ChartReasoner: Code-Driven Modality Bridging for Long-Chain Reasoning in Chart Question Answering

Anonymous ACL submission

Abstract

Recent advances in large language models (LLMs) have significantly improved long-chain reasoning in textual domains, yet extending this capability to visual tasks such as chart-based question answering (ChartQA) remains a major challenge. Existing multimodal approaches often rely on lossy image-to-text conversions that obscure critical structural and semantic information embedded in visualizations. To address this gap, we propose ChartReasoner, a code-driven, two-stage framework designed to enable precise, interpretable reasoning over charts. In the first stage, we train Chat2Code, a high-fidelity model that converts diverse chart images into structured ECharts code, preserving both layout and data semantics. In the second stage, we leverage these symbolic representations to construct ChartReasoning, the first large-scale chart reasoning dataset containing 140K multi-step samples. We then train the final reasoning model using a combination of supervised fine-tuning and reinforcement learning. ChartReasoner achieves strong performance across four representative benchmarks: ChartQA, ChartBench, EvoChart-QA, and ChartQAPro. It performs competitively with state-of-the-art open-source models while using fewer parameters, and approaches the performance of proprietary systems like GPT-4o in out-of-domain setting. Our results demonstrate that symbolic code-driven modeling provides a scalable and effective path toward deep, multimodal reasoning over visual data.

1 Introduction

LLMs have achieved remarkable success in text-based long-chain reasoning, generating highly accurate structured, multi-step solutions to complex problems, exemplified by models like o1 (OpenAI, 2024), o3 (OpenAI, 2025), QwQ (Team, 2025) and DeepSeek-R1 (Guo et al., 2025). These models decompose complex problems into logical sequence steps, each building upon previous deductions to reach well-justified conclusions. However, this reasoning capability remains largely confined to the textual domain, creating a significant gap when applied to visual chart interpretation tasks.

Recent advances in multimodal reasoning extend structured thinking from text to vision by converting images into textual representations to enable chain-of-thought (CoT) reasoning. Methods such as R1-OneVision (Chen et al., 2025) translate visual scenes into formal text, while R1-V (Chen et al., 2025) and MMEureka (Meng et al., 2025) leverage reinforcement learning to enhance object-centric and long-chain reasoning. Despite these innovations, visual content is often treated as auxiliary—serialized into language at the cost of losing fine-grained cues. Local structures, color semantics, spatial layouts, and chart-specific encodings are frequently abstracted or compressed. This lossy transformation undermines tasks that require precise visual grounding, such as ChartQA or scientific diagram analysis. Although approaches like Curr-ReFT (Deng et al., 2025) and LMM-R1 (Peng et al., 2025) adopt staged learning to gradually align visual and textual modalities, they still fall short of preserving the high-fidelity semantics inherent in complex visual data.

ChartQA aims to enable models to understand and reason over structured visualizations such as bar and line charts. Recent models have improved visual-text alignment (Masry et al., 2023; Liu et al., 2023), while ChartLlama (Han et al., 2023) and ChartSFT (Meng et al., 2024) introduce chain-of-thought (CoT) prompting for multi-step reasoning. However, most existing ChartQA models still lack true reasoning capabilities. CoT prompting often leads to superficial reasoning without genuine logical depth. A key unresolved challenge is the accurate reconstruction of chart semantics from visual input. Without faithfully extracting symbolic structures—such as axes, legends, groupings, and value mappings—models struggle with multi-hop reasoning and precise numerical comparison. This gap limits their applicability in real-world analytical scenarios that demand deep understanding and logical rigor.

To address the challenges of chart-based understanding and long-chain reasoning, we propose ChartReasoner, a code-driven, two-stage framework that enhances the reasoning capabilities of multimodal large language models (MLLMs). In the first stage, we train Chat2Code, a high-accuracy model that translates diverse chart images into structured ECharts code, faithfully preserving both visual layout and underlying data semantics. This symbolic representation serves as the foundation for reasoning, bridging the visual–textual modality gap. In the second stage, we construct the ChartReasoning dataset by applying Chat2Code to various benchmarks, yielding 140K multi-step reasoning samples. We then train the final ChartReasoner model through supervised fine-tuning and reinforcement learning to improve reasoning accuracy, consistency, and interpretability. This structured pipeline enables precise, scalable, and logically grounded ChartQA.

Our key contributions are as follows:

- We introduce ChartReasoning, the first large-scale chart reasoning dataset with over 140K multi-step reasoning samples. It supports symbolic and interpretable reasoning across diverse chart types, addressing a key gap in ChartQA research.

- We construct a high-quality Chart2Code dataset comprising 110K diverse synthetic charts generated via a prompt-based pipeline. This dataset serves as a critical bridge between visual input and symbolic structure, enabling accurate and interpretable reasoning in downstream tasks.

- We introduce ChartReasoner, a two-stage code-driven model that demonstrates robust performance on four representative benchmarks: ChartQA, ChartBench, EvoChart-QA, and ChartQAPro. Our model performs competitively with state-of-the-art open-source systems using fewer parameters, and rivals proprietary models like GPT-4o in out-of-domain settings, demonstrating its effectiveness and generalizability.

2 Related Work

ChartQA. To improve MLLMs' ability to understand charts, various ChartQA datasets have been introduced, including FigureQA (Kahou et al., 2017), DVQA (Kafle et al., 2018), PlotQA (Methani et al., 2020), LEAF-QA (Chaudhry et al., 2020), and ChartQA (Masry et al., 2022), covering diverse chart types and visual reasoning tasks. However, these datasets often limit answers to single values or labels, lacking support for complex multi-step reasoning. Recent efforts like ChartX (Xia et al., 2024), RealCQA (Ahmed et al., 2023), and UniChart (Masry et al., 2023) scale up resources via synthetic chart generation and template-based QA, while ChartSFT (Meng et al., 2024) and EvoChart (Huang et al., 2025a) incorporate Chain-of-Thought (CoT) annotations to promote reasoning. However, most CoTs remain shallow, whereas real-world scenarios demand long, multi-hop reasoning. On the model side, compact MLLMs like ChartReader (Cheng et al., 2023), MatCha (Liu et al., 2023), ScreenAI (Baechler et al., 2024), and UniChart (Masry et al., 2023) perform well on earlier benchmarks. LLaVA-based models such as ChartLlama (Han et al., 2023), ChartPaLI (Carbune et al., 2024), ChartInstruct (Masry et al., 2024), ChartAstD (Meng et al., 2024), and TinyChart (Zhang et al., 2024) further enhance multimodal alignment. More recently, open-source generalist VLMs like Phi-3 Vision and InternVL2.5 (Chen et al., 2024a) have achieved strong results on ChartQA benchmarks. However, current models still struggle with long-chain reasoning, particularly when integrating multiple visual cues and performing numerical and logical inference.

Chart-to-Code. Chart-to-code generation aims to reconstruct charts from images via executable code, demanding accurate visual and structural fidelity. Early works such as ChartMimic (Yang et al., 2025a), Plot2Code (Wu et al., 2025), and ChartX (Xia et al., 2024) evaluate MLLMs on layout and content reconstruction. While some methods enhance generation via multi-agent collaboration or preference-based tuning (Li et al., 2025; Zhang et al., 2025), they often rely on handcrafted prompts or costly supervision. ChartCoder (Zhao et al., 2025) advances the field with a two-stage SoT-based training strategy and a large-scale dataset, but its reliance on fixed templates still limits its ability to generalize to diverse, real-world chart formats.

Multimodal Long-Chain Reasoning. Long-chain reasoning has gained traction in NLP with the advent of DeepSeek-R1 (Guo et al., 2025), which emphasizes structured intermediate reasoning. This paradigm has been extended to VLMs through works like R1-OneVision (Yang et al., 2025b) and Vision-R1 (Huang et al., 2025b), which convert images into formal textual representations to enable multimodal CoT training. R1-V (Chen et al., 2025) leverages Group Relative Policy Optimization (GRPO) (Shao et al., 2024) for object counting, showing that small models can outperform larger ones via effective RL. VisualThinker-R1-Zero (Zhou et al., 2025) and MMEureka (Meng et al., 2025) further explore RL-driven reasoning, reporting "visual aha moments" where longer outputs indicate stronger reasoning. Meanwhile, Curr-ReFT (Deng et al., 2025) and LMM-R1 (Peng et al., 2025) adopt staged learning strategies that progressively integrate visual and textual skills, using reward curricula or text-first pretraining followed by multimodal RL. However, despite these advancements, existing approaches primarily focus on natural images or general visual inputs and fall short when applied to structured data representations like charts. In contrast, our work targets the unique challenges of chart-based long-chain reasoning by introducing a code-driven framework that explicitly bridges visual perception with symbolic reasoning, offering a specialized and scalable solution for real-world analytical tasks.

3 Methodology

Understanding charts poses a fundamental challenge for MLLMs, stemming from the modality gap between raw visual inputs and the structured, symbolic semantics of chart elements. To bridge this gap and enable deep, interpretable reasoning, we propose a code-driven two-stage framework that unifies visual perception and symbolic abstraction via structured chart representations.

In the first stage, we introduce Chart2Code, a high-fidelity translation model that converts chart images into executable ECharts code. By leveraging the expressive and structured syntax of ECharts, the model preserves both the visual layout and the semantic structure of charts, including axes, legends, data groupings, and value mappings. To train this model, we construct a 110K-scale synthetic dataset using a prompt-based generation pipeline built on DeepSeek-R1, where chart specifications are rendered into images and paired with code. We apply a hybrid filtering strategy to ensure data quality, and fine-tune a Qwen2.5-VL-based model on these image–code pairs, freezing the visual encoder to retain robust perception while adapting the decoder for accurate symbolic generation.

In the second stage, we leverage Chart2Code to build ChartReasoning, a large-scale dataset comprising 140K multi-step reasoning samples. These samples are created by applying structured code extraction to existing ChartQA benchmarks and prompting DeepSeek-R1 to generate CoT reasoning traces directly over code. This symbolic representation allows the model to reason over explicit, lossless semantics rather than lossy visual tokens. We then train our final model—ChartReasoner—via a two-stage process: supervised fine-tuning establishes baseline logical competence, followed by GRPO-based reinforcement learning that refines reasoning quality through rule-guided reward signals.

Overall, our approach treats code as a compositional and interpretable bridge between vision and language, enabling precise and logically grounded chart-based reasoning, as illustrated in Figure 1.

3.1 Chart-to-Code

Echarts-format Chart Generation. We start with a template library $\mathcal{T} = \{T_1, T_2, \ldots, T_K\}$ covering K chart templates (9 major categories, 49 subtypes). For each template $T_k \in \mathcal{T}$, we prompt DeepSeek-R1 to generate diverse ECharts code (Detailed prompt is provided in Appendix D). Let p_k be the prompt derived from template T_k. The generated ECharts code c_j for a sample j is:

$$c_j = G_{\text{DS-R1}}(p_k) \tag{1}$$

Where $G_{\text{DS-R1}}(\cdot)$ denote the DeepSeek-R1.

Quality Filtering Pipeline. The generated ECharts code is rendered into images, which are subjected to a rigorous quality control process. We combine automated pixel-level filtering with manual review to enhance image quality. In the automated stage, each image is converted to the Hue-Saturation-Value color space to extract saturation and brightness features, and is downsampled to reduce computational overhead. Blank and noisy images are then removed using sparse content detection and white-background noise filtering. In the manual stage, we further eliminate edge cases that are difficult to detect automatically. As a re-

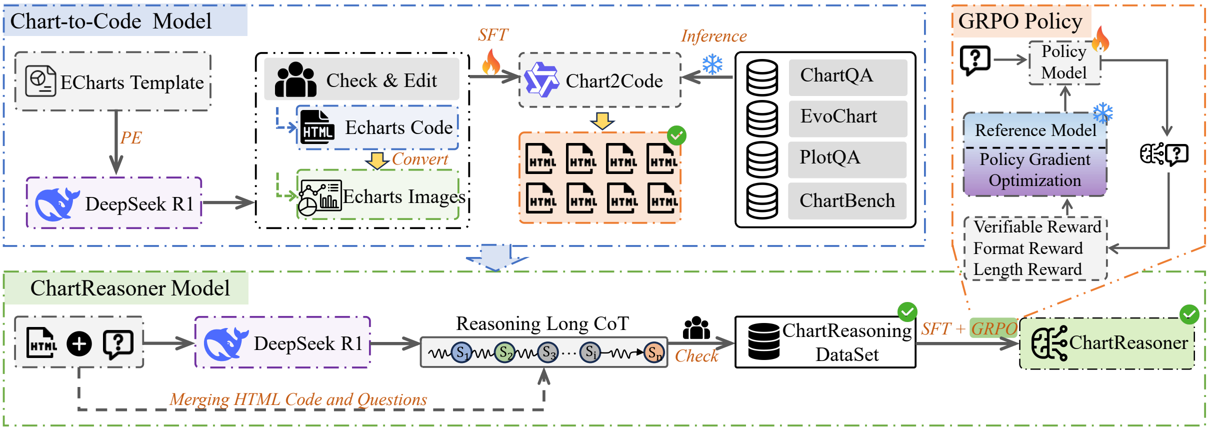


Figure 1: Overview diagram of the data construction pipeline and model training.

