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

Anonymous authors
Paper under double-blind review

Abstract

Recently, large language models have shown remarkable reasoning capabilities through long-chain reasoning before responding. However, how to extend this capability to visual reasoning tasks remains an open challenge. Existing multimodal reasoning approaches transfer such visual reasoning task into textual reasoning task via several image-to-text conversions, which often lose critical structural and semantic information embedded in visualizations, especially for tasks like chart question answering that require a large amount of visual details. To bridge this gap, we propose ChartReasoner, a code-driven novel two-stage framework designed to enable precise, interpretable reasoning over charts. We first train a high-fidelity model to convert diverse chart images into structured ECharts codes, preserving both layout and data semantics as lossless as possible. Then, we design a general chart reasoning data synthesis pipeline, which leverages this pretrained transport model to automatically and scalably generate chart reasoning trajectories and utilizes a code validator to filter out low-quality samples. Finally, we train the final multimodal model using a combination of supervised fine-tuning and reinforcement learning on our synthesized chart reasoning dataset and experimental results on four public benchmarks clearly demonstrate the effectiveness of our proposed ChartReasoner. It can preserve the original details of the charts as much as possible and perform comparably with state-of-the-art open-source models while using fewer parameters, approaching the performance of proprietary systems like GPT-4o in out-of-domain settings.

1 Introduction

Chart question answering (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. The CoT reasoning they produce is often shallow and short, resulting in superficial reasoning without genuine logical depth.

Large language models (LLMs) have achieved remarkable success in text-based long-chain reasoning, producing highly accurate, structured, and multi-step solutions to complex problems. This is exemplified by LLMs such as DeepSeek-R1 (Guo et al., 2025). These models decompose complex problems into logical sequential 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. In particular, ChartQA poses unique challenges that cannot be addressed by simple extensions of text-based reasoning. Existing multimodal approaches typically convert visual inputs into textual representations via image-to-text pipelines. While effective in some contexts, this strategy often fails to preserve the structural and semantic fidelity of the original visualizations. As a result, critical layout, spatial, and quantitative details necessary for accurate reasoning in ChartQA are frequently lost, severely limiting the effectiveness of current models in this domain.

Recent advances in multimodal reasoning extend structured thinking from text to vision by converting images into textual representations to enable CoT reasoning. Methods such as R1-OneVision (Chen et al., 2025) translate visual scenes into formal text, while R1-V (Chen et al., 2025), Curr-ReFT (Deng et al., 2025), LMM-R1 (Peng 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, and spatial layouts are frequently abstracted or compressed. This lossy transformation undermines tasks that require precise visual grounding, such as ChartQA or scientific diagram analysis.

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 Chart2Code, 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 ChartThink dataset by applying Chart2Code to various benchmarks, yielding 140K multi-step reasoning samples. We then train the final ChartReasoner model through supervised fine-tuning (SFT) and reinforcement learning (RL) to improve reasoning accuracy, consistency, and interpretability. This structured pipeline enables precise, scalable, and logically grounded ChartQA.

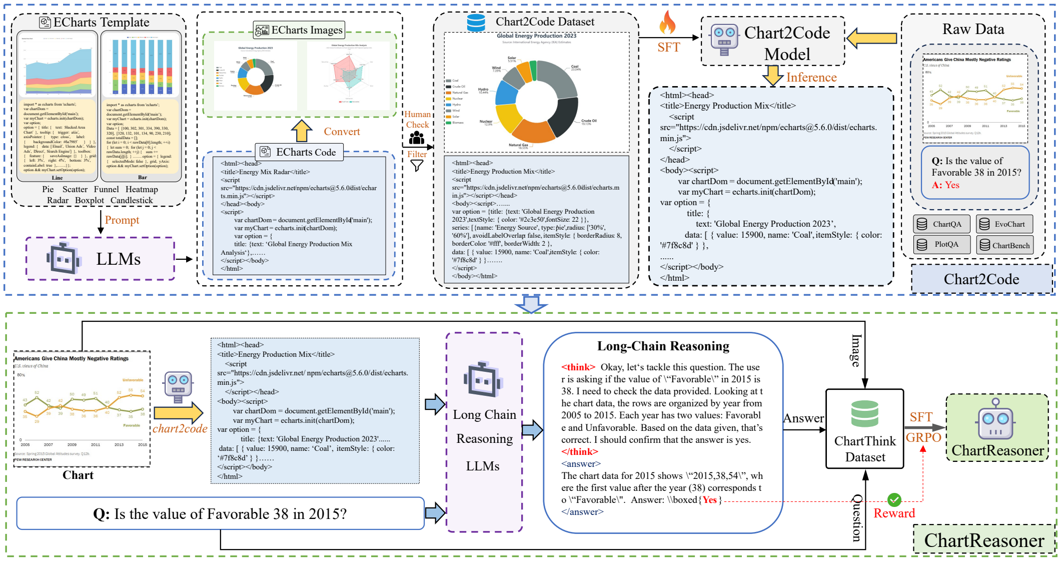
Contributions are as follows:

- We introduce ChartThink, the first large-scale chart reasoning dataset with over 140K multi-step samples, supporting interpretable and logic-driven analysis across diverse chart types. We also construct Chart2Code, a dataset of 110K synthetic charts paired with accurate ECharts code, bridging the visual–textual modality gap.
- We propose ChartReasoner, a two-stage model that first translates chart images into symbolic ECharts code using Chart2Code, then performs multi-step reasoning on the structured representation. This improves accuracy and generalization across ChartQA tasks.
- ChartReasoner is comparable to state-of-the-art open-source models on ChartQA, ChartBench, EvoChart-QA, and ChartQAPro, while using fewer parameters, and approaches the performance of proprietary systems like GPT-4o in out-of-domain settings.

2 Related Work

ChartQA. To enhance MLLMs' chart understanding, datasets like ChartQA and others have been proposed (Kahou et al., 2017; Kafle et al., 2018; Methani et al., 2020; Chaudhry et al., 2020; Masry et al., 2022), covering diverse chart types and visual reasoning tasks. However, most focus on single-value or label-based answers and lack support for complex multi-step reasoning. Recent efforts such as ChartX and related work (Xia et al., 2024; Ahmed et al., 2023; Masry et al., 2023; Meng et al., 2024; Huang et al., 2025a) scale up data via synthetic generation, template-based QA, and CoT annotations, though reasoning depth remains limited. On the model side, compact models like ChartReader and others (Cheng et al., 2023; Liu et al., 2023; Baechler et al., 2024; Masry et al., 2023) show strong results on early benchmarks. LLaVA-based models including ChartLlama and its variants (Han et al., 2023; Carbune et al., 2024; Masry et al., 2024; Meng et al., 2024; Zhang et al., 2024) further enhance multimodal alignment. More recently, generalist vision-language models like Phi-3 Vision (Chen et al., 2024a) have also achieved promising performance. Nevertheless, current models still struggle with long-chain reasoning involving multi-cue integration and numerical-logical inference.

