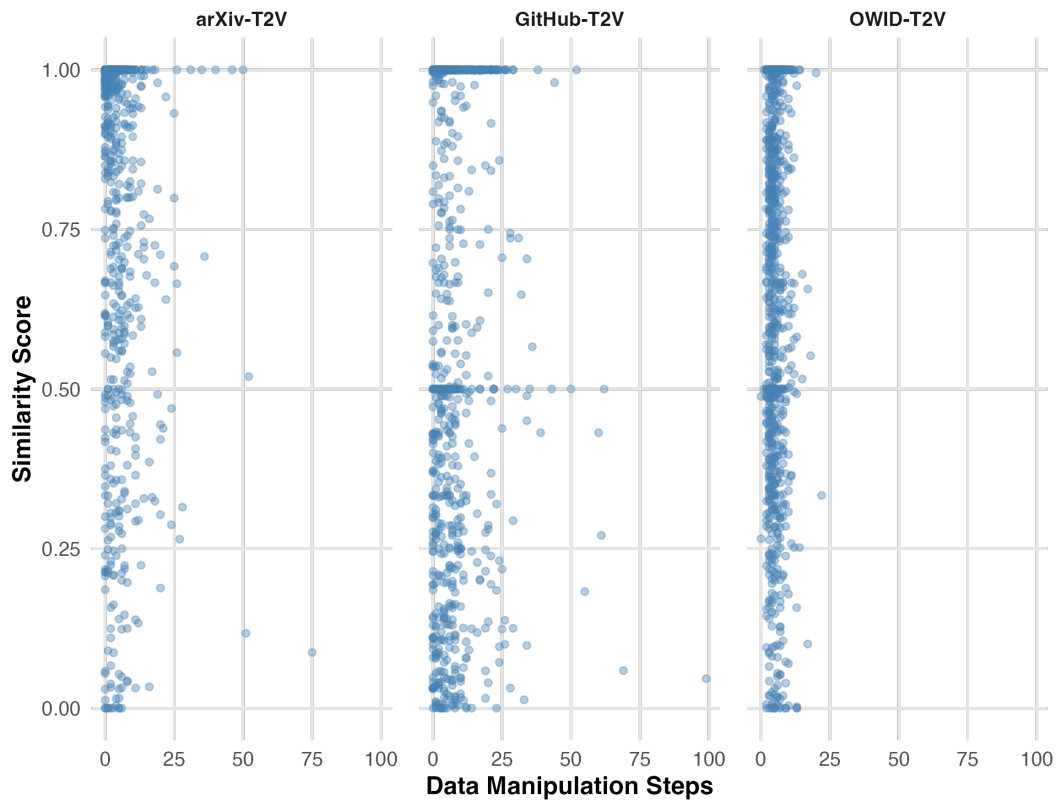


Figure 15: Data manipulation steps against similarity score for different datasets and models.

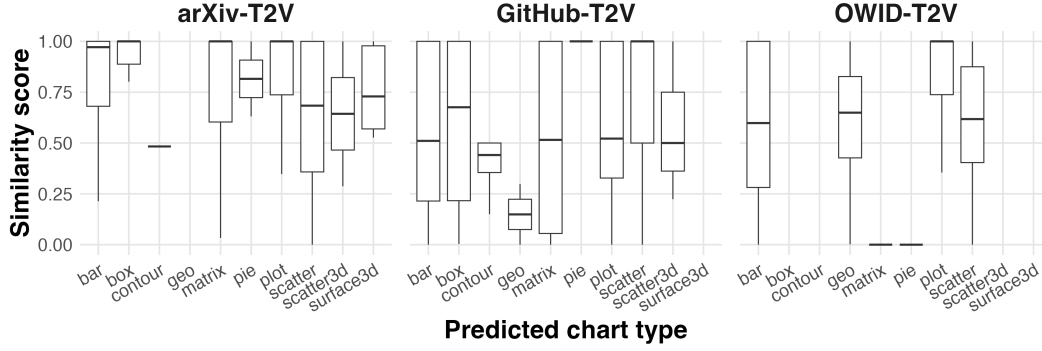


Figure 16: Similarity score across different chart types for each dataset-agent combination (GitHub-T2V, arXiv-T2V, OWID-T2V; Co-GPT-5)