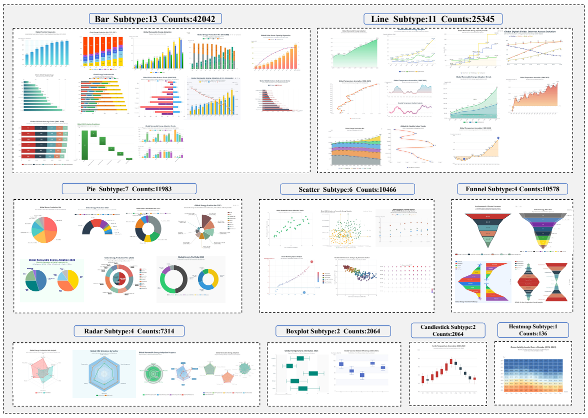


Figure 2: The distribution of chart types and their corresponding subtypes.

sult, we retain approximately 110K high-quality charts from the initial set. Detailed distribution statistics are provided in Figure 2, covering 9 major categories and 49 subcategories. Specific data examples are included in the Appendix E.

Chart2Code Model. To enable high-fidelity chart reconstruction from images, we construct a large-scale chart-to-code dataset and use it to train a multimodal model capable of translating chart images into their corresponding ECharts code. For this purpose, we choose Qwen2.5-VL, a representative open-source vision-language model, and fine-tune it on our dataset for the chart-to-code generation task. Given a chart image x_i, the model predicts its corresponding ECharts code sequence $\mathbf{c_i} = (c_{i,1}, c_{i,2}, \ldots, c_{i,L_i})$, where L_i is the token length of the code. The model is trained to maximize the likelihood of the target sequence conditioned on the input image. The model parameters are denoted as $\theta = \{\theta_{\text{VE}}, \theta_{\text{LD}}\}$, where θ_{VE} refers to the visual encoder (frozen during training), and θ_{LD} denotes the language decoder parameters. The training objective is to minimize the loss function $\mathcal{L}_{\text{C2C}}$, defined as:

$$\mathcal{L}_{\text{C2C}}(\theta_{\text{LD}}) = -\sum_{i=1}^{N_{\text{C2C}}} \sum_{t=1}^{L_i} \log P(c_{i,t} \mid x_i, \mathbf{c}_{i,<t}; \theta) \quad (2)$$

where $\mathbf{c}_{i,<t}$ represents the sequence of previously generated (ground-truth) tokens $(c_{i,1}, \ldots, c_{i,t-1})$ for the i-th sample, N_{C2C} denotes the total number of samples in the chart-to-code dataset.

This training strategy enables the model to effectively extract both structural and semantic information from visual inputs and generate accurate, executable code.

3.2 Code-Driven Long-Chain Reasoning Data

Current Chart QA datasets primarily consist of image-question-answer triplets, lacking explicit annotations of intermediate reasoning steps. This limits their effectiveness in training models that require step-by-step reasoning grounded in chart content. To address this limitation, we construct a code-driven reasoning dataset that extends traditional QA data with model-generated reasoning paths anchored in chart code. The construction pipeline is as follows.

ChartReasoning Construction. We begin by consolidating existing datasets into a unified collection $\mathcal{D}_{\text{QA-orig}} = \{(x_k, q_k, a_k)\}_{k=1}^{N_{\text{orig}}}$, where x_k denotes a chart image, q_k is a question posed about the chart, and a_k is the corresponding ground-truth answer. Each question is categorized by reasoning type, and each chart is labeled according to its structural type. To ensure broad coverage and balanced representation, we perform stratified sampling over both dimensions to obtain a representative subset of samples.

For each selected chart image x_k in this subset, we first employ the trained Chart2Code to generate the corresponding ECharts specification c_k. This generated code, combined with the original question q_k, is then provided as input to the DeepSeek-R1 model. DeepSeek-R1 produces a reasoning path denoted as r_k and a predicted answer $\tilde{a}_k$:

$$(r_k, \tilde{a}_k) = G_{\text{DS-R1}}(\text{Prompt}(\text{Chart2Code}(x_k), q_k)) \quad (3)$$

To ensure data quality, we retain only those samples where the predicted answer $\tilde{a}_k$ exactly matches the ground-truth answer a_k. The final constructed dataset, referred to as ChartReasoning, is defined as follows:

$$\mathcal{D} = \{(x_j, q_j, r_j, a_j)\}_{j=1}^{N} \quad (4)$$

Here, (x_j, q_j, r_j, a_j) represent the chart image, the corresponding question, the generated reasoning path, and the verified answer for the j-th sample, respectively. During training, the input to the reasoning model consists of the chart-question pair (x_j, q_j), while the target output is the concatenated sequence of the reasoning path r_j followed by the final answer a_j.

Data Collection. We construct the ChartReasoning dataset by aggregating and cleaning a wide range of existing ChartQA datasets, including ChartQA (Masry et al., 2022), EvoChart (Huang et al., 2025a), ChartBench (Xu et al., 2023), and PlotQA (Methani et al., 2020). These datasets collectively encompass diverse chart types and question styles commonly found in practical applications.Following the unified code-driven data pipeline introduced earlier, we systematically process all collected data to ensure consistency and correctness. After filtering out low-quality or mismatched samples, we obtain a high-quality subset containing over 140K examples, each paired with verified answers and intermediate reasoning traces.To better understand the dataset composition, we conduct a detailed analysis of the reasoning types and chart structures. As shown in Figure 3, the resulting dataset offers broad coverage across four main reasoning categories and seven commonly used chart types, providing a strong foundation for training models on complex ChartQA.

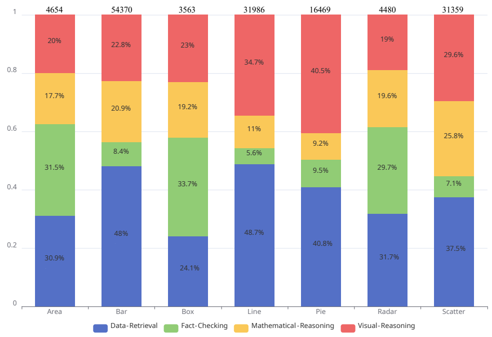


Figure 3: The proportion of different reasoning tasks for each chart type.

3.3 ChartReasoner Training

Supervised Fine-Tuning. The reasoning model is first trained using SFT on the $\mathcal{D}$ dataset. Given a chart image x_j and a question q_j, the model is trained to generate a target output sequence $\mathbf{y}_j = (y_{j,1}, y_{j,2}, \ldots, y_{j,K_j})$, which consists of a reasoning path followed by the final answer, and contains K_j tokens. The model parameters are denoted as $\theta = \{\theta_{\text{VE}}, \theta_{\text{LD}}\}$, where θ_{VE} refers to the visual encoder (kept frozen during training), and θ_{LD} denotes the parameters of the language decoder. The SFT objective is to minimize the loss function $\mathcal{L}_{\text{SFT}}$, defined as:

$$\mathcal{L}_{\text{SFT}}(\theta_{\text{LD}}) = -\sum_{j=1}^{N}\sum_{t=1}^{K_j} \log P(y_{j,t} \mid x_j, q_j, \mathbf{y}_{j,<t}; \theta) \quad (5)$$

where $\mathbf{y}_{j,<t}$ represents the sequence of previously generated (ground-truth) tokens $(y_{j,1}, \ldots, y_{j,t-1})$ for the j-th sample.

This approach improves response uniformity and provides a stable foundation for the subsequent reinforcement learning stage.

Reinforcement Learning with GRPO. While supervised fine-tuning equips the model with fundamental chart understanding, it also reveals a common failure mode: over-generation of verbose reasoning chains, even when the input lacks sufficient information. This over-reasoning behavior compromises answer reliability. To mitigate this, we adopt a reinforcement learning phase using GRPO. Unlike standard policy optimization methods such as PPO (Schulman et al., 2017), GRPO generates multiple candidate responses per input and optimizes them jointly via intra-group normalization. This stabilizes training and encourages the model to favor concise and accurate outputs.

We design structured, rule-based reward functions that explicitly measure answer quality across multiple dimensions—factual accuracy, formatting correctness, and response length. These reward signals guide the model to suppress hallucinations and over-reasoning, promoting disciplined and generalizable reasoning behavior. Overall, this RL phase aligns the model's outputs with practical expectations and user preferences, significantly enhancing robustness across diverse ChartQA scenarios.

4 Experiments

4.1 Experimental Setup

Datasets. To evaluate the performance of our proposed ChartReasoner on ChartQA and multimodal reasoning, we conducted experiments on four representative benchmarks: ChartQA (Masry et al., 2022), EvoChart-QA (Huang et al., 2025a), ChartQAPro (Masry et al., 2025a), and ChartBench (Xu et al., 2023). These datasets cover a broad range of chart types and reasoning tasks, from simple visualizations to complex real-world settings involving dashboards, infographics, and multi-chart compositions. ChartQA and EvoChart-QA emphasize real-world chart understanding with fine-grained reasoning and retrieval tasks, while ChartQAPro focuses on challenging scenarios such as multi-turn, hypothetical, and unanswerable questions. ChartBench provides large-scale evaluation across diverse chart types. We also assessed the Chart2Code on EvoChart-QA to evaluate its capability in reconstructing complex charts.

Evaluation Metrics & Baselines. For ChartQA, we follow the official protocol for each benchmark. For Chart-to-Code, we adopt execution success rate and GPT-4V [1] visual similarity scoring (1–10), following Plot2Code (Wu et al., 2025). The specific prompt is provided in Appendix C We benchmark our model against a wide range of MLLMs, including proprietary models such as Claude-3.5-Sonnet (Anthropic, 2024), Gemini-Flash-1.5/2.0 (Team et al., 2024), GPT-4-turbo, and GPT-4o (Achiam et al., 2023), as well as open-source models like InternVL2 (Chen et al., 2024b), Phi-3-Vision (Abdin et al., 2024), LLaVA-V1.5 (Liu et al., 2024), InternLM-XComposer (Dong et al., 2024), Qwen-VL (Bai et al., 2025; Wang et al., 2024), and CogVLM2 (Hong et al., 2024). We also include domain-specific baselines such as ChartLlama (Han et al., 2023), ChartAst (Meng et al., 2024), ChartIns (Masry et al., 2024), ChartGemma (Masry et al., 2025b), TinyChart (Zhang et al., 2024), and EvoChart (Huang et al., 2025a). For Chart-to-Code evaluation, we adopt ChartCoder (Zhao et al., 2025) as the primary baseline. To further validate the structural richness of our Chart-to-Code dataset, we conduct controlled training experiments using Qwen2.5-VL-7B (Bai et al., 2025) on both EvoChart and our dataset. Further implementation details in Appendix A.

4.2 Main Results

ChartQA Results. We comprehensively evaluate our ChartReasoner model and a wide range of baseline models, including both general-purpose MLLMs and chart-specialized models, across four benchmark datasets. Among them, ChartQA and ChartBench are in-domain datasets, while ChartQAPro and EvoChart-QA serve as out-of-domain evaluations to test generalization performance. The results are shown in Table 1.

In the ChartQA benchmark, the proprietary Claude-3.5-Sonnet model achieves top-tier performance. However, our ChartReasoner significantly outperforms all open-source 7B models and surpasses the majority of chart-specialized baselines, demonstrating its strong reasoning capability in structured visual tasks. A similar trend is observed in ChartBench, where GPT-4o leads among proprietary models, yet our model achieves state-of-the-art results among open-source and domain-specific competitors. These findings confirm that while proprietary models still retain an edge on in-domain datasets, strengthening reasoning and analysis ability can bridge this gap and yield competitive results. In the EvoChart-QA benchmark, which contains long and complex real-world charts, GPT-4o shows relatively weaker performance. In contrast, EvoChart, a chart-specialized model trained on similar data, performs better but shows clear limitations on ChartQA, indicating limited cross-domain generalization due to data-specific overfitting and smaller model scale. Notably, our ChartReasoner matches GPT-4o's performance and outperforms its own base model Qwen2.5-VL, confirming its enhanced capacity for long-chain visual reasoning and data adaptation.Lastly, in the ChartQAPro benchmark, Gemini-Flash-2.0 stands out among proprietary models. Still, ChartReasoner surpasses even GPT-4o in this domain-shifted setting. This reveals that many proprietary models struggle with domain transfer in chart understanding, whereas ChartReasoner's consistent performance under both in-domain and out-of-domain conditions underscores the importance of improv-

[1] The version is gpt-4-vision-preview, and the URL is https://openai.com/index/gpt-4v-system-card.