Multimodal Long-Chain Reasoning. Long-chain reasoning has gained momentum in NLP with the emergence of DeepSeek-R1 (Guo et al., 2025), which emphasizes structured intermediate steps. This paradigm has been extended to vision-language models (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) applies Group Relative Policy Optimization (GRPO) (Shao et al., 2024) to object counting, demonstrating that small models can outperform larger ones with effective reinforcement learning. VisualThinker-R1-Zero (Zhou et al., 2025), Curr-ReFT (Deng et al., 2025), LMM-R1 (Peng et al., 2025), and



```
import * as echarts from 'echarts';
var chartDom =
document.getElementById('main');
var myChart = echarts.init(chartDom);
var option;
option = { title: { text: Slacked Area
Chart' }, tooltip: { trigger: axis',
axisPointer: { type: cross', label:
{ backgroundColor: #6a7985' } } },
legend: { data: [ Email', Union Ads', Video
Ads', Direct', Search Engine'] }, toolbox:
{ feature: { saveAsImage: {} } }, grid:
{ left: 3%', right: 4%', bottom: 3%',
containLabel: true },.......}};
option && myChart.setOption(option);
```

```
import * as echarts from 'echarts';
var chartDom =
document.getElementById('main');
var myChart = echarts.init(chartDom);
var option;
Data = [ [100, 302, 301, 334, 390, 330,
320], [320, 132, 101, 134, 90, 230, 210];
const totalData = [];
for (let i = 0; i < rawData[0].length; ++i)
{ let sum = 0; for (let j = 0; j <
rawData.length; ++j) { sum +=
rawData[j][i]; } ....... option = { legend:
{ selectedMode: false }, grid, yAxis:
option && myChart.setOption(option);
```

```
<html><head>
<title>Energy Mix Radar</title>
<script
src="https://cdn.jsdelivr.net/npm/echarts@5.6.0/dist/echar
ts.min.js"></script>
</head><body>
<script>
      var chartDom = document.getElementById('main');
      var myChart = echarts.init(chartDom);
      var option = {
      title: {text: 'Global Energy Production Mix
Analysis'},......
</script></body>
</html>
```

```
<html><head>
<title>Energy Production Mix</title>
<script
src="https://cdn.jsdelivr.net/npm/echarts@5.6.0/dist/echarts.m
in.js"></script></head>
<body><script>.......
var option = {title: {text: 'Global Energy Production
2023',textStyle: { color: '#2c3e50',fontSize: 22 }},
series: [{name: 'Energy Source', type: pie',radius: ['30%',
'60%'], avoidLabelOverlap: false, itemStyle: { borderRadius: 8,
borderColor: '#fff', borderWidth: 2 },
data: [ { value: 15900, name: 'Coal',itemStyle: { color:
'#7f8c8d' } }.......
</script>
</body></html>
```

```
<html><head>
<title>Energy Production Mix</title>
   <script
src="https://cdn.jsdelivr.net/npm/echarts@5.6.0/dist/echarts.
min.js">
   </script>
</head>
<body><script>
      var chartDom = document.getElementById('main');
      var myChart = echarts.init(chartDom);
var option = {
         title: {
            text: 'Global Energy Production 2023',
         data: [ { value: 15900, name: 'Coal', itemStyle: { color:
'#7f8c8d' } },
......
</script></body>
</html>
```

```
<html><head>
<title>Energy Production Mix</title>
   <script
src="https://cdn.jsdelivr.net/ npm/echarts@5.6.0/dist/echarts.
min.js">
   </script></head>
<body><script>
      var chartDom = document.getElementById('main');
      var myChart = echarts.init(chartDom);
var option = {
         title: {text: 'Global Energy Production 2023'......
 data: [ { value: 15900, name: 'Coal', itemStyle: { color:
'#7f8c8d' } }......
</script></body>
</html>
```

```
<think> Okay, let's tackle this question. The use
r is asking if the value of \"Favorable\" in 2015 is
38. I need to check the data provided. Looking at t
he chart data, the rows are organized by year from
2005 to 2015. Each year has two values: Favorabl
e and Unfavorable. Based on the data given, that's
correct. I should confirm that the answer is yes.
</think>
<answer>
The chart data for 2015 shows \"2015,38,54\", wh
ere the first value after the year (38) corresponds t
o \"Favorable\". Answer: \\boxed{Yes}
</answer>
```

Figure 1: Overview of the data construction pipeline and model training. (1) Chart2Code: We generate a synthetic dataset by rendering chart specifications into paired images and structured code using a prompt-based pipeline. A hybrid filtering strategy is applied to ensure data quality. A vision-language model is then fine-tuned on these image–code pairs. (2) ChartReasoner: Existing ChartQA datasets are converted into structured code using the trained Chart2Code model. Reasoning traces are then generated over code representations to build a symbolic reasoning dataset. The final model is trained in two stages.

MMEureka (Meng et al., 2025) further explore RL-driven reasoning, revealing "visual aha moments" where longer outputs indicate deeper reasoning. Yet most methods still target natural or generic images and remain challenged by structured inputs such as charts.

To address the challenges of chart understanding and long-chain reasoning, we propose ChartReasoner, a code-driven, two-stage framework that enhances the reasoning capabilities of MLLMs. In stage one, Chart2Code translates diverse chart images into structured ECharts code, preserving visual layouts and data semantics. Compared with prior methods such as ChartMimic (Yang et al., 2025a), Plot2Code (Wu et al., 2025), and ChartX (Xia et al., 2024), which focus on layout heuristics or require costly supervision (Li et al., 2025; Zhang et al., 2025), Chart2Code offers greater flexibility and fidelity via executable, symbolic code. While ChartCoder (Zhao et al., 2025) adopts a similar code-centric approach, its reliance on fixed templates limits generalization. In stage two, we generate 140K multi-step reasoning samples by applying Chart2Code to existing ChartQA benchmarks, forming the ChartThink dataset. ChartReasoner is then trained with supervised and reinforcement learning, yielding accurate, consistent, and interpretable reasoning. This structured pipeline bridges the visual–textual gap and supports logically grounded ChartQA.

3 Our Method

Understanding charts remains a core challenge for MLLMs due to the gap between visual inputs and symbolic semantics. To bridge this gap, we propose a code-driven, two-stage framework that integrates visual perception with symbolic reasoning through structured chart representations. In the first stage, we introduce Chart2Code, a model that converts chart images into executable ECharts code, preserving both visual layout and semantic structure. To train it, we generate a 110K synthetic dataset via a prompt-based pipeline, rendering chart specifications into image–code pairs and applying a hybrid filtering strategy to ensure quality. The model is fine-tuned with a frozen visual encoder and a trainable decoder for accurate symbolic generation. In the second stage, we use Chart2Code to construct ChartThink, a 140K dataset of multi-step reasoning samples. These are derived by extracting structured code from existing ChartQA benchmarks and prompting a language model to generate chain-of-thought reasoning over the code. This symbolic abstraction enables precise and lossless reasoning over chart semantics. Our final model, ChartReasoner, is trained in two

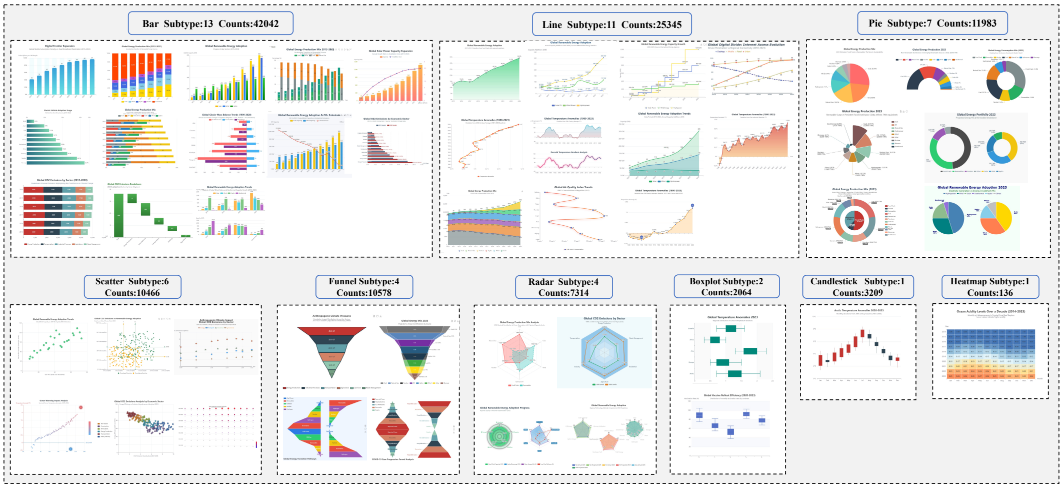


Figure 2: Statistics and distribution of chart types and subtypes in the Chart2Code dataset.

phases: supervised fine-tuning to establish logical competence, followed by reinforcement learning with rule-guided rewards to deepen reasoning capabilities. Overall, our framework treats code as a compositional and interpretable bridge between vision and language, enabling accurate and logically grounded chart understanding, as illustrated in Figure 1.

3.1 CHART2CODE

Echarts-format Chart Generation. We begin with a template library $\mathcal{T} = \{T_1, T_2, \ldots, T_K\}$, which spans K chart templates across 9 major categories and 49 subtypes. For each template $T_k \in \mathcal{T}$, we construct a prompt p_k to guide an LLM in generating diverse ECharts code. The detailed prompt design is provided in Appendix E. The ECharts code c_j for a sample j is obtained as:

$$c_j = G_{\text{LLM}}(p_k), \tag{1}$$

where $G_{\text{LLM}}(\cdot)$ denotes the LLM that produces structured chart code given the input prompt.

Quality Filtering Pipeline. The generated ECharts code is rendered into images and 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 result, 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 F.

