Effective Training Data Synthesis for Improving MLLM Chart Understanding

Supplementary Material

A. Chart Types and Subplot Combinations

As illustrated in Fig. 1, our ECD dataset includes 29 distinct chart types for single plot, *i.e.,* "line", "bar", "pie", "area", "errorpoint", "treemap", "funnel", "node", "density", "histogram", "box", "bubble", "candlestick", "heatmap", "radar", "rose", "3d", "errorbar", "quiver", "scatter", "violin", "contour", "bar + line overlay", "pie + bar overlay", "histogram + density overlay", "violin + box overlay", "scatter + histogram overlay", "scatter + density overlay" and "hexbin + hist overlay".

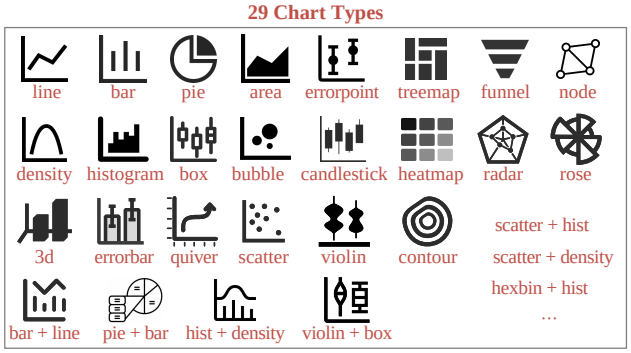

Figure 1. Distribution of Chart Types in our proposed effective chart dataset (ECD). Our dataset encompasses 29 diverse chart types, surpassing the variety present in existing datasets.

Additionally, for the combinations of subplots, our dataset only uses random combinations of 2 single-plot chart types (excluding overlays, can be the same type, *e.g.,* "line + line"). Following the pipeline outlined in this paper, it can be seamlessly extended to incorporate combinations of more chart types in practice, such as combinations involving 3 chart types, (*e.g.,* "line + bar + pie", etc.). Ultimately, 252 distinct combinations are successfully generated (excluding unsuccessful examples after filtering).

B. Chart Themes and Layout Distribution

As shown in Fig. 2, we present the distribution of chart themes and layouts in the proposed ECD dataset.

Chart Themes Distribution. Our dataset contains 25 scientific themes, with an average of approximately 421.4 images per theme. The number of images in each theme is as follows: Art and Design: 450, Agriculture: 453, Geography: 359, Medicine: 423, Engineering: 485, Law: 383, Biology: 451, Sports: 399, Mathematics: 495, Environmental Science: 450, Anthropology: 406, Sociology: 379, Computer Science: 474, Education: 380, Architecture: 405, Psychology: 436, Economics: 345, Statistics: 415, History: 332, Chemistry: 501, Media and Journalism: 383, Finance: 412, Physics: 500, Astronomy: 414, Linguistics: 405. Among them, the Chemistry theme has the highest count, while the History theme has the lowest.

Chart Layouts Distribution. We include 13 primary layouts, ranging from "1 by 1 (1, 1)" to "4 by 2 (4, 2)". The "1 by 1" layout represents a single chart image, with a total of 6,414 instances (accounting for 60.9%), while the other layouts represent multiple images, comprising 4,121 instances (accounting for 39.1%). This includes 631 instances of multiple images of the same chart type (*e.g.,* line + line subplots) and 3,490 instances of multiple images of different chart types (*e.g.,* line + bar subplots).

Models	ECDBench		
	Reasoning	Descriptive	Average
Claude Series			
Claude 3.5 Sonnet [2]	41.99	68.14	55.07
Claude 3.7 Sonnet [3]	43.38	**69.61**	56.50
Claude 4 Sonnet [4]	**44.20**	69.36	**56.78**
Gemini Series			
Gemini-2.5-Pro [6]	**44.36**	**76.88**	**60.62**
GPT Series			
Random (GPT-4o) [1]	4.58	1.63	3.10
GPT-4o-mini [1]	24.26	57.27	40.77
GPT-4o [1]	35.62	70.18	52.90
o1 [8]	40.52	74.18	57.35
o3 [14]	56.13	74.51	65.32
o4-mini [14]	**57.03**	**77.45**	**67.24**
Qwen-VL Series			
Qwen2.5-VL-7B [5]	19.04	57.35	38.19
Qwen2.5-VL-32B [5]	24.92	53.92	39.42
Qwen2.5-VL-72B [5]	**38.81**	**68.46**	**53.64**

Table 1. Performance comparison of several closed-source and open-source MLLMs on ECDBench. Among the listed models, o4-mini achieved the best performance.