Model Name	**Size**	**Evochart-QA**	**ChartQA**	**ChartBench**	**ChartQAPro**
		Closed-source			
Claude-3.5-Sonnet (Anthropic, 2024)	–	–	**90.80**	–	43.58
Gemini-Flash-2.0 (Team et al., 2024)	–	–	–	–	**46.85**
Gemini-1.5-Flash (Team et al., 2024)	–	27.90	79.00	–	42.96
Gemini-1.5-Pro (Team et al., 2024)	–	32.20	87.20	–	–
GPT-4-turbo (Achiam et al., 2023)	–	40.30	62.30	–	–
GPT-4o (Achiam et al., 2023)	–	**49.80**	85.70	**59.45**	37.67
		Open-source			
InternVL2-Llama3 (Chen et al., 2024b)	76B	–	**88.40**	–	–
Qwen2-VL (Wang et al., 2024)	72B	–	88.30	–	–
Intern-VL2 (Chen et al., 2024b)	40B	**49.00**	86.20	–	–
CogVLM2 (Hong et al., 2024)	19B	21.90	81.00	–	–
Intern-VL2 (Chen et al., 2024b)	8B	38.60	81.50	–	–
Intern-VL2.5 (Chen et al., 2024a)	8B	–	84.80	–	35.67
LLaVA-v1.5 (Liu et al., 2024)	7B	–	55.32	23.39	–
Internlm-XComp.-v2 (Dong et al., 2024)	7B	–	72.64	47.78	–
QwenVL-Chat (Bai et al., 2023)	7B	19.70	83.00	26.98	35.59
Qwen2.5-VL (Bai et al., 2025)	7B	46.80	85.00	**54.06**	**36.61**
Phi3-Vision (Abdin et al., 2024)	4B	39.50	81.40	–	24.73
		Chart Expert			
ChartLlama (Han et al., 2023)	13B	9.50	69.66	21.71	–
ChartAst-S (Meng et al., 2024)	13B	12.90	79.90	–	–
ChartIns-Llama2 (Masry et al., 2024)	7B	16.80	66.64	–	4.88
EvoChart (Huang et al., 2025a)	4B	**54.20**	81.50	–	–
ChartIns-FlanT5 (Masry et al., 2024)	3B	24.30	64.20	–	–
ChartGemma (Masry et al., 2025b)	3B	30.60	80.16	–	6.84
TinyChart (Abdin et al., 2024)	3B	25.50	83.60	–	13.25
ChartReasoner-SFT(Ours)	7B	47.04	86.76	55.10	37.94
ChartReasoner-GRPO(Ours)	7B	48.10	**86.93**	**55.20**	**39.97**

Table 1: Comparisons of ChartReasoner and Baselines on Four ChartQA Benchmarks.

ing reasoning and abstraction abilities to enhance chart-centric generalization.

Comparing models trained with SFT alone to those further refined with GRPO reveals consistent gains across all benchmarks. GRPO improves reasoning quality and reduces over-explanation and encourages more structured, precise outputs—highlighting its effectiveness in enhancing visual reasoning.

4.3 Ablation Experiment

Chart-to-Code Performance Evaluation. To comprehensively evaluate the effectiveness of our Chart2Code, we present the results in Table 2, which consolidates comparisons across different datasets and training scales. Specifically, we assess model performance on a real-world test set derived from EvoChart-QA, using GPT-4V visual similarity scores and pass rates as evaluation metrics.

Our Chart2Code, trained on the proposed ECharts-based dataset, significantly outperforms models trained on the EvoChart dataset under comparable training sizes. The results highlight notable improvements in both visual fidelity and pass rate, demonstrating the higher quality and diversity of our data. This suggests that our dataset enables better generalization and more accurate chart reconstruction, even for complex and diverse chart types encountered in practice.

In addition, our model trained on ECharts-based data exhibits superior performance compared to those trained on large-scale, Python-generated chart datasets. Despite the latter having access to more training examples, their performance lags behind in both robustness and reconstruction accuracy. This underscores the importance of data realism and expressiveness—qualities more inherently present in ECharts specifications—for effectively training chart generation models.

We also analyze the impact of data volume by training models on subsets of 30k, 50k, 70k, and 110k chart–code pairs. Results show that while increasing the dataset size generally improves performance, the gains begin to plateau beyond 70k examples. This saturation effect suggests that 110k samples are sufficient to maximize the model's reconstruction capability.

Sensitivity Analysis We conduct a sensitivity analysis to evaluate how chart type affects both chart reconstruction and downstream reasoning performance. As shown in Table 2, the Chart2Code module exhibits strong performance on bar and pie charts, while its accuracy declines for scatter

Model	Data	Similarity	bar	line	pie	scatter	Rate	Types
ChartCoder	160k	3.64	4.18	3.91	3.25	3.22	82.40%	27
Chart2Code-Evo.	70k	3.84	4.63	4.16	3.94	2.63	89.10%	4
Chart2Code(Ours)	30k	2.39	3.12	2.24	2.81	1.39	88.20%	**49**
Chart2Code(Ours)	50k	3.62	4.37	3.81	4.17	2.13	90.60%	**49**
Chart2Code(Ours)	70k	4.21	5.17	4.20	4.23	3.24	91.00%	**49**
Chart2Code(Ours)	110k	**4.34**	**5.26**	**4.21**	**5.12**	**3.77**	**92.40%**	**49**

Table 2: A performance comparison of Chart2Code models trained on different datasets in terms of GPT-4V similarity (including specific chart types) and EvoChart-QA pass rates.

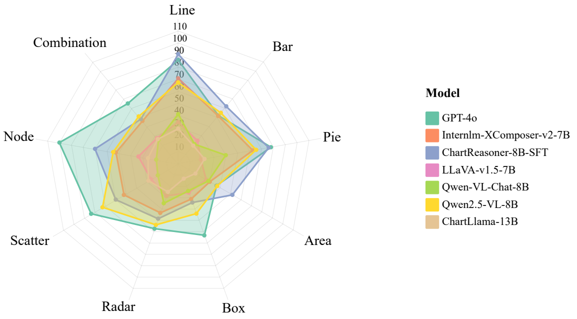


Figure 4: ChartBench Performance Across Chart Types.

and line charts. Scatter plots often contain dense, overlapping points that hinder precise encoding, whereas many line charts in EvoChart are multi-series or include complex visual encodings, making them particularly challenging to parse. These characteristics, along with their relative scarcity in the training data, contribute to consistently lower reconstruction accuracy—especially for line charts—across all models.

This reconstruction quality directly influences reasoning performance in the ChartReasoner module. As shown in Figure 4 and Figure 5, ChartReasoner achieves competitive results on bar and pie charts but underperforms on scatter, line, and box plots. Notably, the stronger results on bar, line, and pie charts within ChartBench align with their higher frequency in our training data, which enhances reconstruction robustness and, in turn, improves reasoning accuracy. These observations highlight a strong correlation between reconstruction reliability and downstream performance, underscoring the importance of both visual complexity and data distribution in building effective chart reasoning systems.

Impact of Different Dataset Sources. We further investigate how different ChartQA-style datasets affect downstream reasoning performance when used to construct CoT-style training data. Specifically, we sample 20k instances from ChartQA, EvoChart, ChartBench, and PlotQA, and convert them into reasoning examples via our chart-to-code distillation pipeline. Results in Table 3 reveal that training data sourced from the same distribution as the evaluation benchmark yields the best performance, confirming the impact of dataset alignment. Notably, PlotQA-based training performs poorly across all benchmarks. This is likely due to its synthetic nature, limited visual diversity, and narrow chart type coverage—restricted to bar, line, and dot—making it less representative of real-world charts. In contrast, EvoChart-derived data achieve stronger generalization, particularly on EvoChart-QA and ChartQAPro. EvoChart charts better resemble real-world styles and include a broader set of chart types, such as pie and scatter, enhancing their cross-domain utility. While ChartBench data yield strong in-domain results, their performance on other benchmarks is less competitive, suggesting limited transferability. Overall, these findings underscore the importance of dataset diversity and visual-semantic complexity in training robust chart reasoning models.

DataSet	Evochart-QA	ChartQA	ChartBench	ChartQAPro
ChartQA	41.3	**86.56**	51.43	35.64
EvoChart	**42.8**	85.48	52.38	**36.05**
ChartBench	40.5	81.56	**54.76**	32.36
PlotQA	40.2	83.00	47.80	34.69

Table 3: Impact of Different Dataset Sources on Downstream Chart Reasoning Performance.

5 Conclusion

We present ChartReasoner, a code-driven, two-stage framework that bridges the gap between visual chart understanding and long-chain reasoning in multimodal large language models. By introducing Chat2Code, which converts chart images into high-fidelity ECharts code, and constructing the ChartReasoning dataset with over 140K multi-step reasoning samples, our approach enables precise, interpretable, and scalable ChartQA. Extensive evaluations across four benchmarks demonstrate that ChartReasoner achieves strong generalization and reasoning performance, outperforming open-source baselines and approaching proprietary models like GPT-4o. Our work highlights the importance of symbolic representation and structured reasoning for advancing chart-based visual understanding.By tightly coupling visual parsing with logical reasoning, ChartReasoner offers a unified paradigm for complex analytical tasks. Future work may explore extending this framework to broader domains such as scientific visualization.

Limitations

Our study is comprehensive but has certain limitations that we aim to address in future research.First, due to computational constraints, we conduct all experiments using a 7B-parameter model. Although this setting yields promising results, scaling to larger models may further enhance performance and generalization capabilities. Second, the current evaluation focuses primarily on benchmark-style synthetic and semi-structured charts. The generalization of our method to more complex, real-world visualizations remains an open challenge.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, and 1 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Saleem Ahmed, Bhavin Jawade, Shubham Pandey, Srirangaraj Setlur, and Venu Govindaraju. 2023. Realcqa: Scientific chart question answering as a test-bed for first-order logic. In *Proceedings of the ICDAR*.

Anthropic. 2024. Introducing the next generation of claude. Accessed: 2025-05-18.

Gilles Baechler, Srinivas Sunkara, Maria Wang, Fedir Zubach, Hassan Mansoor, Vincent Etter, Victor Cărbune, Jason Lin, Jindong Chen, and Abhanshu Sharma. 2024. Screenai: A vision-language model for ui and infographics understanding. In *Proceedings of the IJCAI*.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Victor Carbune, Hassan Mansoor, Fangyu Liu, Rahul Aralikatte, Gilles Baechler, Jindong Chen, and Abhanshu Sharma. 2024. Chart-based reasoning: Transferring capabilities from llms to vlms. In *Findings of the ACL*.

Ritwick Chaudhry, Sumit Shekhar, Utkarsh Gupta, Pranav Maneriker, Prann Bansal, and Ajay Joshi. 2020. Leaf-qa: Locate, encode & attend for figure question answering. In *Proceedings of the WACV*.

Liang Chen, Lei Li, Haozhe Zhao, Yifan Song, and Vinci. 2025. R1-v: Reinforcing super generalization ability in vision-language models with less than $3. https://github.com/Deep-Agent/R1-V. Accessed: 2025-02-02.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, and 1 others. 2024a. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*.

Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, and 1 others. 2024b. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *Science China Information Sciences*.

Zhi-Qi Cheng, Qi Dai, and Alexander G Hauptmann. 2023. Chartreader: A unified framework for chart derendering and comprehension without heuristic rules. In *Proceedings of the ICCV*.

Huilin Deng, Ding Zou, Rui Ma, Hongchen Luo, Yang Cao, and Yu Kang. 2025. Boosting the generalization and reasoning of vision language models with curriculum reinforcement learning. *arXiv preprint arXiv:2503.07065*.

Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Bin Wang, Linke Ouyang, Xilin Wei, Songyang Zhang, Haodong Duan, Maosong Cao, and 1 others. 2024. Internlm-xcomposer2: Mastering free-form text-image composition and comprehension in vision-language large model. *arXiv preprint arXiv:2401.16420*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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

Wenyi Hong, Weihan Wang, Ming Ding, Wenmeng Yu, Qingsong Lv, Yan Wang, Yean Cheng, Shiyu Huang, Junhui Ji, Zhao Xue, and 1 others. 2024. Cogvlm2: Visual language models for image and video understanding. *arXiv preprint arXiv:2408.16500*.

Muye Huang, Han Lai, Xinyu Zhang, Wenjun Wu, Jie Ma, Lingling Zhang, and Jun Liu. 2025a. Evochart: A benchmark and a self-training approach towards real-world chart understanding. In *Proceedings of the AAAI*.

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and Shaohui Lin. 2025b. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*.

Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. 2018. Dvqa: Understanding data visualizations via question answering. In *Proceedings of the CVPR*.

Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Ákos Kádár, Adam Trischler, and Yoshua Bengio. 2017. Figureqa: An annotated figure dataset for visual reasoning. *arXiv preprint arXiv:1710.07300*.

Bingxuan Li, Yiwei Wang, Jiuxiang Gu, Kai-Wei Chang, and Nanyun Peng. 2025. Metal: A multi-agent framework for chart generation with test-time scaling. *arXiv preprint arXiv:2502.17651*.

Fangyu Liu, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Yasemin Altun, Nigel Collier, and Julian Martin Eisenschlos. 2023. Matcha: Enhancing visual language pretraining with math reasoning and chart derendering. In *Proceedings of the ACL*.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024. Improved baselines with visual instruction tuning. In *Proceedings of the CVPR*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Findings of the ICLR*.

Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Proceedings of the ACL*.

Ahmed Masry, Mohammed Saidul Islam, Mahir Ahmed, Aayush Bajaj, Firoz Kabir, Aaryaman Kartha, Md Tahmid Rahman Laskar, Mizanur Rahman, Shadikur Rahman, Mehrad Shahmohammadi, and 1 others. 2025a. Chartqapro: A more diverse and challenging benchmark for chart question answering. *arXiv preprint arXiv:2504.05506*.

Ahmed Masry, Parsa Kavehzadeh, Xuan Long Do, Enamul Hoque, and Shafiq Joty. 2023. Unichart: A universal vision-language pretrained model for chart comprehension and reasoning. In *Proceedings of the EMNLP*.

Ahmed Masry, Mehrad Shahmohammadi, Md Rizwan Parvez, Enamul Hoque, and Shafiq Joty. 2024. Chartinstruct: Instruction tuning for chart comprehension and reasoning. In *Findings of the ACL*.

Ahmed Masry, Megh Thakkar, Aayush Bajaj, Aaryaman Kartha, Enamul Hoque, and Shafiq Joty. 2025b. Chartgemma: Visual instruction-tuning for chart reasoning in the wild. In *Findings of the ACL*.

Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Tiancheng Han, Botian Shi, Wenhai Wang, Junjun He, and 1 others. 2025. Mm-eureka: Exploring the frontiers of multimodal reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2503.07365*.

Fanqing Meng, Wenqi Shao, Quanfeng Lu, Peng Gao, Kaipeng Zhang, Yu Qiao, and Ping Luo. 2024. Chartassisstant: A universal chart multimodal language model via chart-to-table pre-training and multitask instruction tuning. In *Findings of the ACL*.

Nitesh Methani, Pritha Ganguly, Mitesh M Khapra, and Pratyush Kumar. 2020. Plotqa: Reasoning over scientific plots. In *Proceedings of the WACV*.

OpenAI. 2024. Introducing openai o1. `https://openai.com/o1/`. Accessed: 2025-5-18.

OpenAI. 2025. Openai o3 and o4-mini system card. Accessed: 2025-05-19.

Yingzhe Peng, Gongrui Zhang, Miaosen Zhang, Zhiyuan You, Jie Liu, Qipeng Zhu, Kai Yang, Xingzhong Xu, Xin Geng, and Xu Yang. 2025. Lmm-r1: Empowering 3b lmms with strong reasoning abilities through two-stage rule-based rl. *arXiv preprint arXiv:2503.07536*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, and 1 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, and 1 others. 2024. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Chengyue Wu, Yixiao Ge, Qiushan Guo, Jiahao Wang, Zhixuan Liang, Zeyu Lu, Ying Shan, and Ping Luo. 2025. Plot2code: A comprehensive benchmark for evaluating multi-modal large language models in code generation from scientific plots. In *Findings of the ACL*.

Renqiu Xia, Bo Zhang, Hancheng Ye, Xiangchao Yan, Qi Liu, Hongbin Zhou, Zijun Chen, Peng Ye, Min Dou, Botian Shi, and 1 others. 2024. Chartx & chartvlm: A versatile benchmark and foundation model for complicated chart reasoning. *arXiv preprint arXiv:2402.12185*.

Zhengzhuo Xu, Sinan Du, Yiyan Qi, Chengjin Xu, Chun Yuan, and Jian Guo. 2023. Chartbench: A benchmark for complex visual reasoning in charts. *arXiv preprint arXiv:2312.15915*.

Cheng Yang, Chufan Shi, Yaxin Liu, Bo Shui, Junjie Wang, Mohan Jing, Linran Xu, Xinyu Zhu, Siheng Li, Yuxiang Zhang, and 1 others. 2025a. Chartmimic: Evaluating lmm's cross-modal reasoning capability via chart-to-code generation. In *Proceedings of the ICLR*.

Yi Yang, Xiaoxuan He, Hongkun Pan, Xiyan Jiang, Yan Deng, Xingtao Yang, Haoyu Lu, Dacheng Yin, Fengyun Rao, Minfeng Zhu, and 1 others. 2025b. R1-onevision: Advancing generalized multimodal reasoning through cross-modal formalization. *arXiv preprint arXiv:2503.10615*.

Liang Zhang, Anwen Hu, Haiyang Xu, Ming Yan, Yichen Xu, Qin Jin, Ji Zhang, and Fei Huang. 2024. Tinychart: Efficient chart understanding with visual token merging and program-of-thoughts learning. In *Proceedings of the ACL*.

Zhihan Zhang, Yixin Cao, and Lizi Liao. 2025. Enhancing chart-to-code generation in multimodal large language models via iterative dual preference learning. *arXiv preprint arXiv:2504.02906*.

Xuanle Zhao, Xianzhen Luo, Qi Shi, Chi Chen, Shuo Wang, Wanxiang Che, Zhiyuan Liu, and Maosong Sun. 2025. Chartcoder: Advancing multimodal large language model for chart-to-code generation. *arXiv preprint arXiv:2501.06598*.

Hengguang Zhou, Xirui Li, Ruochen Wang, Minhao Cheng, Tianyi Zhou, and Cho-Jui Hsieh. 2025. R1-zero's" aha moment" in visual reasoning on a 2b non-sft model. *arXiv preprint arXiv:2503.05132*.

A Implementation Details.

We use Qwen2.5-VL-7B (Bai et al., 2025) as the backbone and perform supervised fine-tuning on 8 A100 80GB GPUs. The vision tower and projection layers are frozen, while the language model is fully trainable. Training runs for 4 epochs with an effective batch size of 8, using BF16 precision. We apply the AdamW (Loshchilov and Hutter, 2019) optimizer with learning rate 1e-5. The maximum sequence length is 4096 tokens, and images are resized to 512×512 pixels. We further apply GRPO for 2 epochs starting from the SFT checkpoint. The model generates 8 completions per input, with reward-weighted selection based on accuracy, format correctness, and length suitability.

B Qualitative Analysis

To further illustrate the performance improvements brought by our model in chart-based multimodal reasoning, we conduct a qualitative analysis using representative examples. These cases help demonstrate how enhanced reasoning capabilities can effectively assist visual understanding, especially when direct visual recognition is ambiguous or when the question requires complex logical interpretation. As illustrated in Figures 6–9, these examples further demonstrate the effectiveness of our method.

Visual-Aided Reasoning. One core strength of our ChartReasoner lies in its ability to perform visual reasoning that supplements and corrects potentially uncertain visual recognition. As shown in Figure 6, the example question is: "What is the label of the highest bar of February?" This task requires the model to first locate February on the x-axis and then identify the label corresponding to its highest bar—thus constituting a visual reasoning problem.

While baseline models such as Qwen2.5VL fail to correctly locate "February" and incorrectly identify "Sales" as the highest category, ChartReasoner demonstrates a more accurate analysis by first reasoning through the axis structure: "The x-axis data is [January, February, March, ..., December], so February is the second month." This allows it to correctly localize the February column and extract the corresponding bar label, thereby arriving at the correct answer.

This example highlights that reasoning capabilities can effectively compensate for limitations in visual recognition, particularly when axis elements or data labels are densely packed, occluded, or ambiguously rendered.

Complex Semantic Reasoning. In addition to visual grounding, ChartReasoner also excels in handling complex semantic questions that require precise logical understanding. As shown in Figure 7, the example question is: "How many percent of U.S. coffee drinkers drink less than 2 cups of coffee at home on a weekday?" The key to this question lies in correctly interpreting the condition "less than 2 cups." However, Qwen2.5VL incorrectly includes the "2 cups" category in its calculation, leading to a wrong answer. In contrast, ChartReasoner demonstrates its advanced reasoning by recognizing the logical boundary of the query and explicitly excluding the 2-cups group from its aggregation, yielding the correct answer. This indicates that reasoning ability is critical for precise comprehension of quantitative and conditional logic, which is often required in real-world ChartQA scenarios.

C Prompt Design for Visual Evaluation with GPT-4V

To comprehensively assess the visual quality of generated charts, we adopt a structured prompt-based evaluation approach using GPT-4V. The prompt instructs the model to compare a generated chart with its corresponding ground-truth version and assign a similarity score ranging from 1 to 10. The scoring is based on four key criteria: Colors (accuracy of color schemes), Axes & Scale (consistency of axis ranges and units), Data Points Position (placement and alignment of bars, lines, or markers), and Overall Layout (correctness of titles, labels, legends, etc.).

This prompt enables GPT-4V to produce fine-grained visual judgments that go beyond traditional execution-based metrics (e.g., code correctness), capturing layout-level discrepancies that impact real-world interpretability. An example of such a prompt is illustrated in Figure 10. This evaluation method bridges the gap between syntactic correctness and perceptual fidelity in chart generation tasks.

D Prompt Engineering for ECharts Code Generation

To enable effective chart generation, we employ domain-specific prompt engineering tailored to the ECharts visualization framework. The prompts are constructed to cover 18 thematic domains and 111 subtopics, spanning social, economic, technological, and environmental dimensions. This ensures diverse coverage of chart types and semantic contexts.

Each prompt clearly specifies the chart topic, the intended visual form (e.g., bar chart, line chart, scatter plot), and any constraints on layout or data encoding. As demonstrated in Figure 11, this guided prompting allows models like DeepSeek R1 to leverage their strong reasoning abilities to produce structurally varied and semantically rich visualizations. These prompts are essential to ensure that the generated charts are not only syntactically valid but also meaningful and domain-relevant.

E Chart-to-Code Dataset Detailed Case

To further illustrate the design of our Chart-to-Code Dataset, we present selected examples that directly show the generated ECharts HTML code alongside the corresponding rendered chart. These examples also highlight the flexibility of the chart template system and the reasoning capability of the DeepSeek R1 model in generating structurally complex and thematically rich charts. By showcasing a range of chart types—including bar, line, and pie charts—these cases reflect the robustness of our prompt engineering approach and the effectiveness of the multi-stage quality filtering pipeline described in the methodology. Figures 12–15 present more detailed examples from the Chart2Code dataset.