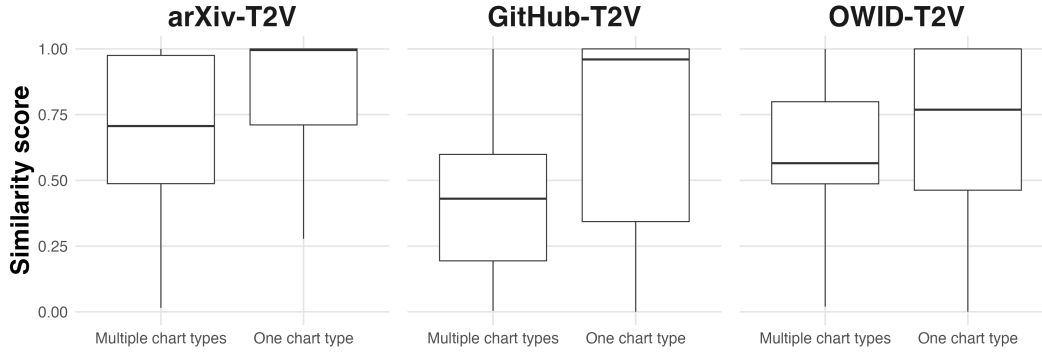


Figure 17: Similarity score for one versus multiple predicted chart types for each dataset-agent combination (GitHub-T2V, arXiv-T2V, OWID-T2V; Co-GPT-5)

H.4 Analysis 4 - Chart Type vs Displayed Data Similarity

Similarity vs Primary Chart type Across all three dataset-agent combinations tested with the Co-GPT-5 agent, a Kruskal-Wallis test found significant differences in similarity score across chart types. For the GitHub-T2V subset ($N = 842$), $H(6) = 35.07$, $p = 4.17 \times 10^{-6}$; for the arXiv-T2V subset ($N = 801$), $H(5) = 51.28$, $p = 7.57 \times 10^{-10}$; and for the OWID-T2V subset ($N = 1021$), $H(3) = 196.83$, $p = 2.04 \times 10^{-42}$. Note that, due to sample size constraints, not all chart types pictured in Figure 16 were used in statistical testing. A similarity score that differs across primary predicted chart types indicates that some figures may be more difficult to plot clearly across datasets and models. As shown in Figure 16, simpler charts, such as box plots and scatter plots, tend to have higher similarity scores across datasets and models.

Similarity for One versus Multiple Chart Types Figure 17 compares the similarity score for visualisations with one predicted chart type to the score for visualisations with multiple predicted chart types, for the first run of each dataset-agent combination tested. For all experiment settings, visualisations with one chart type tended to have a higher similarity score than visualisations with multiple chart types. Mann-Whitney U tests found the difference between one and multiple chart types was significant for all experiment settings (for the arXiv-T2V dataset with Co-GPT-5 agent: $U = 58,299$, $p = 3.76e - 10$; $N_{\text{one}} = 671$, $N_{\text{multiple}} = 130$). This suggests that visualisations requiring multiple chart types are more difficult to create across datasets and agents.

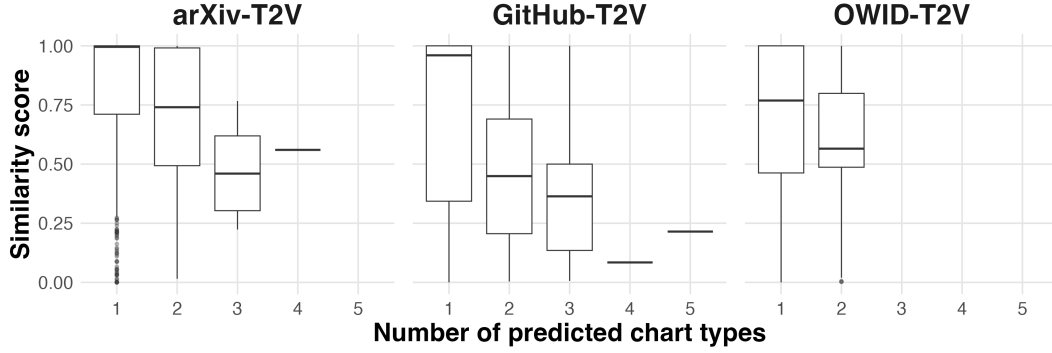


Figure 18: Similarity score versus number of chart types predicted for each dataset-agent combination (GitHub-T2V, arXiv-T2V, OWID-T2V; Co-GPT-5)