Chart2Code Model. To enable high-fidelity chart reconstruction from images, we construct a large-scale chart2code dataset and use it to train a multimodal model capable of translating chart images into their corresponding ECharts code. We fine-tune a vision-language model on this 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 $\mathcal{L}_{\text{C2C}}$:

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

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 chart2code 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 ChartThink Construction and Collection

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.

ChartThink Construction. We begin by consolidating existing datasets into a unified collection, denoted as $\mathcal{D}_{\text{orig}} = \{(x_k, q_k, a_k)\}_k$, where x_k represents a chart image, q_k is a corresponding question, and a_k is the ground-truth answer. Each question is annotated with a reasoning type, and each chart is labeled with a structural type. To ensure broad coverage and balanced representation, we perform stratified sampling across both dimensions to select a representative subset. For each sampled chart image x_k, we apply the trained Chart2Code model to generate its corresponding ECharts specification c_k. The generated code, together with the question q_k, is then provided as input to a long-chain reasoning LLM, which outputs a reasoning path r_k and a predicted answer $\tilde{a}_k$:

$$(r_k, \tilde{a}_k) = G_{\text{LC-R}}(\text{Prompt}(\text{Chart2Code}(x_k), q_k)) \tag{3}$$

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 ChartThink, is defined as:

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

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.

ChartThink Collection. We construct the ChartThink 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. Figure 3 presents the ChartThink dataset statistics and its distribution over four reasoning categories and seven chart types, providing a strong foundation for training models on complex ChartQA.

3.3 ChartReasoner

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 $\mathcal{L}_{\text{SFT}}$, is defined as:

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

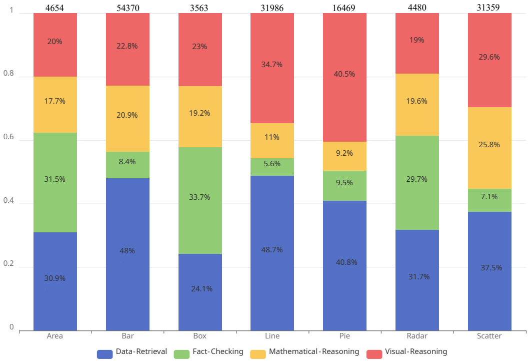


Figure 3: ChartThink dataset statistics and distribution.

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 dimensions such as factual accuracy, formatting correctness, and response length. These reward signals guide the model to suppress hallucinations and over-reasoning, promoting disciplined and generalizable reasoning. 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. We evaluate ChartReasoner on four benchmarks—ChartQA (Masry et al., 2022), EvoChart-QA (Huang et al., 2025a), ChartQAPro (Masry et al., 2025a), and ChartBench (Xu et al., 2023)—covering diverse chart types and reasoning skills, from simple plots to real-world dashboards and multi-chart compositions. EvoChart-QA and ChartQAPro serve as out-of-domain tests for generalization to realistic, structurally diverse charts; ChartQAPro also includes multi-turn, hypothetical, and unanswerable queries. ChartBench provides large-scale, type-diverse evaluation. We further assess Chart2Code on EvoChart-QA to measure chart reconstruction. Additional setup details are in Appendix B.

Evaluation Metrics & Baselines. For ChartQA, we follow the official protocol for each benchmark. For Chart2Code, 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 D.

We benchmark our model against a wide range of MLLMs, including proprietary models like Claude-3.5-Sonnet[2], 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 such as 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

[1] The version is gpt-4-vision-preview

[2] https://www.anthropic.com/news/claude-3-5-sonnet

Model Name	**Size(B)**	**Evochart-QA**	**ChartQA**	**ChartBench**	**ChartQAPro**
Closed-source					
Claude-3.5-Sonnet	–	56.96	**90.80**	56.95	43.58
Gemini-2.0-Flash (Team et al., 2024)	–	**64.64**	84.80	55.63	**46.85**
Gemini-1.5-Flash (Team et al., 2024)	–	27.90	79.00	51.42	42.96
Gemini-1.5-Pro (Team et al., 2024)	–	32.20	87.20	56.26	43.24
GPT-4-turbo (Achiam et al., 2023)	–	40.30	62.30	43.14	32.14
GPT-4o (Achiam et al., 2023)	–	49.80	85.70	**59.45**	37.67
Open-source					
Intern-VL2.5 (Chen et al., 2024a)	78	**57.44**	**88.30**	**65.57**	**45.31**
QvQ-Preview (Bai et al., 2025)	72	54.32	86.48	42.40	41.26
Qwen2-VL (Wang et al., 2024)	72	54.00	**88.30**	54.77	39.42
Intern-VL2 (Chen et al., 2024b)	40	49.00	86.20	39.63	35.21
CogVLM2 (Hong et al., 2024)	19	21.90	81.00	35.14	31.47
Intern-VL2 (Chen et al., 2024b)	8	38.60	81.50	37.80	22.53
Intern-VL2.5 (Chen et al., 2024a)	8	45.20	84.80	52.96	35.67
LLaVA-v1.5 (Liu et al., 2024)	7	17.40	55.32	23.39	16.33
Internlm-XComp.-v2 (Dong et al., 2024)	7	27.80	72.64	47.78	23.75
QwenVL-Chat (Bai et al., 2023)	7	19.70	83.00	26.98	35.59
Qwen2.5-VL (Bai et al., 2025)	7	46.80	85.00	54.06	36.61
Phi3-Vision (Abdin et al., 2024)	4	39.50	81.40	35.06	24.73
Chart MLLMs					
ChartLlama (Han et al., 2023)	13	9.50	69.66	21.71	7.28
ChartAst-S (Meng et al., 2024)	13	12.90	79.90	11.27	8.15
ChartIns-Llama2 (Masry et al., 2024)	7	16.80	66.64	25.28	4.88
EvoChart (Huang et al., 2025a)	4	**54.20**	81.50	36.14	25.05
ChartIns-FlanT5 (Masry et al., 2024)	3	24.30	64.20	22.15	4.27
ChartGemma (Masry et al., 2025b)	3	30.60	80.16	35.24	6.84
TinyChart (Abdin et al., 2024)	3	25.50	83.60	28.86	13.25
ChartReasoner-SFT(Ours)	7	47.04	86.76	55.10	37.94
ChartReasoner-GRPO(Ours)	7	48.10	**86.93**	**55.20**	**39.97**

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

include domain-specific baselines: 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, we use ChartCoder (Zhao et al., 2025) as the main baseline. To assess the structural richness of our dataset, we perform controlled training using Qwen2.5-VL-7B (Bai et al., 2025) on both EvoChart and our dataset.

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 a 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 improving reasoning and abstraction abilities to enhance chart-centric generalization.

Models refined using GRPO after SFT consistently outperform those trained with SFT alone across all evaluation benchmarks. In addition to improvements in accuracy, GRPO significantly enhances the quality of reasoning by minimizing excessive explanations and fostering outputs that are more structured and precise. Our further analysis on the ChartQA test set reveals that GRPO reduces the length of excessively long reasoning chains, with the average token length decreasing from 699.03 to 618.22. Moreover, GRPO completely eliminates the issue of truncated responses, a problem observed in the SFT model, where 59 instances of truncation occurred, compared to zero with GRPO. These results demonstrate the dual benefits of GRPO: improving overall performance while optimizing the reasoning process, enabling the model to generate more concise, logically consistent, and reliable outputs, particularly in visual reasoning tasks.

4.3 Ablation Study

Chart-to-Code Performance Evaluation. We report comprehensive results in Table 2, aggregating comparisons across datasets and training scales. For a real-world EvoChart-QA test set, Chart2Code is evaluated using GPT-4V visual-similarity scores and pass rates.

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.

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	30k	2.39	3.12	2.24	2.81	1.39	88.20%	49
Chart2Code	50k	3.62	4.37	3.81	4.17	2.13	90.60%	49
Chart2Code	70k	4.21	5.17	4.20	4.23	3.24	91.00%	49
Chart2Code	110k	**4.34**	**5.26**	**4.21**	**5.12**	**3.77**	**92.40%**	49

Table 2: Chart2Code Performance on EvoChart-QA: GPT-4V Similarity Scores (including Breakdown by Chart Types) and Overall Pass Rates.