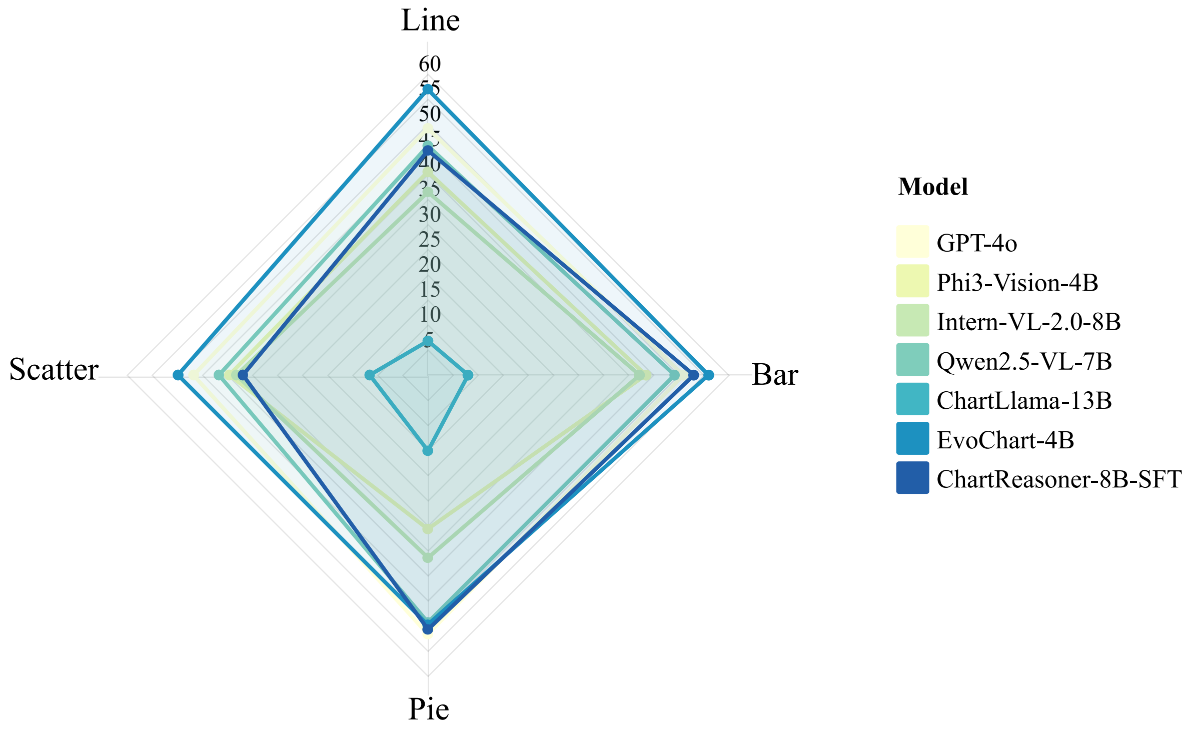
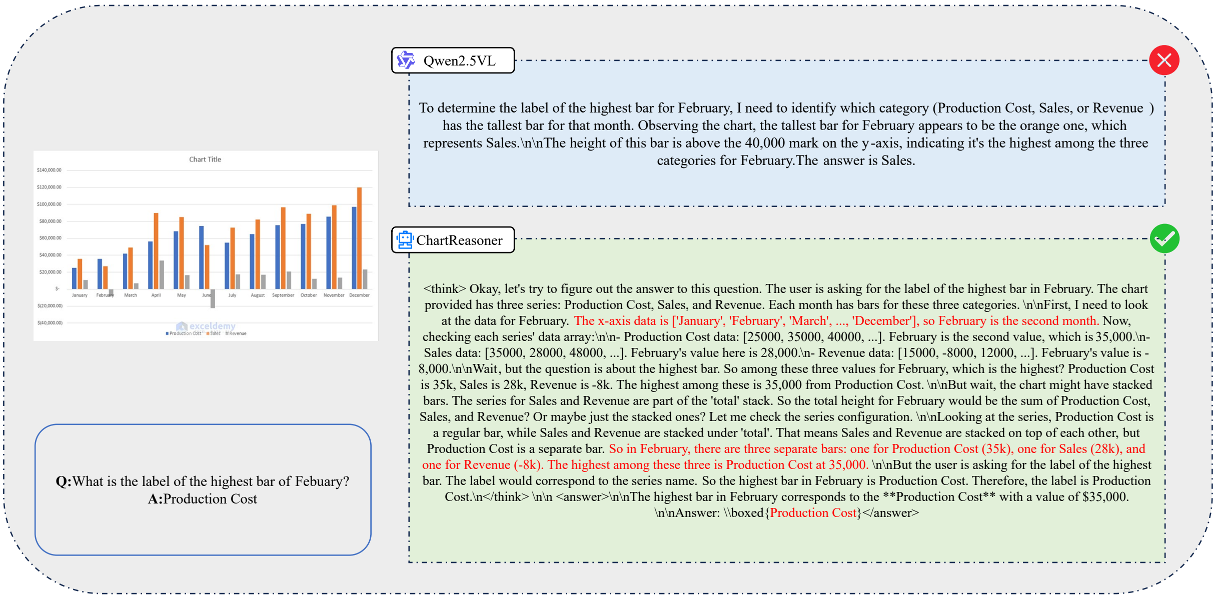


Figure 5: EvoChart Performance Across Chart Types.

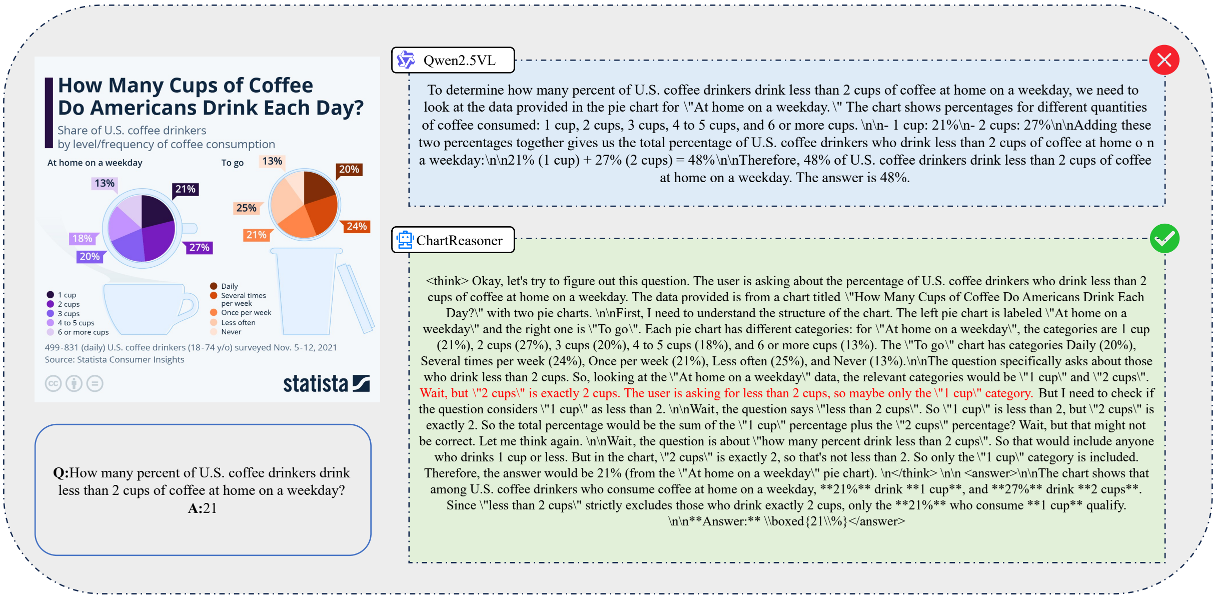


Q:What is the label of the highest bar of Febuary?
A:Production Cost

To determine the label of the highest bar for February, I need to identify which category (Production Cost, Sales, or Revenue) has the tallest bar for that month. Observing the chart, the tallest bar for February appears to be the orange one, which represents Sales.\n\nThe height of this bar is above the 40,000 mark on the y-axis, indicating it's the highest among the three categories for February.The answer is Sales.

<think> Okay, let's try to figure out the answer to this question. The user is asking for the label of the highest bar in February. The chart provided has three series: Production Cost, Sales, and Revenue. Each month has bars for these three categories. \n\nFirst, I need to look at the data for February. The x-axis data is ['January', 'February', 'March', ..., 'December'], so February is the second month. Now, checking each series' data array:\n\n- Production Cost data: [25000, 35000, 40000, ...]. February is the second value, which is 35,000.\n- Sales data: [35000, 28000, 48000, ...]. February's value here is 28,000.\n- Revenue data: [15000, -8000, 12000, ...]. February's value is -8,000.\n\nWait, but the question is about the highest bar. So among these three values for February, which is the highest? Production Cost is 35k, Sales is 28k, Revenue is -8k. The highest among these is 35,000 from Production Cost. \n\nBut wait, the chart might have stacked bars. The series for Sales and Revenue are part of the 'total' stack. So the total height for February would be the sum of Production Cost, Sales, and Revenue? Or maybe just the stacked ones? Let me check the series configuration. \n\nLooking at the series, Production Cost is a regular bar, while Sales and Revenue are stacked under 'total'. That means Sales and Revenue are stacked on top of each other, but Production Cost is a separate bar. So in February, there are three separate bars: one for Production Cost (35k), one for Sales (28k), and one for Revenue (-8k). The highest among these three is Production Cost at 35,000. \n\nBut the user is asking for the label of the highest bar. The label would correspond to the series name. So the highest bar in February is Production Cost. Therefore, the label is Production Cost.\n</think> \n\n <answer>\n\nThe highest bar in February corresponds to the **Production Cost** with a value of $35,000. \n\nAnswer: \\boxed{Production Cost}</answer>

Figure 6: Comparison of Model Responses in ChartQA (Example 1).



Q:How many percent of U.S. coffee drinkers drink less than 2 cups of coffee at home on a weekday?
A:21

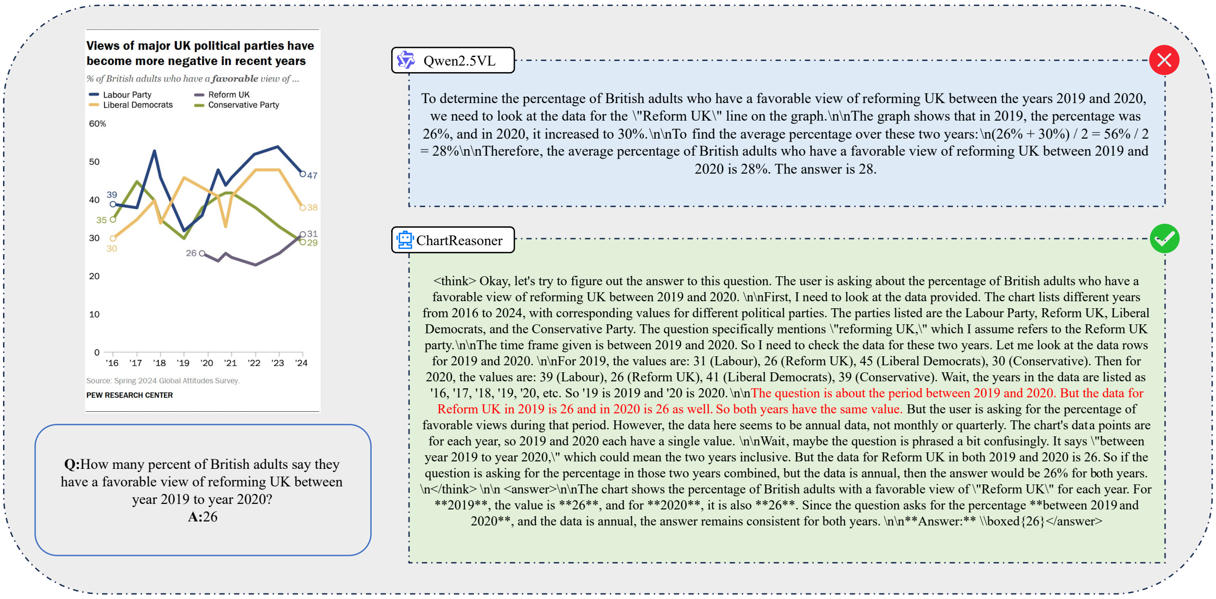
Qwen2.5VL

To determine how many percent of U.S. coffee drinkers drink less than 2 cups of coffee at home on a weekday, we need to look at the data provided in the pie chart for \"At home on a weekday. \" The chart shows percentages for different quantities of coffee consumed: 1 cup, 2 cups, 3 cups, 4 to 5 cups, and 6 or more cups. \n\n- 1 cup: 21%\n- 2 cups: 27%\n\nAdding these two percentages together gives us the total percentage of U.S. coffee drinkers who drink less than 2 cups of coffee at home o n a weekday:\n\n21% (1 cup) + 27% (2 cups) = 48%\n\nTherefore, 48% of U.S. coffee drinkers drink less than 2 cups of coffee at home on a weekday. The answer is 48%.

ChartReasoner

<think> Okay, let's try to figure out this question. The user is asking about the percentage of U.S. coffee drinkers who drink less than 2 cups of coffee at home on a weekday. The data provided is from a chart titled \"How Many Cups of Coffee Do Americans Drink Each Day?\" with two pie charts. \n\nFirst, I need to understand the structure of the chart. The left pie chart is labeled \"At home on a weekday\" and the right one is \"To go\". Each pie chart has different categories: for \"At home on a weekday\", the categories are 1 cup (21%), 2 cups (27%), 3 cups (20%), 4 to 5 cups (18%), and 6 or more cups (13%). The \"To go\" chart has categories Daily (20%), Several times per week (24%), Once per week (21%), Less often (25%), and Never (13%).\n\nThe question specifically asks about those who drink less than 2 cups. So, looking at the \"At home on a weekday\" data, the relevant categories would be \"1 cup\" and \"2 cups\". Wait, but \"2 cups\" is exactly 2 cups. The user is asking for less than 2 cups, so maybe only the \"1 cup\" category. But I need to check if the question considers \"1 cup\" as less than 2. \n\nWait, the question says \"less than 2 cups\". So \"1 cup\" is less than 2, but \"2 cups\" is exactly 2. So the total percentage would be the sum of the \"1 cup\" percentage plus the \"2 cups\" percentage? Wait, but that might not be correct. Let me think again. \n\nWait, the question is about \"how many percent drink less than 2 cups\". So that would include anyone who drinks 1 cup or less. But in the chart, \"2 cups\" is exactly 2, so that's not less than 2. So only the \"1 cup\" category is included. Therefore, the answer would be 21% (from the \"At home on a weekday\" pie chart). \n</think> \n\n <answer>\n\nThe chart shows that among U.S. coffee drinkers who consume coffee at home on a weekday, **21%** drink **1 cup**, and **27%** drink **2 cups**. Since \"less than 2 cups\" strictly excludes those who drink exactly 2 cups, only the **21%** who consume **1 cup** qualify. \n\n**Answer:** \\boxed{21\\%}</answer>

Figure 7: Comparison of Model Responses in ChartQA (Example 2).