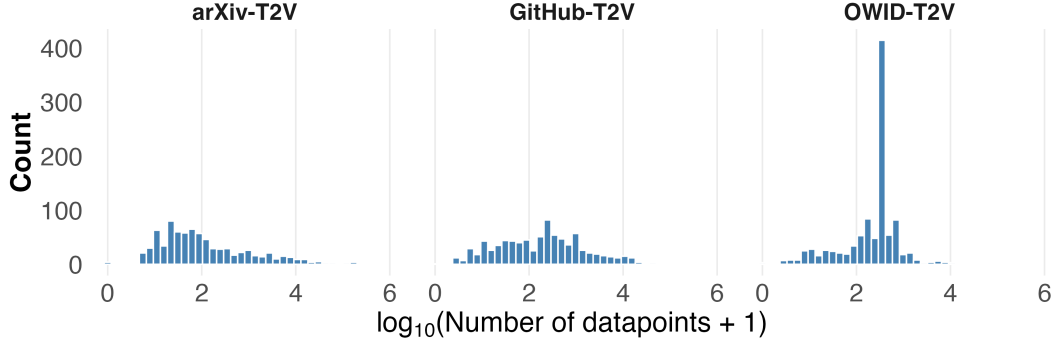


Figure 19: Logged distribution of number of data points in visualisations produced in run 1 for each dataset-agent combination (GitHub-T2V, arXiv-T2V, OWID-T2V; Co-GPT-5)

Similarity Score versus Number of Chart Types Figure 18 shows the similarity score across the number of predicted chart types, for the first run of each dataset-agent combination tested. As the number of predicted chart types increases, the similarity score tends to decrease. This observation was confirmed with significant negative associations between similarity score and the number of chart types found. For the arXiv-T2V dataset with Co-GPT-5, both Pearson and Spearman coefficients were significant ($r = -0.193, p = 3.77e - 08$; $\rho = -0.225, p = 1.16e - 10$). This was true of all other dataset-agent combinations tested.

H.5 Analysis 5 - Readability vs Number of Data Points

Figure 19 shows the distribution of the logged number of data points in the visualisations produced in response to human natural language requests for the first run on 1) GitHub-T2V, 2) arXiv-T2V, and 3) OWID-T2V subsets using Co-GPT-5 agent. OWID-T2V figures tend to contain more data points than other subsets, with a sharp spike at 300 data points reaching a count of around 400, suggesting many datasets in this source have a very similar, standardized number of data points.

Figure 20 shows the distribution of the readability scores (1-5) of these same generated visualisations. Across both datasets and models, readability scores are skewed to the high end of the scale. However, the arXiv-T2V scores are more dispersed between 4 and 5, while the GitHub-T2V and OWID-T2V scores are more concentrated at 5.

The number of data points in a visualisation was not meaningfully associated with the readability score for any of the runs investigated. For the arXiv-T2V dataset using human NL requests Co-GPT-5 agent, we found no significant Pearson coefficient, suggesting there

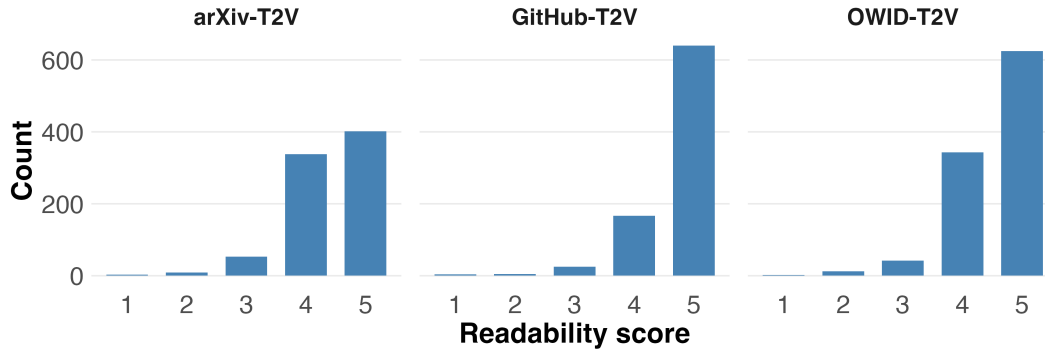


Figure 20: Distribution of readability scores for visualisations produced in run 1 for each dataset-agent combination (GitHub-T2V, arXiv-T2V, OWID-T2V; Co-GPT-5)