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 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

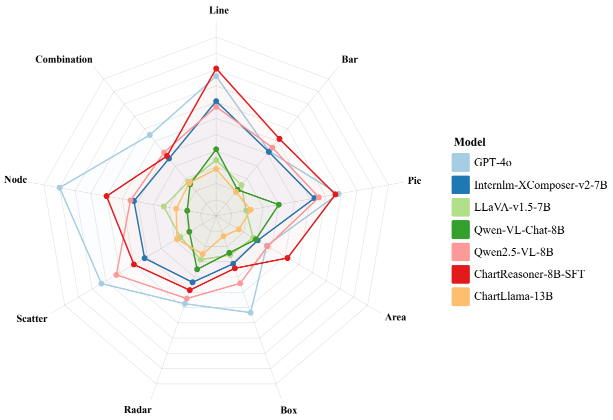


Figure 4: ChartBench Performance Across Chart Types.

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.

a strong correlation between reconstruction reliability and downstream performance, underscoring the importance of both visual complexity and data distribution in ChartQA tasks.

Impact of Different Dataset Sources. We further investigate how the choice of training datasets influences downstream chart reasoning performance when generating chain-of-thought examples. To this end, we sample 20k instances from ChartQA, EvoChart, ChartBench, and PlotQA, and convert them into reasoning examples using our chart-to-code distillation pipeline. As shown in Table 3, using training data that share the same distribution as the evaluation benchmark consistently leads to the best performance, highlighting the importance of distribution alignment. Among the evaluated datasets, PlotQA yields the lowest performance across all benchmarks. This outcome is likely due to its synthetic construction, limited visual variety, and narrow set of chart types, which are primarily restricted to bar, line, and dot charts. These characteristics make it less representative of real-world chart scenarios. In comparison, training data derived from EvoChart achieve better generalization, especially on EvoChart-QA and ChartQAPro. EvoChart includes a broader range of chart types, such as pie and scatter, and its charts are more visually aligned with those found in practical applications, which improves cross-domain performance. ChartBench offers strong results when evaluated on its own distribution but shows reduced effectiveness on other benchmarks, suggesting limited transferability. Overall, these findings emphasize the importance of dataset diversity when training chart reasoning models capable of robust generalization.

5 Conclusion

We present ChartReasoner, a code-driven two-stage framework that bridges visual chart understanding and long-chain reasoning in multimodal LLMs. Our approach introduces Chart2Code, which converts chart images into high-fidelity ECharts code, and builds ChartThink, a dataset of 140K+ multi-step reasoning samples, which enables precise, interpretable, and scalable ChartQA. Unlike prior methods that rely on shallow CoT or lossy image-to-text conversion, ChartReasoner uses structured symbolic representations to preserve layout, semantics, and quantitative details. Extensive evaluations on four benchmarks show strong generalization and reasoning performance. The framework supports faithful information extraction and logical inference, yielding more accurate and transparent decisions. Overall, our results highlight the value of symbolic representation and structured reasoning by tightly integrating visual parsing with logical inference.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint, arXiv:2404.14219*, 2024. URL https://arxiv.org/abs/2404.14219.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *Preprint, arXiv:2303.08774*, 2023. URL https://arxiv.org/abs/2303.08774.

Saleem Ahmed, Bhavin Jawade, Shubham Pandey, Srirangaraj Setlur, and Venu Govindaraju. Realcqa: Scientific chart question answering as a test-bed for first-order logic. In *Proceedings of the ICDAR*, pp. 66–83, 2023. doi: https://doi.org/10.1007/978-3-031-41682-8_5. URL https://link.springer.com/chapter/10.1007/978-3-031-41682-8_5.

Gilles Baechler, Srinivas Sunkara, Maria Wang, Fedir Zubach, Hassan Mansoor, Vincent Etter, Victor Cărbune, Jason Lin, Jindong Chen, and Abhanshu Sharma. Screenai: A vision-language model for ui and infographics understanding. In *Proceedings of the IJCAI*, pp. 3058–3068, 2024. doi: 10.24963/ijcai.2024/339. URL https://www.ijcai.org/proceedings/2024/339.

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

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *Preprint, arXiv:2502.13923*, 2025. URL https://arxiv.org/abs/2502.13923.

Victor Carbune, Hassan Mansoor, Fangyu Liu, Rahul Aralikatte, Gilles Baechler, Jindong Chen, and Abhanshu Sharma. Chart-based reasoning: Transferring capabilities from llms to vlms. In *Findings of the NACCL*, pp. 989–1004, 2024. doi: 10.18653/v1/2024.findings-naacl.62. URL https://aclanthology.org/2024.findings-naacl.62.

Ritwick Chaudhry, Sumit Shekhar, Utkarsh Gupta, Pranav Maneriker, Prann Bansal, and Ajay Joshi. Leaf-qa: Locate, encode & attend for figure question answering. In *Proceedings of the WACV*, pp. 3501–3510, 2020. doi: 10.1109/WACV45572.2020.9093269. URL https://openaccess.thecvf.com/content_WACV_2020/papers/Chaudhry_LEAF-QA_Locate_Encode__Attend_for_Figure_Question_Answering_WACV_2020_paper.pdf.

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

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *Preprint, arXiv:2412.05271*, 2024a. URL https://arxiv.org/abs/2412.05271.

Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, et al. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *Science China Information Sciences*, 2024b. doi: https://doi.org/10.1007/s11432-024-4231-5. URL https://link.springer.com/article/10.1007/s11432-024-4231-5.

Zhi-Qi Cheng, Qi Dai, and Alexander G Hauptmann. Chartreader: A unified framework for chart derendering and comprehension without heuristic rules. In *Proceedings of the ICCV*, pp. 22145–22156, 2023. doi: 10.1109/ICCV51070.2023.02029. URL https://openaccess.thecvf.com/content/ICCV2023/html/Cheng_ChartReader_A_Unified_Framework_for_Chart_Derendering_and_Comprehension_without_ICCV_2023_paper.html.

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

Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Bin Wang, Linke Ouyang, Xilin Wei, Songyang Zhang, Haodong Duan, Maosong Cao, et al. Internlm-xcomposer2: Mastering free-form text-image composition and comprehension in vision-language large model. *Preprint, arXiv:2401.16420*, 2024. URL https://arxiv.org/abs/2401.16420.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint, arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

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. *Preprint, arXiv:2311.16483*, 2023. URL https://arxiv.org/abs/2311.16483.

Wenyi Hong, Weihan Wang, Ming Ding, Wenmeng Yu, Qingsong Lv, Yan Wang, Yean Cheng, Shiyu Huang, Junhui Ji, Zhao Xue, et al. Cogvlm2: Visual language models for image and video understanding. *Preprint, arXiv:2408.16500*, 2024. URL https://arxiv.org/abs/2408.16500.

Muye Huang, Han Lai, Xinyu Zhang, Wenjun Wu, Jie Ma, Lingling Zhang, and Jun Liu. Evochart: A benchmark and a self-training approach towards real-world chart understanding. In *Proceedings of the AAAI*, pp. 3680–3688, 2025a. doi: https://doi.org/10.1609/aaai.v39i4.32383. URL https://ojs.aaai.org/index.php/AAAI/article/view/32383.

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

Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. Dvqa: Understanding data visualizations via question answering. In *Proceedings of the CVPR*, pp. 5648–5656, 2018. doi: 10.1109/CVPR.2018.00592. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Kafle_DVQA_Understanding_Data_CVPR_2018_paper.html.

Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Ákos Kádár, Adam Trischler, and Yoshua Bengio. Figureqa: An annotated figure dataset for visual reasoning. *Preprint, arXiv:1710.07300*, 2017. URL https://arxiv.org/abs/1710.07300.

Bingxuan Li, Yiwei Wang, Jiuxiang Gu, Kai-Wei Chang, and Nanyun Peng. Metal: A multi-agent framework for chart generation with test-time scaling, 2025. URL https://arxiv.org/abs/2502.17651.

Fangyu Liu, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Yasemin Altun, Nigel Collier, and Julian Martin Eisenschlos. Matcha: Enhancing visual language pretraining with math reasoning and chart derendering. In *Proceedings of the ACL*, pp. 12756–12770, 2023. doi: 10.18653/v1/2023.acl-long.714. URL https://aclanthology.org/2023.acl-long.714.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the CVPR*, pp. 26286–26296, 2024. doi: 10.1109/CVPR52733.2024.02484. URL https://openaccess.thecvf.com/content/CVPR2024/html/Liu_Improved_Baselines_with_Visual_Instruction_Tuning_CVPR_2024_paper.html.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the ICLR*, 2019. URL https://arxiv.org/abs/1711.05101.

Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the ACL*, pp. 2263–2279, 2022. doi: 10.18653/v1/2022.findings-acl.177. URL https://aclanthology.org/2022.findings-acl.177.

Ahmed Masry, Parsa Kavehzadeh, Xuan Long Do, Enamul Hoque, and Shafiq Joty. Unichart: A universal vision-language pretrained model for chart comprehension and reasoning. In *Proceedings of the EMNLP*, pp. 14662–14684, 2023. doi: 10.18653/v1/2023.emnlp-main.906. URL https://aclanthology.org/2023.emnlp-main.906.

Ahmed Masry, Mehrad Shahmohammadi, Md Rizwan Parvez, Enamul Hoque, and Shafiq Joty. Chartinstruct: Instruction tuning for chart comprehension and reasoning. In *Findings of the ACL*, pp. 10387–10409, 2024. doi: 10.18653/v1/2024.findings-acl.619. URL https://aclanthology.org/2024.findings-acl.619.

Ahmed Masry, Mohammed Saidul Islam, Mahir Ahmed, Aayush Bajaj, Firoz Kabir, Aaryaman Kartha, Md Tahmid Rahman Laskar, Mizanur Rahman, Shadikur Rahman, Mehrad Shahmohammadi, et al. Chartqapro: A more diverse and challenging benchmark for chart question answering. *Preprint, arXiv:2504.05506*, 2025a. URL https://arxiv.org/abs/2504.05506.

Ahmed Masry, Megh Thakkar, Aayush Bajaj, Aaryaman Kartha, Enamul Hoque, and Shafiq Joty. Chartgemma: Visual instruction-tuning for chart reasoning in the wild. In *Proceedings of the ACL*, pp. 625–643, 2025b. URL https://aclanthology.org/2025.coling-industry.54.

Fanqing Meng, Wenqi Shao, Quanfeng Lu, Peng Gao, Kaipeng Zhang, Yu Qiao, and Ping Luo. Chartassisstant: A universal chart multimodal language model via chart-to-table pre-training and multitask instruction tuning. In *Findings of the ACL*, pp. 7775–7803, 2024. doi: 10.18653/v1/2024.findings-acl.463. URL https://aclanthology.org/2024.findings-acl.463.

Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Tiancheng Han, Botian Shi, Wenhai Wang, Junjun He, et al. Mm-eureka: Exploring the frontiers of multimodal reasoning with rule-based reinforcement learning. *Preprint, arXiv:2503.07365*, 2025. URL https://arxiv.org/abs/2503.07365.

Nitesh Methani, Pritha Ganguly, Mitesh M Khapra, and Pratyush Kumar. Plotqa: Reasoning over scientific plots. In *Proceedings of the WACV*, pp. 1516–1525, 2020. doi: 10.1109/WACV45572.2020.9093523. URL https://openaccess.thecvf.com/content_WACV_2020/papers/Methani_PlotQA_Reasoning_over_Scientific_Plots_WACV_2020_paper.pdf.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *Preprint, arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint, arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint, arXiv:2403.05530*, 2024. URL https://arxiv.org/abs/2403.05530.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *Preprint, arXiv:2409.12191*, 2024. URL https://arxiv.org/abs/2409.12191.

Chengyue Wu, Yixiao Ge, Qiushan Guo, Jiahao Wang, Zhixuan Liang, Zeyu Lu, Ying Shan, and Ping Luo. Plot2code: A comprehensive benchmark for evaluating multi-modal large language models in code generation from scientific plots. In *Findings of the NACCL*, pp. 3006–3028, 2025. URL `https://aclanthology.org/2025.findings-naacl.164`.

Renqiu Xia, Bo Zhang, Hancheng Ye, Xiangchao Yan, Qi Liu, Hongbin Zhou, Zijun Chen, Peng Ye, Min Dou, Botian Shi, et al. Chartx & chartvlm: A versatile benchmark and foundation model for complicated chart reasoning. *Preprint, arXiv:2402.12185*, 2024. URL `https://arxiv.org/abs/2402.12185`.

Zhengzhuo Xu, Sinan Du, Yiyan Qi, Chengjin Xu, Chun Yuan, and Jian Guo. Chartbench: A benchmark for complex visual reasoning in charts. *Preprint, arXiv:2312.15915*, 2023. URL `https://arxiv.org/abs/2312.15915`.

Cheng Yang, Chufan Shi, Yaxin Liu, Bo Shui, Junjie Wang, Mohan Jing, Linran Xu, Xinyu Zhu, Siheng Li, Yuxiang Zhang, et al. Chartmimic: Evaluating lmm's cross-modal reasoning capability via chart-to-code generation. In *Proceedings of the ICLR*, 2025a. URL `https://arxiv.org/abs/2406.09961`.

Yi Yang, Xiaoxuan He, Hongkun Pan, Xiyan Jiang, Yan Deng, Xingtao Yang, Haoyu Lu, Dacheng Yin, Fengyun Rao, Minfeng Zhu, et al. R1-onevision: Advancing generalized multimodal reasoning through cross-modal formalization. *Preprint, arXiv:2503.10615*, 2025b. URL `https://arxiv.org/abs/2503.10615`.

Liang Zhang, Anwen Hu, Haiyang Xu, Ming Yan, Yichen Xu, Qin Jin, Ji Zhang, and Fei Huang. Tinychart: Efficient chart understanding with visual token merging and program-of-thoughts learning. In *Proceedings of the ACL*, pp. 1882–1898, 2024. doi: 10.18653/v1/2024.emnlp-main.112. URL `https://aclanthology.org/2024.emnlp-main.112`.

Zhihan Zhang, Yixin Cao, and Lizi Liao. Enhancing chart-to-code generation in multimodal large language models via iterative dual preference learning. *Preprint, arXiv:2504.02906*, 2025. URL `https://arxiv.org/abs/2504.02906`.

Xuanle Zhao, Xianzhen Luo, Qi Shi, Chi Chen, Shuo Wang, Wanxiang Che, Zhiyuan Liu, and Maosong Sun. Chartcoder: Advancing multimodal large language model for chart-to-code generation. In *Proceedings of the ACL*, 2025. URL `https://arxiv.org/abs/2501.06598`.

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

A Appendix

B Implementation Details.

We use DeepSeek-R1 as a controllable code and reasoning generator in both stages of data construction. 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 & 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.

C 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 Figure 6–Figure 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-cup 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.

D 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.

E 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 the 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.

F Chart2Code Dataset Detailed Case

To further illustrate the design of our Chart2Code 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. Figure 12–Figure 15 present more detailed examples from the Chart2Code dataset.