Q:How many percent of British adults say they have a favorable view of reforming UK between year 2019 to year 2020?
A:26

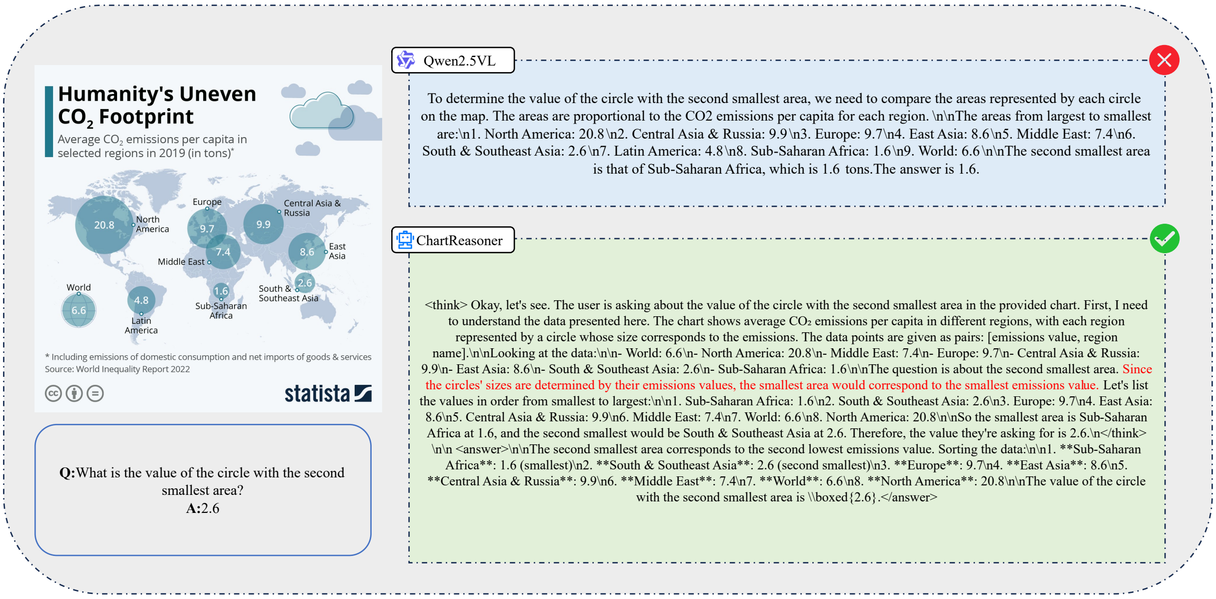
Qwen2.5VL

To determine the percentage of British adults who have a favorable view of reforming UK between the years 2019 and 2020, we need to look at the data for the \"Reform UK\" line on the graph.\n\nThe graph shows that in 2019, the percentage was 26%, and in 2020, it increased to 30%.\n\nTo find the average percentage over these two years:\n(26% + 30%) / 2 = 56% / 2 = 28%\n\nTherefore, the average percentage of British adults who have a favorable view of reforming UK between 2019 and 2020 is 28%. The answer is 28.

ChartReasoner

<think> Okay, let's try to figure out the answer to this question. The user is asking about the percentage of British adults who have a favorable view of reforming UK between 2019 and 2020. \n\nFirst, I need to look at the data provided. The chart lists different years from 2016 to 2024, with corresponding values for different political parties. The parties listed are the Labour Party, Reform UK, Liberal Democrats, and the Conservative Party. The question specifically mentions \"reforming UK,\" which I assume refers to the Reform UK party.\n\nThe time frame given is between 2019 and 2020. So I need to check the data for these two years. Let me look at the data rows for 2019 and 2020. \n\nFor 2019, the values are: 31 (Labour), 26 (Reform UK), 45 (Liberal Democrats), 30 (Conservative). Then for 2020, the values are: 39 (Labour), 26 (Reform UK), 41 (Liberal Democrats), 39 (Conservative). Wait, the years in the data are listed as '16, '17, '18, '19, '20, etc. So '19 is 2019 and '20 is 2020. \n\nThe question is about the period between 2019 and 2020. But the data for Reform UK in 2019 is 26 and in 2020 is 26 as well. So both years have the same value. But the user is asking for the percentage of favorable views during that period. However, the data here seems to be annual data, not monthly or quarterly. The chart's data points are for each year, so 2019 and 2020 each have a single value. \n\nWait, maybe the question is phrased a bit confusingly. It says \"between year 2019 to year 2020,\" which could mean the two years inclusive. But the data for Reform UK in both 2019 and 2020 is 26. So if the question is asking for the percentage in those two years combined, but the data is annual, then the answer would be 26% for both years. \n</think> \n\n <answer>\n\nThe chart shows the percentage of British adults with a favorable view of \"Reform UK\" for each year. For **2019**, the value is **26**, and for **2020**, it is also **26**. Since the question asks for the percentage **between 2019 and 2020**, and the data is annual, the answer remains consistent for both years. \n\n**Answer:** \\boxed{26}</answer>

Figure 8: Comparison of Model Responses in ChartQA (Example 3).


Humanity's Uneven CO$_2$ Footprint
Average CO$_2$ emissions per capita in selected regions in 2019 (in tons)*
20.8
North America
Europe
9.7
9.9
Central Asia & Russia
7.4
Middle East
8.6
East Asia
World
6.6
4.8
Latin America
1.6
Sub-Saharan Africa
2.6
South & Southeast Asia
* Including emissions of domestic consumption and net imports of goods & services
Source: World Inequality Report 2022
statista
Qwen2.5VL
ChartReasoner

Q:What is the value of the circle with the second smallest area?
A:2.6

Qwen2.5VL

To determine the value of the circle with the second smallest area, we need to compare the areas represented by each circle on the map. The areas are proportional to the CO2 emissions per capita for each region. \n\nThe areas from largest to smallest are:\n1. North America: 20.8\n2. Central Asia & Russia: 9.9\n3. Europe: 9.7\n4. East Asia: 8.6\n5. Middle East: 7.4\n6. South & Southeast Asia: 2.6\n7. Latin America: 4.8\n8. Sub-Saharan Africa: 1.6\n9. World: 6.6\n\nThe second smallest area is that of Sub-Saharan Africa, which is 1.6 tons.The answer is 1.6.

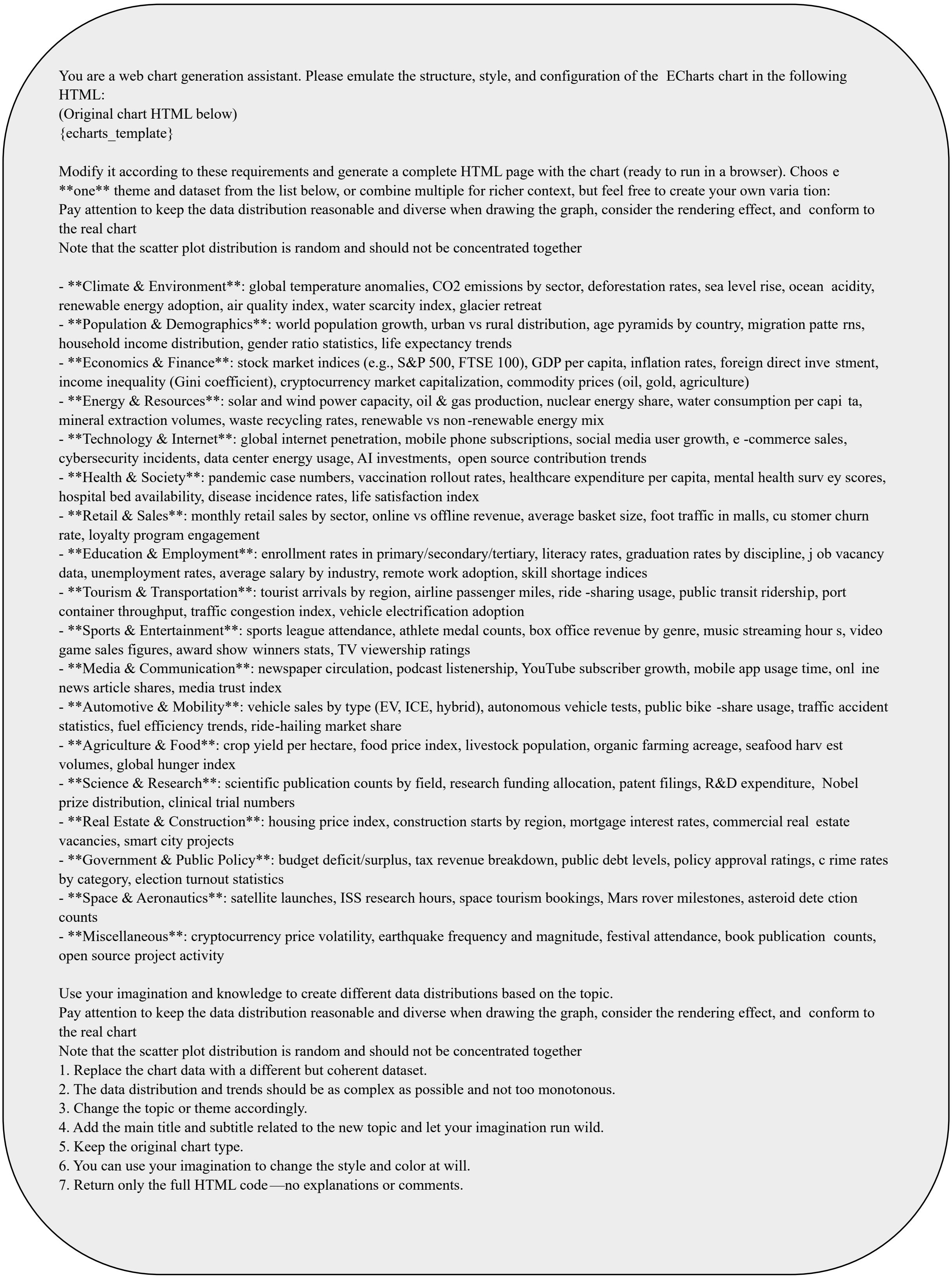
ChartReasoner

<think> Okay, let's see. The user is asking about the value of the circle with the second smallest area in the provided chart. First, I need to understand the data presented here. The chart shows average CO$_2$ emissions per capita in different regions, with each region represented by a circle whose size corresponds to the emissions. The data points are given as pairs: [emissions value, region name].\n\nLooking at the data:\n\n- World: 6.6\n- North America: 20.8\n- Middle East: 7.4\n- Europe: 9.7\n- Central Asia & Russia: 9.9\n- East Asia: 8.6\n- South & Southeast Asia: 2.6\n- Sub-Saharan Africa: 1.6\n\nThe question is about the second smallest area. Since the circles' sizes are determined by their emissions values, the smallest area would correspond to the smallest emissions value. Let's list the values in order from smallest to largest:\n\n1. Sub-Saharan Africa: 1.6\n2. South & Southeast Asia: 2.6\n3. Europe: 9.7\n4. East Asia: 8.6\n5. Central Asia & Russia: 9.9\n6. Middle East: 7.4\n7. World: 6.6\n8. North America: 20.8\n\nSo the smallest area is Sub-Saharan Africa at 1.6, and the second smallest would be South & Southeast Asia at 2.6. Therefore, the value they're asking for is 2.6.\n</think> \n\n <answer>\n\nThe second smallest area corresponds to the second lowest emissions value. Sorting the data:\n\n1. **Sub-Saharan Africa**: 1.6 (smallest)\n2. **South & Southeast Asia**: 2.6 (second smallest)\n3. **Europe**: 9.7\n4. **East Asia**: 8.6\n5. **Central Asia & Russia**: 9.9\n6. **Middle East**: 7.4\n7. **World**: 6.6\n8. **North America**: 20.8\n\nThe value of the circle with the second smallest area is \\boxed{2.6}.</answer>

Figure 9: Comparison of Model Responses in ChartQA (Example 4).

Please evaluate the similarity between a reference image created using matplotlib and an image generated by code provided by an AI assistant. Consider factors such as the overall appearance, colors, shapes, positions, and other visual elements of the im ages. Begin your evaluation by providing a short explanation. Be as objective as possible.
After providing your explanation, you must rate the response on a scale of 1 to 10 by strictly following this format: Rating: [[5]]

Figure 10: GPT-4V Visual Evaluation Prompt.