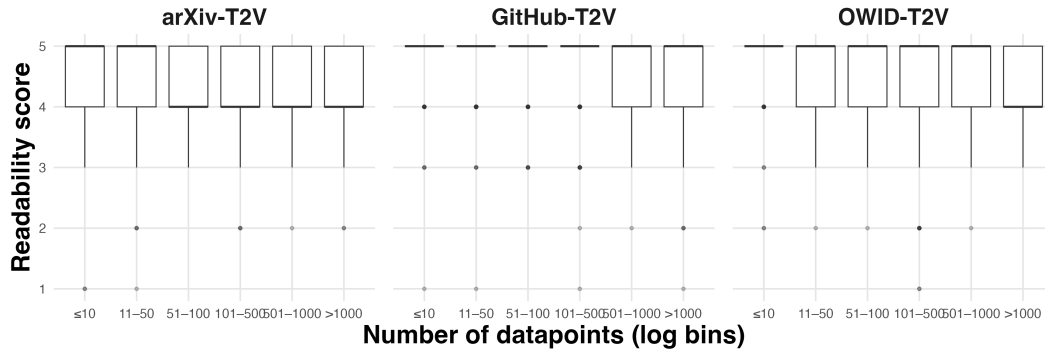


Figure 21: Readability score across logged bins of number of data points in visualisations produced in run 1 for each dataset-agent combination (GitHub-T2V, arXiv-T2V, OWID-T2V; Co-GPT-5)

was no linear association between request length and similarity score ($r = -0.039$, $p = 0.269$). The Spearman coefficient, while significant, was close to zero ($\rho = -0.153$, $p = 1.38e-05$), suggestive of only a weak monotonic association. This pattern was replicated across the other combinations of datasets and agents: Pearson coefficients were non-significant, and where Spearman correlations were significant, they were close to zero and not consistent in direction. There was thus no clear or consistent relationship between the number of data points in a visualisation and the readability score. Readability score across logged bins of the number of data points in a visualisation for each dataset-agent combination investigated is presented in Figure 21.

Attribute	Matplotlib-nb	GitHub-T2V	arXiv-T2V
x-title	8.5	10.0	3.6
y-title	8.6	9.0	4.1
x-y-title-fontsize	2.7	1.2	3.7
x-y-title-color	0.2	0.0	0.2
x-y-lim	2.4	0.9	2.2
x-y-ticks-labels	3.1	1.6	4.6
x-y-ticks-labels-color	0.1	0.0	0.3
x-y-ticks-labels-rotation	0.7	0.4	0.8
x-y-scale	0.4	0.1	0.2
x-y-ticks-fontsize	0.9	0.1	3.1
x-axis-ticks-visible	0.7	1.0	1.9
y-axis-ticks-visible	0.4	0.2	0.9
x-y-scale-position	0.2	0.0	0.5
invert-x-y-axis	0.1	0.0	0.0
filled-color	8.5	12.9	9.1
edge-color	0.7	0.7	1.0
opacity	2.4	3.4	2.5
linewidth	2.5	6.2	3.5
markersize	1.8	2.5	1.9
linestyle	1.0	3.5	2.6
line-capstyle	0.0	0.0	0.0
markerstyle	1.4	2.7	2.9
bar-thickness	0.4	0.3	1.9
bar-data-stacking	0.0	0.0	0.1
hist-bins	1.0	1.4	0.1
pie-explode	0.1	0.0	0.0
pie-label-distance	0.0	0.0	0.0
pie-percentage-distance	0.0	0.0	0.0
pie-precision-digits	0.1	0.1	0.0
pie-radius	0.0	0.0	0.0
errbar-cap-size	0.1	0.0	0.2
errbar-cap-thick	0.0	0.0	0.0
errbar-color	0.0	0.0	0.1
errbar-visible	0.4	0.3	0.4
ann-text-label	0.5	0.3	0.9
ann-fontsize	0.8	2.8	2.0
ann-position	0.9	0.0	3.4
ann-font	0.1	0.0	0.3
title	8.8	8.3	1.8
title-fontsize	1.2	0.4	0.7
title-color	0.0	0.0	0.0
title-position	0.1	0.0	0.0
subtitle	0.5	0.4	0.1
subtitle-fontsize	0.2	0.1	0.1
legend-title	0.1	0.4	0.2
legend-fontsize	0.5	0.1	1.6
legend-position	2.3	1.7	3.6
legend-labels	8.2	6.4	7.2
legend-labels-color	0.0	0.0	0.0
legend-is-display	5.2	3.9	3.7
grid-visible	0.7	0.1	1.1
grid-color	0.1	0.0	0.3
grid-linestyle	0.2	0.0	1.0
grid-linewidth	0.1	0.0	0.5
size	8.6	7.4	4.5
dpi	0.9	0.2	3.0
saving-format	0.2	0.0	0.4
bounding-box-border	0.0	0.0	0.0
background	0.1	0.0	0.1
margin-padding	3.1	1.5	7.4
multiple-plots	7.4	7.4	3.5

Figure 22: Heat map of the most frequent aesthetic attributes over 4 datasets. The attributes are classified by different categories with colours, such as x and y axes , data appearance , annotation , title and subtitle , legend , grid , figure format , and others .