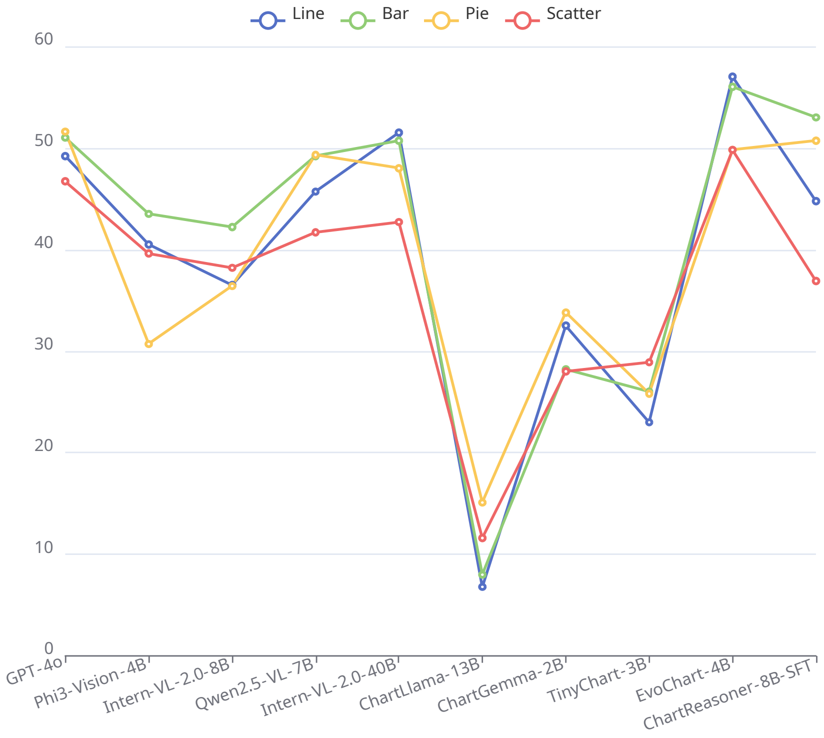
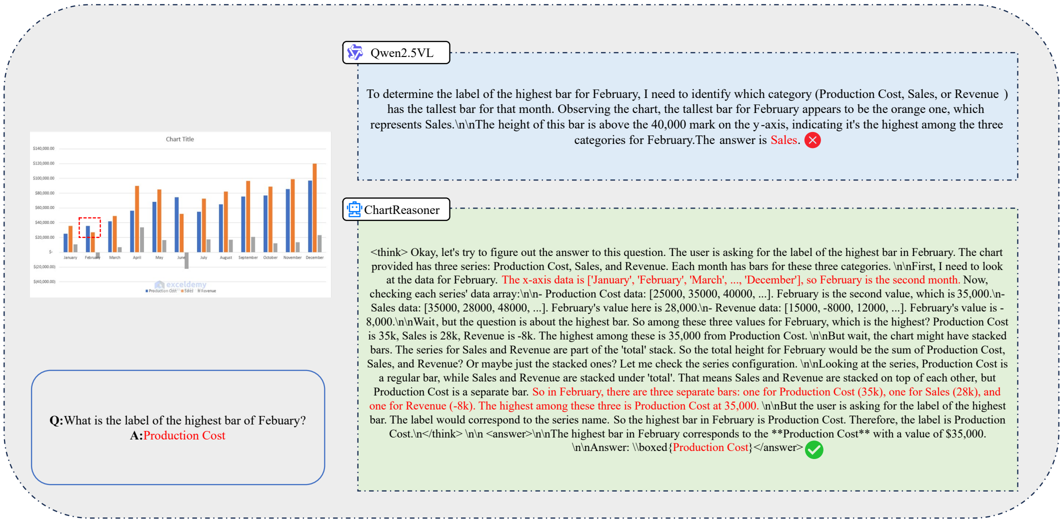


Figure 5: EvoChart Performance Across Chart Types.



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

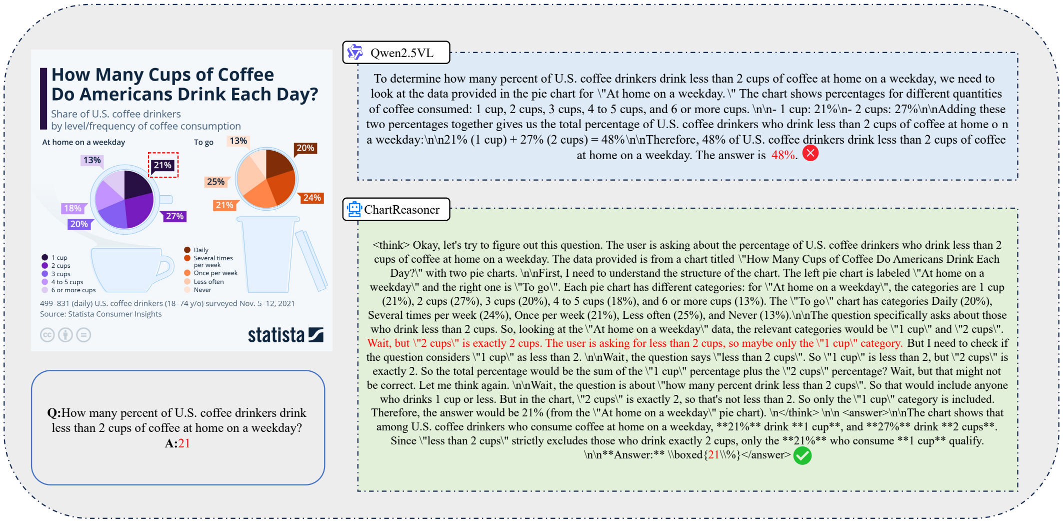
Qwen2.5VL

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. ✗

ChartReasoner

<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 on a Bar Chart.



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

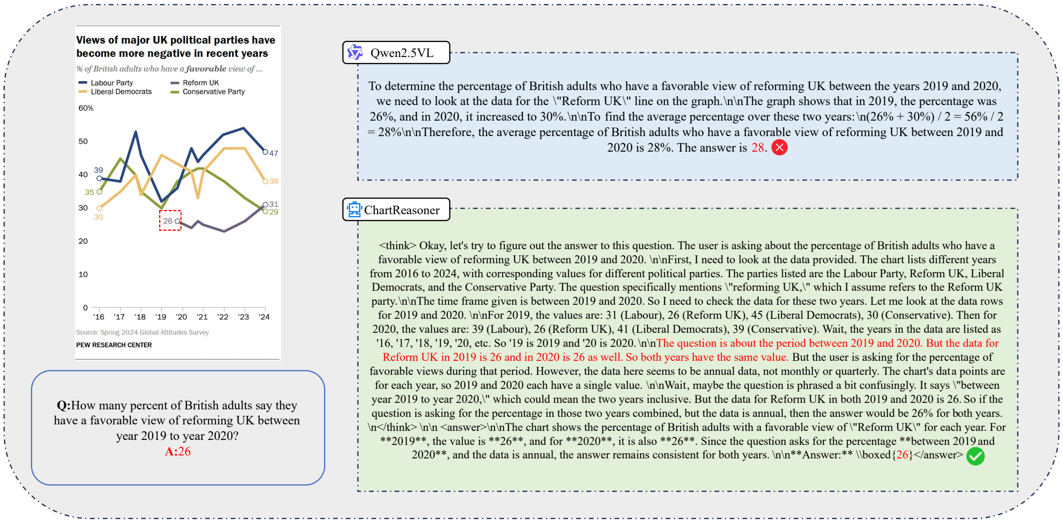
Qwen2.5VL

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 on a Pie Chart.



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

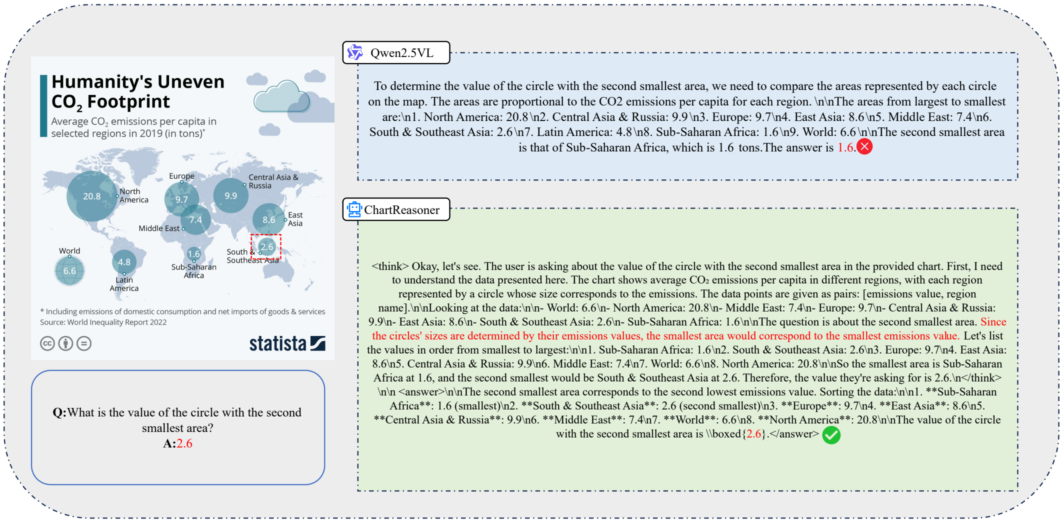
Qwen2.5VL

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 on a Line Chart.


Humanity's Uneven CO2 Footprint
Average CO2 emissions per capita in selected regions in 2019 (in tons)*
20.8
North America
Europe
9.7
Central Asia & Russia
9.9
7.4
Middle East
8.6
East Asia
2.6
South & Southeast Asia
World
6.6
4.8
Latin America
1.6
Sub-Saharan Africa
* Including emissions of domestic consumption and net imports of goods & services
Source: World Inequality Report 2022
statista

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 on a Scatter Chart.

Prompt (GPT-4V Visual Evaluation.)

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 images. 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.

Prompt (ECharts Code Generation.)