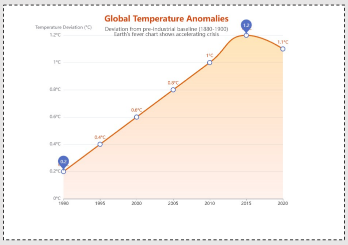
You are a web chart generation assistant. Please emulate the structure, style, and configuration of the ECharts chart in the following HTML:
(Original chart HTML below)
{echarts_template}

Modify it according to these requirements and generate a complete HTML page with the chart (ready to run in a browser). Choos e **one** theme and dataset from the list below, or combine multiple for richer context, but feel free to create your own varia tion:
Pay attention to keep the data distribution reasonable and diverse when drawing the graph, consider the rendering effect, and conform to the real chart
Note that the scatter plot distribution is random and should not be concentrated together

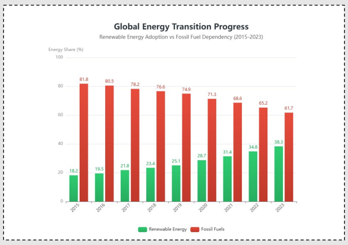
- **Climate & Environment**: global temperature anomalies, CO2 emissions by sector, deforestation rates, sea level rise, ocean acidity, renewable energy adoption, air quality index, water scarcity index, glacier retreat
- **Population & Demographics**: world population growth, urban vs rural distribution, age pyramids by country, migration patte rns, household income distribution, gender ratio statistics, life expectancy trends
- **Economics & Finance**: stock market indices (e.g., S&P 500, FTSE 100), GDP per capita, inflation rates, foreign direct inve stment, income inequality (Gini coefficient), cryptocurrency market capitalization, commodity prices (oil, gold, agriculture)
- **Energy & Resources**: solar and wind power capacity, oil & gas production, nuclear energy share, water consumption per capi ta, mineral extraction volumes, waste recycling rates, renewable vs non -renewable energy mix
- **Technology & Internet**: global internet penetration, mobile phone subscriptions, social media user growth, e -commerce sales, cybersecurity incidents, data center energy usage, AI investments, open source contribution trends
- **Health & Society**: pandemic case numbers, vaccination rollout rates, healthcare expenditure per capita, mental health surv ey scores, hospital bed availability, disease incidence rates, life satisfaction index
- **Retail & Sales**: monthly retail sales by sector, online vs offline revenue, average basket size, foot traffic in malls, cu stomer churn rate, loyalty program engagement
- **Education & Employment**: enrollment rates in primary/secondary/tertiary, literacy rates, graduation rates by discipline, j ob vacancy data, unemployment rates, average salary by industry, remote work adoption, skill shortage indices
- **Tourism & Transportation**: tourist arrivals by region, airline passenger miles, ride -sharing usage, public transit ridership, port container throughput, traffic congestion index, vehicle electrification adoption
- **Sports & Entertainment**: sports league attendance, athlete medal counts, box office revenue by genre, music streaming hour s, video game sales figures, award show winners stats, TV viewership ratings
- **Media & Communication**: newspaper circulation, podcast listenership, YouTube subscriber growth, mobile app usage time, onl ine news article shares, media trust index
- **Automotive & Mobility**: vehicle sales by type (EV, ICE, hybrid), autonomous vehicle tests, public bike -share usage, traffic accident statistics, fuel efficiency trends, ride-hailing market share
- **Agriculture & Food**: crop yield per hectare, food price index, livestock population, organic farming acreage, seafood harv est volumes, global hunger index
- **Science & Research**: scientific publication counts by field, research funding allocation, patent filings, R&D expenditure, Nobel prize distribution, clinical trial numbers
- **Real Estate & Construction**: housing price index, construction starts by region, mortgage interest rates, commercial real estate vacancies, smart city projects
- **Government & Public Policy**: budget deficit/surplus, tax revenue breakdown, public debt levels, policy approval ratings, c rime rates by category, election turnout statistics
- **Space & Aeronautics**: satellite launches, ISS research hours, space tourism bookings, Mars rover milestones, asteroid dete ction counts
- **Miscellaneous**: cryptocurrency price volatility, earthquake frequency and magnitude, festival attendance, book publication counts, open source project activity

Use your imagination and knowledge to create different data distributions based on the topic.
Pay attention to keep the data distribution reasonable and diverse when drawing the graph, consider the rendering effect, and conform to the real chart
Note that the scatter plot distribution is random and should not be concentrated together
1. Replace the chart data with a different but coherent dataset.
2. The data distribution and trends should be as complex as possible and not too monotonous.
3. Change the topic or theme accordingly.
4. Add the main title and subtitle related to the new topic and let your imagination run wild.
5. Keep the original chart type.
6. You can use your imagination to change the style and color at will.
7. Return only the full HTML code—no explanations or comments.

Figure 11: ECharts Code Generation Prompt.

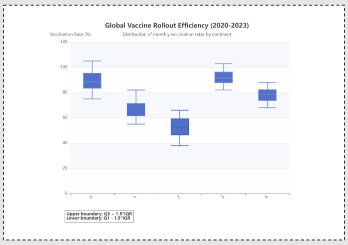


```
<html>
<head>
    <meta charset="utf-8" />
    <title>Climate Change Analysis</title>
    <script src="https://cdn.jsdelivr.net/npm/echarts@5.6.0/dist/echarts.min.js"></script>
    <style>
        html, body { width: 100%; height: 100%; margin: 0; padding: 0; }
        #main { width: 100%; height: 100%; }
    </style>
</head>
<body>
    <div id="main"></div>
    <script>
        var chartDom = document.getElementById('main');
        var myChart = echarts.init(chartDom);
        var option;

        option = {
            title: {
                text: 'Global Temperature Anomalies',
                subtext: 'Deviation from pre-industrial baseline (1880-1900)\nEarth\'s fever chart shows
accelerating crisis',
                left: 'center',
                textStyle: { color: '#d74b14', fontSize: 22 },
                subtextStyle: { color: '#666', fontSize: 14 }
            },
            xAxis: {
                type: 'category',
                boundaryGap: false,
                data: ['1990', '1995', '2000', '2005', '2010', '2015', '2020'],
                axisLabel: { color: '#333' }
            },
            yAxis: {
                type: 'value',
                name: 'Temperature Deviation (°C)',
                min: 0,
                axisLabel: { formatter: '{value}°C' }
            },
            series: [{
                data: [0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.1],
                type: 'line',
                smooth: true,
                symbolSize: 12,
                lineStyle: { color: '#e36209', width: 3 },
                areaStyle: {
                    color: new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                        { offset: 0, color: 'rgba(255,165,0,0.4)' },
                        { offset: 1, color: 'rgba(255,69,0,0.1)' }
                    ])
                },
                label: {
                    show: true,
                    formatter: '{c}°C',
                    color: '#d74b14',
                    fontSize: 12,
                    position: 'top'
                },
                markPoint: {
                    data: [
                        { type: 'max', name: 'Record High' },
                        { type: 'min', name: 'Coolest Year' }
                    ]
                }
            }],
            tooltip: { trigger: 'axis' },
            grid: { containLabel: true, left: '10%' }
        };

        option && myChart.setOption(option);
        window.addEventListener('resize', function() { myChart.resize(); });
    </script>
</body>
</html>
```

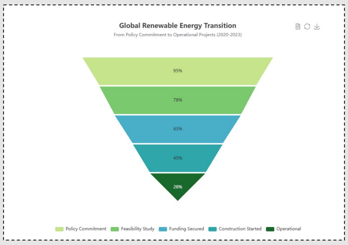


```
<html>
<head>
    <meta charset="utf-8" />
    <title>Energy Transition Dashboard</title>
    <script src="https://cdn.jsdelivr.net/npm/echarts@5.6.0/dist/echarts.min.js"></script>
    <style>
        html, body { width: 100%; height: 100%; margin: 0; padding: 0; }
        #main { width: 100%; height: 100%; }
    </style>
</head>
<body>
    <div id="main"></div>
    <script>
        var chartDom = document.getElementById('main');
        var myChart = echarts.init(chartDom);

        var option = {
            title: {
                text: 'Global Energy Transition Progress',
                subtext: 'Renewable Energy Adoption vs Fossil Fuel Dependency (2015-2023)',
                left: 'center',
                textStyle: { color: '#2c343c', fontSize: 22 },
                subtextStyle: { color: '#666', fontSize: 14 }
            },
            tooltip: { trigger: 'axis' },
            xAxis: {
                type: 'category',
                data: ['2015','2016','2017','2018','2019','2020','2021','2022','2023'],
                axisLabel: { rotate: 45, color: '#555' }
            },
            yAxis: {
                type: 'value',
                name: 'Energy Share (%)',
                axisLine: { lineStyle: { color: '#999' } }
            },
            legend: {
                data: ['Renewable Energy', 'Fossil Fuels'],
                bottom: 10
            },
            series: [{
                name: 'Renewable Energy',
                type: 'bar',
                data: [18.2, 19.5, 21.8, 23.4, 25.1, 28.7, 31.4, 34.8, 38.3],
                itemStyle: {
                    color: new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                        { offset: 0, color: '#2ecc71' },
                        { offset: 1, color: '#27ae60' }
                    ])
                },
                label: { show: true, position: 'top', color: '#27ae60' }
            },{
                name: 'Fossil Fuels',
                type: 'bar',
                data: [81.8, 80.5, 78.2, 76.6, 74.9, 71.3, 68.6, 65.2, 61.7],
                itemStyle: {
                    color: new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                        { offset: 0, color: '#e74c3c' },
                        { offset: 1, color: '#c0392b' }
                    ])
                },
                label: { show: true, position: 'top', color: '#c0392b' }
            }],
            grid: { top: 100, bottom: 100 }
        };

        myChart.setOption(option);
        window.addEventListener('resize', function() { myChart.resize(); });
    </script>
</body>
</html>
```

Figure 12: Example 1 from the Chart-to-Code Dataset.

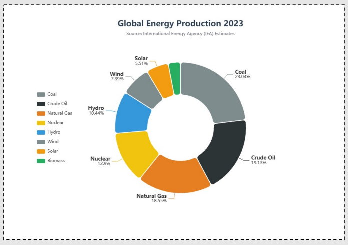


```
<html>
<head>
    <meta charset="utf-8" />
    <title>ECharts</title>
    <script src="https://cdn.jsdelivr.net/npm/echarts@5.6.0/dist/echarts.min.js"></script>
    <style>
        html, body { width: 100%; height: 100%; margin: 0; padding: 0; }
        #main { width: 100%; height: 100%; }
    </style>
</head>
<body>
    <div id="main"></div>
    <script>
        var chartDom = document.getElementById('main');
        var myChart = echarts.init(chartDom);
        var option;

        option = {
            title: [
                { text: 'Global Vaccine Rollout Efficiency (2020-2023)', subtext: 'Distribution of monthly
vaccination rates by continent', left: 'center' },
                { text: 'Upper boundary: Q3 + 1.5*IQR\nLower boundary: Q1 - 1.5*IQR', borderColor: '#666',
borderWidth: 1, textStyle: { fontSize: 12 }, left: '10%', top: '90%' }
            ],
            dataset: [
                {
                    source: [
                        [82, 75, 88, 92, 85, 78, 95, 98, 91, 87, 89, 93, 84, 79, 96, 102, 105, 98, 88, 81],
                        [65, 58, 72, 68, 63, 70, 75, 82, 79, 66, 61, 57, 64, 71, 69, 74, 63, 59, 55, 62],
                        [45, 38, 52, 49, 57, 61, 55, 63, 66, 58, 42, 39, 48, 53, 60, 65, 59, 51, 47, 44],
                        [88, 92, 95, 97, 89, 85, 91, 99, 101, 93, 87, 84, 90, 94, 96, 103, 98, 89, 86, 82],
                        [72, 68, 75, 79, 81, 77, 83, 85, 88, 79, 74, 70, 76, 80, 82, 87, 84, 78, 73, 69]
                    ]
                },
                { transform: { type: 'boxplot', config: { itemNameFormatter: ['Europe', 'Asia', 'Africa',
'N.America', 'S.America'] } } },
                { fromDatasetIndex: 1, fromTransformResult: 1 }
            ],
            tooltip: { trigger: 'item', axisPointer: { type: 'shadow' }, formatter: '{b}<br/>{a}: {c}%' },
            grid: { left: '12%', right: '10%', bottom: '18%' },
            xAxis: { type: 'category', boundaryGap: true, axisLabel: { rotate: 45 } },
            yAxis: { type: 'value', name: 'Vaccination Rate (%)', splitArea: { show: true } },
            series: [
                { name: 'Vaccination', type: 'boxplot', datasetIndex: 1, itemStyle: { color: '#5470C6',
borderWidth: 2 },
                   label: { show: true, position: 'top', formatter: 'Median: {c}' } },
                { name: 'Outliers', type: 'scatter', datasetIndex: 2, symbolSize: 8, itemStyle: { color:
'#EE6666' } }
            ],
            visualMap: { show: false, dimension: 1, min: 40, max: 110, inRange: { colorLightness: [0.5,
0.8] } }
        };

        option && myChart.setOption(option);
        window.addEventListener('resize', function() { myChart.resize(); });
    </script>
</body>
</html>
```