C. ECDBench Statistics and More Results

The ECDBench we constructed comprises a total of 1,224 chart images, including 364 single-plot charts and 860 combined subplot charts. Among the combined-subplot charts, 457 feature combinations of two chart types (*e.g.,* "bar + line". with various layouts ranging from (1,2) to (3,3) / (4,2), etc.), while 403 involve combinations of three chart types (*e.g.,* "bar + pie + scatter". with (1,3) or (3,1) layout). The average size of images in ECDBench is 1378 × 968 px. ECDBench includes 2,448 QA pairs, with each image ac-

companied by 1 descriptive QA and 1 reasoning-based QA. For the manual checks and annotations of ECDBench, we employ PhD students who are familiar with common scientific charts.

Tab. 1 presents a more detailed comparison of evaluation results on several state-of-the-art (SOTA) proprietary and open-source MLLMs, including the Claude series, Gemini series, GPT series, and Qwen-VL series. Among all the evaluated MLLMs, o4-mini consistently achieves the highest performance across all three metrics (57.03% reasoning acc, 77.45% descriptive acc and 67.24% average acc).

D. FID and Avearage Pixel Entropy

In this paper, we employ the FID (Fréchet Inception Distance) and Average Pixel Entropy as metrics to assess the image realism and complexity of chart datasets respectively. Here, we present the specific calculation formula.

Fréchet Inception Distance (FID): For the real dataset I_{real} and Synthetic dataset I_{syn}:

$$\text{FID}(I_{\text{real}}, I_{\text{syn}}) = \|\mu_{\text{real}} - \mu_{\text{syn}}\|^2 + \text{Tr}\left(\Sigma_{\text{real}} + \Sigma_{\text{syn}} - 2\sqrt{\Sigma_{\text{real}}\Sigma_{\text{syn}}}\right), \tag{1}$$

where μ_{real} and μ_{syn} are the mean feature vectors of the Inception network embeddings, Σ_{real} and Σ_{syn} are their covariance matrices, and $\text{Tr}(\cdot)$ denotes the matrix trace operator.

Average Pixel Entropy: The average pixel entropy can be defined as:

$$H(x) = -\sum_{i=1}^{n} p(i) \log p(i), \tag{2}$$

$$\text{AvgEnt}(I) = \frac{1}{N} \sum_{k=1}^{N} H(x_k), \tag{3}$$

where $p(i)$ is the normalized histogram probability of intensity value i, n is the number of discrete intensity levels (*e.g.,* 256 for 8-bit images), $H(x_k)$ is entropy of the k-th image.

E. Computational and GPT-4o Cost

Dataset	#Images	#QA Pairs	$Per-image	$Per-QA	$Total
CharXiv	2.3k	5k	0.37	0.170	∼0.85k
ReachQA	3.2k	19.9k	0.10	0.016	∼0.31k
ECD	10.5k	321.5k	0.20	0.007	∼2.15k

Table 2. Cost ($) comparison between ECD and existing datasets.

Generating 1,000 single-plot images takes an average of 7.1 seconds per image and 22.40 USD in total. On the other hand, generating 1,000 combined subplots costs 22.6 seconds per image and 85.12 USD in total. During diversification, each set of 1,000 images costs 30.14 USD and requires an average of 40.18 seconds per image. The figure size post-processing and the image filtering costs 27.90 USD per 1,000 images in total. Additionally, generating QA pairs for 1,000 chart images costs 38.97 USD. In total, generating ECD costs ∼2,145 USD. We add cost comparison in Tab. 2. While ECD has the highest total cost due to its large number of image and QA pairs, its per QA cost is the lowest, and its per-image cost is medium (but not rediculously high on 0.2$ per image), because it has many multi-subplot charts, which is essential in its design. We also tried replacing GPT-4o with Qwen2.5-VL-32B for training data generation. Results are shown in Tab. 3. We find that both yield improvement on ReachQA benchmark, but the dataset generated by Qwen2.5-VL-32B does not improve performance on CharXiv.

Models	CharXiv	ReachQA
LLaVA-Next-Llama3-8B	35.06	15.65
+ ECD by Qwen2.5-VL-32B	34.90	21.40
+ ECD by GPT-4o	45.66	18.85

Table 3. Comparison on ECD generation (5k images, 30k QAs).

F. Cross-task MLLM SFT Impact

To assess the impact of our synthetic chart QA data on model generalization, we fine-tune LLaVA-Next-Llama3-8B on our dataset and evaluate on MathVista [10] (math images, testmini split), MMBench [9] (mixed natural and scientific images, dev-en split), and RealworldQA [16] (natural images, test split). In Tab. 4, our model leads to slight improvements on MathVista and RealworldQA and a decrease on MMBench. These early results might be inconclusive, and we plan to expand them in the future.