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). Choose one theme and dataset from the list below, or combine multiple for richer context, but feel free to create your own variation:

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 and 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 and Demographics: world population growth, urban vs rural distribution, age pyramids by country, migration patterns, household income distribution, gender ratio statistics, life expectancy trends

Economics and Finance: stock market indices (e.g., SandP 500, FTSE 100), GDP per capita, inflation rates, foreign direct investment, income inequality (Gini coefficient), cryptocurrency market capitalization, commodity prices (oil, gold, agriculture)

Energy and Resources: solar and wind power capacity, oil and gas production, nuclear energy share, water consumption per capita, mineral extraction volumes, waste recycling rates, renewable vs non-renewable energy mix

Technology and 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 and Society: pandemic case numbers, vaccination rollout rates, healthcare expenditure per capita, mental health survey scores, hospital bed availability, disease incidence rates, life satisfaction index

Retail and Sales: monthly retail sales by sector, online vs offline revenue, average basket size, foot traffic in malls, customer churn rate, loyalty program engagement

Education and Employment: enrollment rates in primary/secondary/tertiary, literacy rates, graduation rates by discipline, job vacancy data, unemployment rates, average salary by industry, remote work adoption, skill shortage indices

Tourism and Transportation: tourist arrivals by region, airline passenger miles, ride-sharing usage, public transit ridership, port container throughput, traffic congestion index, vehicle electrification adoption

Sports and Entertainment: sports league attendance, athlete medal counts, box office revenue by genre, music streaming hours, video game sales figures, award show winners stats, TV viewership ratings

Media and Communication: newspaper circulation, podcast listenership, YouTube subscriber growth, mobile app usage time, online news article shares, media trust index

Automotive and 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 and Food: crop yield per hectare, food price index, livestock population, organic farming acreage, seafood harvest volumes, global hunger index

Science and Research: scientific publication counts by field, research funding allocation, patent filings, RandD expenditure, Nobel prize distribution, clinical trial numbers

Real Estate and Construction: housing price index, construction starts by region, mortgage interest rates, commercial real estate vacancies, smart city projects

Government and Public Policy: budget deficit/surplus, tax revenue breakdown, public debt levels, policy approval ratings, crime rates by category, election turnout statistics

Space and Aeronautics: satellite launches, ISS research hours, space tourism bookings, Mars rover milestones, asteroid detection counts Miscellaneous: cryptocurrency price volatility, earthquake frequency and magnitude, festival attendance, book publication counts, open source project activity

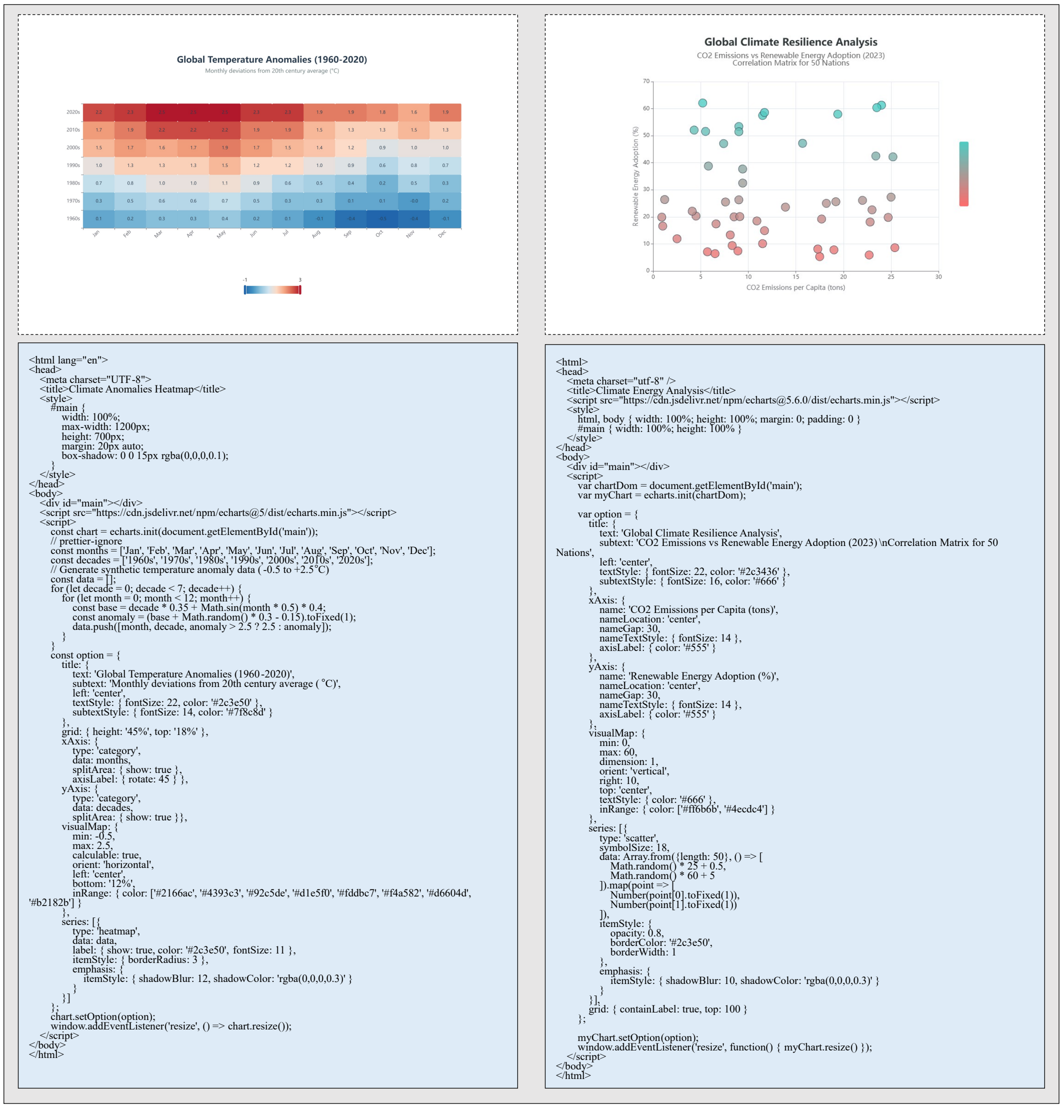
Use your imagination and knowledge to create different data distributions based on the topic.

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.

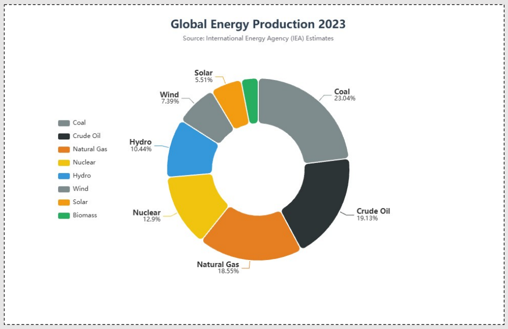


	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2020s	2.2	2.3	2.5	2.5	2.5	2.3	2.3	1.9	1.9	1.8	1.6	1.9
2010s	1.7	1.9	2.2	2.2	2.2	1.9	1.9	1.5	1.3	1.3	1.5	1.3
2000s	1.5	1.7	1.6	1.7	1.9	1.7	1.5	1.4	1.2	0.9	1.0	1.0
1990s	1.0	1.3	1.3	1.3	1.5	1.2	1.2	1.0	0.9	0.6	0.8	0.7
1980s	0.7	0.8	1.0	1.0	1.1	0.9	0.6	0.5	0.4	0.2	0.5	0.3
1970s	0.3	0.5	0.6	0.6	0.7	0.5	0.3	0.3	0.1	0.1	-0.0	0.2
1960s	0.1	0.2	0.3	0.3	0.4	0.2	0.1	-0.1	-0.4	-0.5	-0.4	-0.1