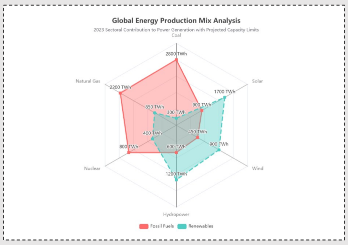


```
<html>
<head>
    <meta charset="utf-8" />
    <title>Renewable Energy Adoption Funnel</title>
    <script src="https://cdn.jsdelivr.net/npm/echarts@5.6.0/dist/echarts.min.js"></script>
    <style>
        html, body { width: 100%; height: 100%; margin: 0; padding: 0; }
        #main { width: 100%; height: 100%; }
    </style>
</head>
<body>
    <div id="main"></div>
    <script>
        var chartDom = document.getElementById('main');
        var myChart = echarts.init(chartDom);
        var option;

        option = {
            title: {
                text: 'Global Renewable Energy Transition',
                subtext: 'From Policy Commitment to Operational Projects (2020-2023)',
                left: 'center'},
            tooltip: {
                trigger: 'item',
                formatter: '{a} <br/>{b} : {c}%'},
            toolbox: {
                feature: {
                    dataView: { readOnly: false },
                    restore: {},
                    saveAsImage: {}}},
            legend: {
                data: ['Policy Commitment', 'Feasibility Study', 'Funding Secured', 'Construction Started',
'Operational'],
                bottom: 10},
            series: [{
                name: 'Project Pipeline',
                type: 'funnel',
                left: '15%',
                top: 100,
                bottom: 100,
                width: '70%',
                min: 0,
                max: 100,
                sort: 'descending',
                gap: 2,
                label: {
                    show: true,
                    position: 'inside',
                    formatter: '{c}%'
                },
                itemStyle: {
                    borderColor: '#fff',
                    borderWidth: 2,
                    color: params => ['#c6e48b', '#7bc96f', '#49afc9', '#2fa6aa', '#19692d'][params.dataIndex]
                },
                emphasis: {
                    label: {
                        fontSize: 18,
                        fontWeight: 'bold'
                    }
                },
                data: [
                    { value: 95, name: 'Policy Commitment' },
                    { value: 78, name: 'Feasibility Study' },
                    { value: 63, name: 'Funding Secured' },
                    { value: 45, name: 'Construction Started' },
                    { value: 28, name: 'Operational' }
                ]
            }]
        };

        option && myChart.setOption(option);
        window.addEventListener('resize', function() { myChart.resize(); });
    </script>
</body>
</html>
```

Figure 13: Example 2 from the Chart-to-Code Dataset.

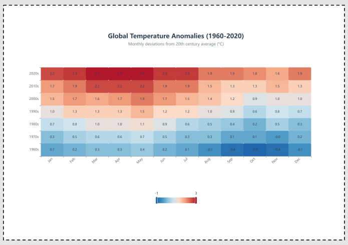


```
<html>
<head>
    <meta charset="utf-8" />
    <title>Energy Production Mix</title>
    <script src="https://cdn.jsdelivr.net/npm/echarts@5.6.0/dist/echarts.min.js"></script>
    <style>
        html, body { width: 100%; height: 100%; margin: 0; padding: 0; }
        #main { width: 100%; height: 100%; }
    </style>
</head>
<body>
    <div id="main"></div>
    <script>
        var chartDom = document.getElementById('main');
        var myChart = echarts.init(chartDom);

        var option = {
            title: {
                text: 'Global Energy Production 2023',
                subtext: 'Source: International Energy Agency (IEA) Estimates',
                left: 'center',
                textStyle: { color: '#2c3e50', fontSize: 22 }
            },
            tooltip: { trigger: 'item', formatter: '{b}: {c} TWh ({d}%)' },
            legend: { orient: 'vertical', left: 'left', top: 'middle' },
            series: [{
                name: 'Energy Source',
                type: 'pie',
                radius: ['30%', '60%'],
                avoidLabelOverlap: false,
                itemStyle: { borderRadius: 8, borderColor: '#fff', borderWidth: 2 },
                label: {
                    show: true,
                    formatter: '{b|{b}}\n{d}%',
                    rich: { b: { fontWeight: 'bold', fontSize: 14 } }
                },
                data: [
                    { value: 15900, name: 'Coal', itemStyle: { color: '#7f8c8d' } },
                    { value: 13200, name: 'Crude Oil', itemStyle: { color: '#2d3436' } },
                    { value: 12800, name: 'Natural Gas', itemStyle: { color: '#e67e22' } },
                    { value: 8900, name: 'Nuclear', itemStyle: { color: '#f1c40f' } },
                    { value: 7200, name: 'Hydro', itemStyle: { color: '#3498db' } },
                    { value: 5100, name: 'Wind', itemStyle: { color: '#7f8c8d' } },
                    { value: 3800, name: 'Solar', itemStyle: { color: '#f39c12' } },
                    { value: 2100, name: 'Biomass', itemStyle: { color: '#27ae60' } }
                ],
                emphasis: {
                    label: { show: true, fontSize: 18 },
                    itemStyle: { shadowBlur: 20, shadowColor: 'rgba(0, 0, 0, 0.3)' }
                }
            }]
        };

        myChart.setOption(option);
        window.addEventListener('resize', () => myChart.resize());
    </script>
</body>
</html>
```

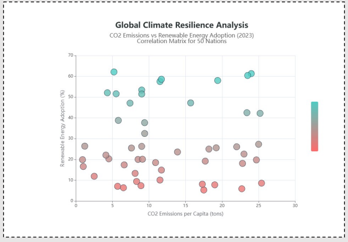


```
<html>
<head>
    <meta charset="utf-8" />
    <title>Energy Mix Radar</title>
    <script src="https://cdn.jsdelivr.net/npm/echarts@5.6.0/dist/echarts.min.js"></script>
    <style>
        html, body { width: 100%; height: 100%; margin: 0; padding: 0; }
        #main { width: 100%; height: 100%; }
    </style>
</head>
<body>
    <div id="main"></div>
    <script>
        var chartDom = document.getElementById('main');
        var myChart = echarts.init(chartDom);

        var option = {
            title: {
                text: 'Global Energy Production Mix Analysis',
                subtext: '2023 Sectoral Contribution to Power Generation with Projected Capacity Limits',
                left: 'center'
            },
            legend: {
                data: ['Fossil Fuels', 'Renewables'],
                bottom: 10
            },
            radar: {
                shape: 'polygon',
                splitNumber: 5,
                axisLine: { lineStyle: { color: 'rgba(100, 100, 100, 0.8)' } },
                splitArea: { show: false },
                indicator: [
                    { name: 'Coal', max: 3500 },
                    { name: 'Natural Gas', max: 2800 },
                    { name: 'Nuclear', max: 1200 },
                    { name: 'Hydropower', max: 1800 },
                    { name: 'Wind', max: 1500 },
                    { name: 'Solar', max: 2500 }
                ]
            },
            series: [{
                type: 'radar',
                color: ['#FF6B6B', '#4ECDC4'],
                areaStyle: { opacity: 0.4 },
                label: { show: true, formatter: '{c} TWh' },
                data: [
                    {
                        value: [2800, 2200, 800, 600, 450, 900],
                        name: 'Fossil Fuels',
                        symbol: 'rect',
                        lineStyle: { width: 3 }
                    },
                    {
                        value: [300, 850, 400, 1200, 900, 1700],
                        name: 'Renewables',
                        symbol: 'roundRect',
                        lineStyle: { type: 'dashed', width: 3 }
                    }
                ]
            }],
            tooltip: { trigger: 'item' }
        };

        myChart.setOption(option);
        window.addEventListener('resize', myChart.resize);
    </script>
</body>
</html>
```

Figure 14: Example 3 from the Chart-to-Code Dataset.



```
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Climate Anomalies Heatmap</title>
   <style>
      #main {
         width: 100%;
         max-width: 1200px;
         height: 700px;
         margin: 20px auto;
         box-shadow: 0 0 15px rgba(0,0,0,0.1);
      }
   </style>
</head>
<body>
   <div id="main"></div>
   <script src="https://cdn.jsdelivr.net/npm/echarts@5/dist/echarts.min.js"></script>
   <script>
      const chart = echarts.init(document.getElementById('main'));
      // prettier-ignore
      const months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'];
      const decades = ['1960s', '1970s', '1980s', '1990s', '2000s', '2010s', '2020s'];
      // Generate synthetic temperature anomaly data ( -0.5 to +2.5°C)
      const data = [];
      for (let decade = 0; decade < 7; decade++) {
         for (let month = 0; month < 12; month++) {
            const base = decade * 0.35 + Math.sin(month * 0.5) * 0.4;
            const anomaly = (base + Math.random() * 0.3 - 0.15).toFixed(1);
            data.push([month, decade, anomaly > 2.5 ? 2.5 : anomaly]);
         }
      }
      const option = {
         title: {
            text: 'Global Temperature Anomalies (1960 -2020)',
            subtext: 'Monthly deviations from 20th century average ( °C)',
            left: 'center',
            textStyle: { fontSize: 22, color: '#2c3e50' },
            subtextStyle: { fontSize: 14, color: '#7f8c8d' }
         },
         grid: { height: '45%', top: '18%' },
         xAxis: {
            type: 'category',
            data: months,
            splitArea: { show: true },
            axisLabel: { rotate: 45 } },
         yAxis: {
            type: 'category',
            data: decades,
            splitArea: { show: true }},
         visualMap: {
            min: -0.5,
            max: 2.5,
            calculable: true,
            orient: 'horizontal',
            left: 'center',
            bottom: '12%',
            inRange: { color: ['#2166ac', '#4393c3', '#92c5de', '#d1e5f0', '#fddbc7', '#f4a582', '#d6604d',
'#b2182b'] }
         },
         series: [{
            type: 'heatmap',
            data: data,
            label: { show: true, color: '#2c3e50', fontSize: 11 },
            itemStyle: { borderRadius: 3 },
            emphasis: {
               itemStyle: { shadowBlur: 12, shadowColor: 'rgba(0,0,0,0.3)' }
            }
         }]
      };
      chart.setOption(option);
      window.addEventListener('resize', () => chart.resize());
   </script>
</body>
</html>
```



```
<html>
<head>
   <meta charset="utf-8" />
   <title>Climate Energy Analysis</title>
   <script src="https://cdn.jsdelivr.net/npm/echarts@5.6.0/dist/echarts.min.js"></script>
   <style>
      html, body { width: 100%; height: 100%; margin: 0; padding: 0 }
      #main { width: 100%; height: 100% }
   </style>
</head>
<body>
   <div id="main"></div>
   <script>
      var chartDom = document.getElementById('main');
      var myChart = echarts.init(chartDom);

      var option = {
         title: {
            text: 'Global Climate Resilience Analysis',
            subtext: 'CO2 Emissions vs Renewable Energy Adoption (2023) \nCorrelation Matrix for 50
Nations',
            left: 'center',
            textStyle: { fontSize: 22, color: '#2c3436' },
            subtextStyle: { fontSize: 16, color: '#666' }
         },
         xAxis: {
            name: 'CO2 Emissions per Capita (tons)',
            nameLocation: 'center',
            nameGap: 30,
            nameTextStyle: { fontSize: 14 },
            axisLabel: { color: '#555' }
         },
         yAxis: {
            name: 'Renewable Energy Adoption (%)',
            nameLocation: 'center',
            nameGap: 30,
            nameTextStyle: { fontSize: 14 },
            axisLabel: { color: '#555' }
         },
         visualMap: {
            min: 0,
            max: 60,
            dimension: 1,
            orient: 'vertical',
            right: 10,
            top: 'center',
            textStyle: { color: '#666' },
            inRange: { color: ['#ff6b6b', '#4ecdc4'] }
         },
         series: [{
            type: 'scatter',
            symbolSize: 18,
            data: Array.from({length: 50}, () => [
               Math.random() * 25 + 0.5,
               Math.random() * 60 + 5
            ]).map(point => [
               Number(point[0].toFixed(1)),
               Number(point[1].toFixed(1))
            ]),
            itemStyle: {
               opacity: 0.8,
               borderColor: '#2c3e50',
               borderWidth: 1
            },
            emphasis: {
               itemStyle: { shadowBlur: 10, shadowColor: 'rgba(0,0,0,0.3)' }
            }
         }],
         grid: { containLabel: true, top: 100 }
      };

      myChart.setOption(option);
      window.addEventListener('resize', function() { myChart.resize() });
   </script>
</body>
</html>
```

Figure 15: Example 4 from the Chart-to-Code Dataset.