Models	Mathvista [10]	MMBench [9]	RealWorldQA [16]
LLaVA-Next-Llama3-8B	37.00	**79.03**	58.95
+ ECD	**38.40**	77.36	**59.22**

Table 4. GPT-Acc (%) on 3 common vision-language benchmarks.

G. Analysis of the Generation Pipeline

We construct several variants to demonstrate the effectiveness of the proposed pipeline.

Separate data and code generation vs. joint generation is shown in Tab. 5 (a) vs Tab. 5 (c). We separately generate the code and data, comparing it with the case where the code and data are generated together. We observe that our method is superior to generating data and code jointly, evidenced by the lower FID and higher entropy.

Effectiveness of conditional and sequential data generation for multi-subplot charts. We condition each sub-

Generation Strategy	FID↓	Avg. Entropy↑
(a) Code and Data Joint Gen.	126.86 ± 3.11	1.91
(b) Subplots Parallel Gen.	123.26 ± 2.15	1.90
(c) Ours	**120.63 ± 3.23**	**2.05**

Table 5. Comparison of three chart generation strategies. (a) Generating chart code and data simultaneously. (b) Generate all the subplots data simultaneously. (c) Our data generation method (decoupling code and data; conditional generation of subplots). We define 12 different layouts for the common "bar+line" combination, and generate 10 images for each layout. We use FID (with CharXiv) [15] and average pixel entropy to evaluate the realism and complexity of generated chart images.

plot on previous ones to maintain consistency. We compare this method with joint multi-subplot generation in Tab. 5 (b) vs Tab. 5 (c). We observe a decline in the realism and complexity if we do joint generation, which indicates the effectiveness of our design.

H. Numerical Diversity, Chart Style Statistics and Multi-Subfigure Reasoning Analysis

Numerical Diversity. Our current method has three strategies for numerical diversity. 1) We randomly sample data trend, *e.g.*, increasing, decreasing, or stable. 2) we randomly sample the number of elements, *e.g.*, the number of groups in a bar plot. 3) We use a high temperature (1.0) in GPT-4o to encourage varied parameter output.

Chart Style Statistics. We calculate the percentage of charts undergoing various changes using our synthesis method. For line-charts we find: 100.00% charts have changes in line color, 99.65% in font size, 95.09% in line width, 67.37% with non-default background, 59.30% with transparency, 47.37% in grid, 39.65% in new annotations, 29.12% in area shading, 17.19% include arrows, 11.23% draw threshold lines, 10.18% remove axis borders, 10.18% add zoom-in insets and 3.51% include error bars. In total, 58.95% of line charts include 6 or more distinct changes.

Multi-Subfigure Reasoning QA Pairs. We analyze 500 reasoning QA pairs. ∼13.4% are identified by GPT-4o as requiring multi-subfigure reasoning. Moreover, while some QAs can be answered by a single figure in a multi-subfigure chart, such QAs are still difficult because of the need for subfigure localization.

I. Discussions and Limitations

What makes a high-quality chart training set? This paper mainly uses FID and average entropy as indicators to the similarity to real images and image complexity. However, the fact that ECD performs better than other datasets does not mean that ECD always has a lower FID and higher average entropy. We validated that the number of themes and chart types are also useful. Other factors such as text formats, font sizes, and even colors may also influence the training set quality.

Lack of a dedicated vision encoder for chart. FID calculation requires feature extraction using InceptionNet-V3, pretrained on ImageNet (a natural image dataset). There are domain gaps between natural images and chart images, which might limit the efficacy of FID. The community still lacks a proper chart vision encoder to better compute indicators that require feature extraction.

J. Comparison with Other Chart Datasets

To facilitate further comparison with other datasets, Fig. 3 present qualitative comparisons. This figure presents a visual comparison between our dataset's chart images and those from several chart QA training datasets, including PlotQA [13], ChartQA [11], ChartBench [18], SimChart9k [17], ChartAssistant [12], and ReachQA [7]. Compared to these datasets, ours features a greater variety of chart types, more diverse combinations, more intricate details, and richer visual representations.