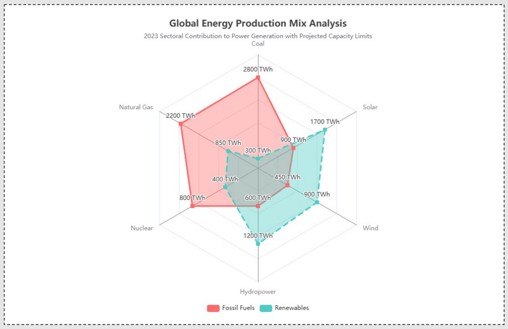
```
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Climate Anomalies Heatmap</title>
   <style>
      #main {
         width: 100%;
         max-width: 1200px;
         height: 700px;
         margin: 20px auto;
         box-shadow: 0 0 15px rgba(0,0,0,0.1);
      }
   </style>
</head>
<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 12: Example 1 from the Chart2Code Dataset.

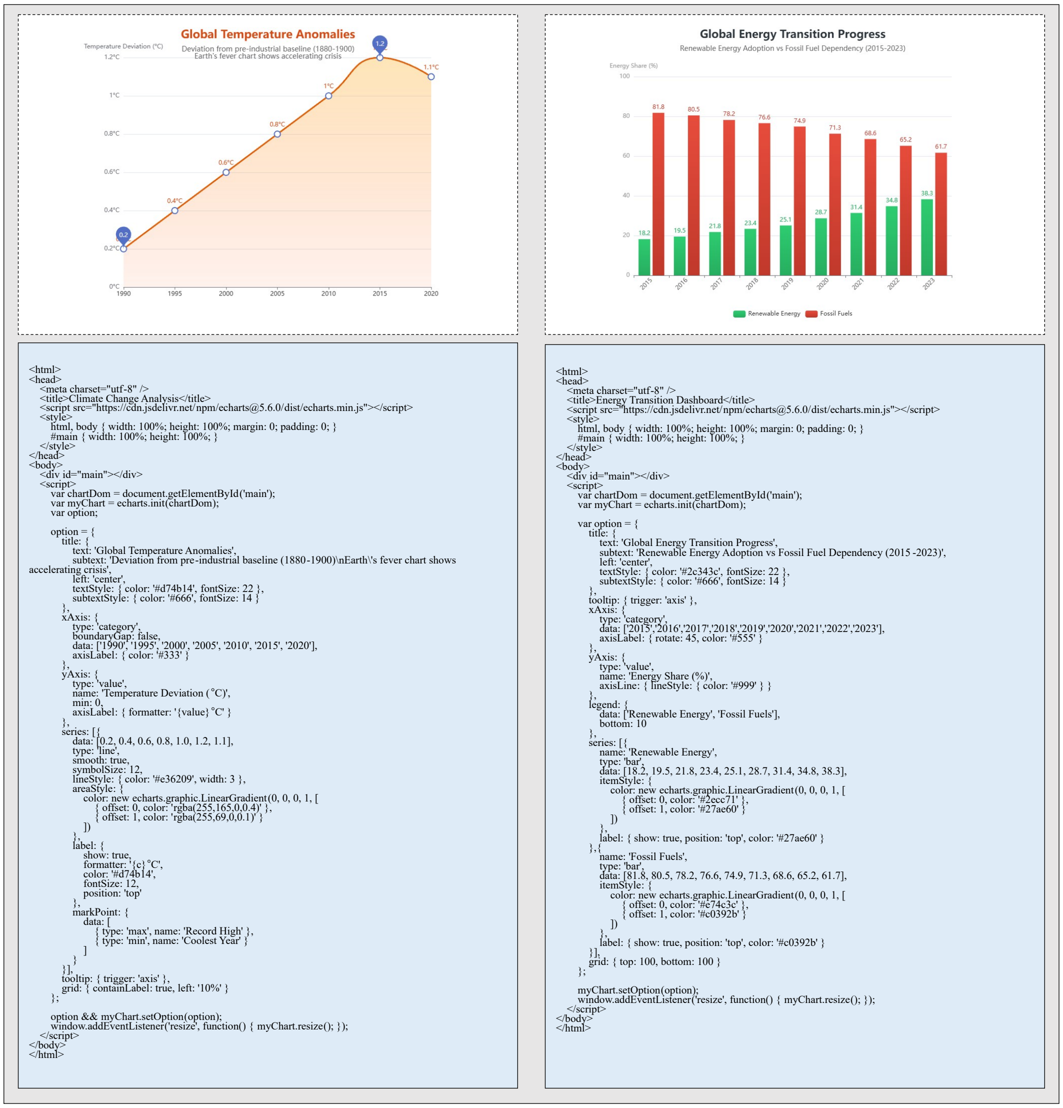


```
<html>
<head>
   <meta charset="utf-8" />
   <title>Energy Production Mix</title>
   <script src="https://cdn.jsdelivr.net/npm/echarts@5.6.0/dist/echarts.min.js"></script>
   <style>
      html, body { width: 100%; height: 100%; margin: 0; padding: 0; }
      #main { width: 100%; height: 100%; }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

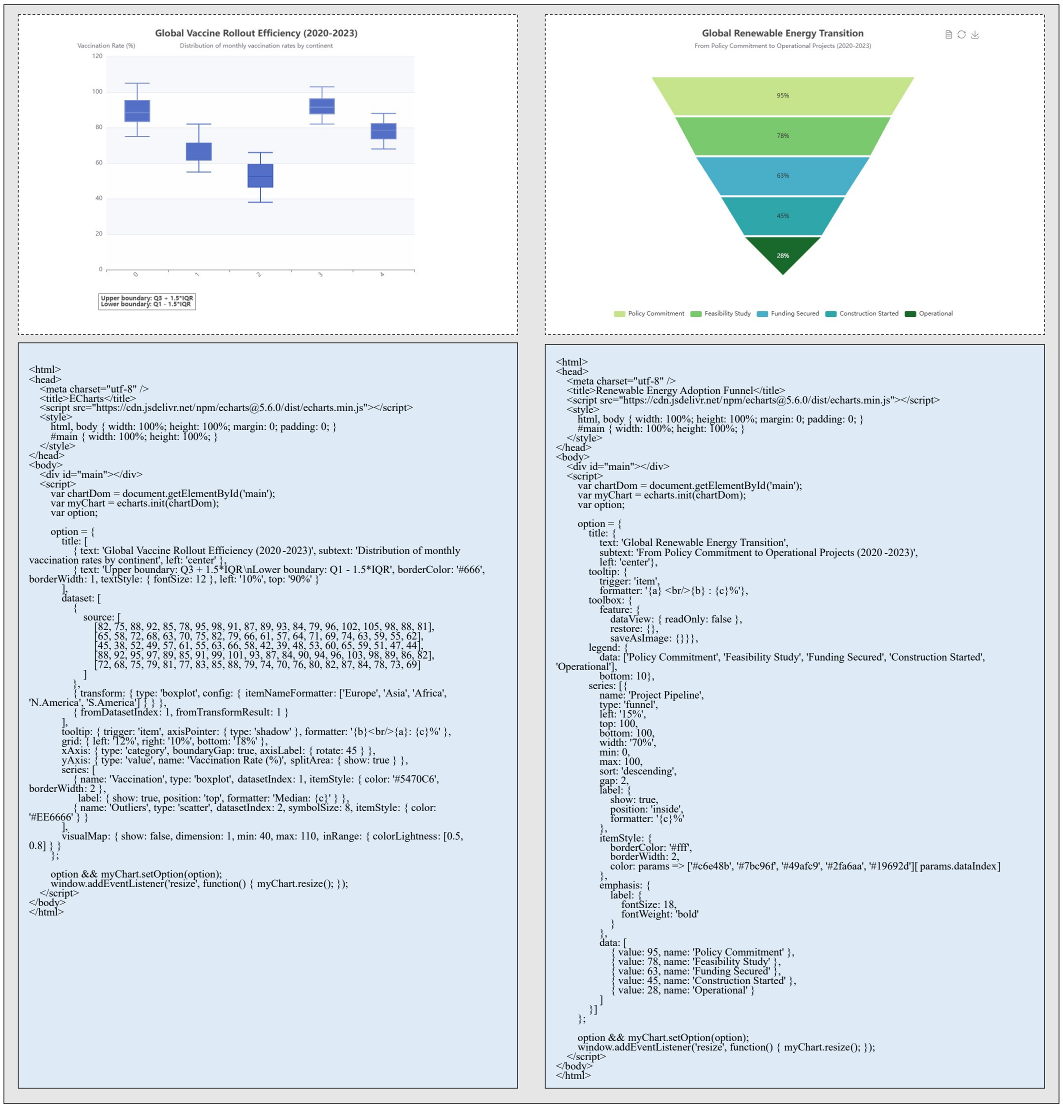
Figure 13: Example 2 from the Chart2Code Dataset.



```
<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 14: Example 3 from the Chart2Code 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 15: Example 4 from the Chart2Code Dataset.