K. Prompt for Chart Image Generation

Fig. 4, Fig. 5, Fig. 6 and Fig. 7 illustrate the prompts we use to guide GPT-4o in generating single plot chart, combined subplot charts, and diverse visual variations. I) *Single plot chart*: We employ predefined chart functions and examples to prompt GPT-4o to generate the data for a single plot chart. II) *Combined subplot chart*: each single plot in the combined chart is generated sequentially based on conditional inputs (*i.e.*, data of previous generated plots), ensuring semantic continuity across the entire chart composition. III) *Diversification*: We define 5 and 6 diversification strategies for the single-plot chart and the combined subplot chart, respectively, and randomly select 1 strategy for each chart generation. It is important to note that additional prompts are incorporated to guide GPT-4o in prioritizing detail modifications while ensuring the preservation of the originally generated data. Fig. 8 presents the post-processing prompt applied to chart images, designed to empower GPT-4o in refining figure size and resolution. This step aims to enhance the conveyance of visual information, particularly in improving the clarity of elements such as text.

L. Prompt for Chart Image Rating

Fig. 9 and Fig. 10 illustrate the prompts utilized for chart image evaluation, incorporating two distinct rating criteria: "visual clarity" and "semantic coherence". The first criterion is employed to assess the visual integrity of the chart (on a scale from 1 to 5), specifically targeting the exclusion

of images exhibiting excessive blank spaces or chaotic overlaps. The second criterion evaluates the semantic coherence (also on a scale of 1 to 5), ensuring that the chart adheres to a well-defined and logically structured theme. The integration of these two metrics serves to filter out charts of insufficient quality.

M. Prompt for Chart QA Generation

Fig. 11 and Fig. 12 illustrate the prompts we provide to GPT-4o for generating question-answer (QA) pairs based on chart code, where chart images are also incorporated as input to assist in determining whether the questions can be answered solely based on visual information. The generated QA pairs are categorized into two types: descriptive and reasoning. Descriptive QA pairs focus on the identification of fundamental textual, numerical, and graphical elements within the chart. Reasoning QA pairs, on the other hand, emphasize hierarchical analysis, computational inference, and logical deduction, with the generated answers incorporating rationale as part of the reasoning process. Additionally, for each generated QA pair, GPT-4o is tasked with producing a confidence rating, which serves as a measure for subsequent filtering.

N. Prompt for Evaluation on ECDBench

Fig. 13 illustrates the prompt used for evaluating model predictions on ECDBench. Similarly, we employ the LLM-as-Judge approach to assess the responses. For answers containing numerical results, a 5% tolerance is permitted to account for visual recognition discrepancies.

References

[1] 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. *arXiv preprint arXiv:2303.08774*, 2023. 1

[2] Anthropic. Introducing claude 3.5 sonnet. https://www.anthropic.com/news/claude-3-5-sonnet, 2024. 1

[3] Anthropic. Claude 3.7 sonnet and claude code. https://www.anthropic.com/news/claude-3-7-sonnet, 2025. 1

[4] Anthropic. Introducing claude 4. https://www.anthropic.com/news/claude-4, 2025. 1

[5] 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. *arXiv preprint arXiv:2502.13923*, 2025. 1

[6] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025. 1

[7] Wei He, Zhiheng Xi, Wanxu Zhao, Xiaoran Fan, Yiwen Ding, Zifei Shan, Tao Gui, Qi Zhang, and Xuanjing Huang. Distill visual chart reasoning ability from llms to mllms. *arXiv preprint arXiv:2410.18798*, 2024. 3, 6

[8] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024. 1

[9] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision (ECCV)*, pages 216–233. Springer, 2024. 2

[10] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*, 2023. 2

[11] 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 Association for Computational Linguistics: ACL 2022*, pages 2263–2279, 2022. 3, 6

[12] 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. *arXiv preprint arXiv:2401.02384*, 2024. 3, 6

[13] Nitesh Methani, Pritha Ganguly, Mitesh M Khapra, and Pratyush Kumar. Plotqa: Reasoning over scientific plots. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1527–1536, 2020. 3, 6

[14] OpenAI. Openai o3 and o4-mini system card. https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf, 2025. 1

[15] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016. 3

[16] xAI Organization. Realworldqa: A benchmark for real-world understanding. https://huggingface.co/datasets/xai-org/RealworldQA, 2025. 2

[17] Renqiu Xia, Bo Zhang, Haoyang Peng, Hancheng Ye, Xiangchao Yan, Peng Ye, Botian Shi, Yu Qiao, and Junchi Yan. Structchart: Perception, structuring, reasoning for visual chart understanding. *arXiv preprint arXiv:2309.11268*, 2023. 3, 6

[18] Zhengzhuo Xu, Sinan Du, Yiyan Qi, Chengjin Xu, Chun Yuan, and Jian Guo. Chartbench: A benchmark for complex visual reasoning in charts. *arXiv preprint arXiv:2312.15915*, 2023. 3, 6

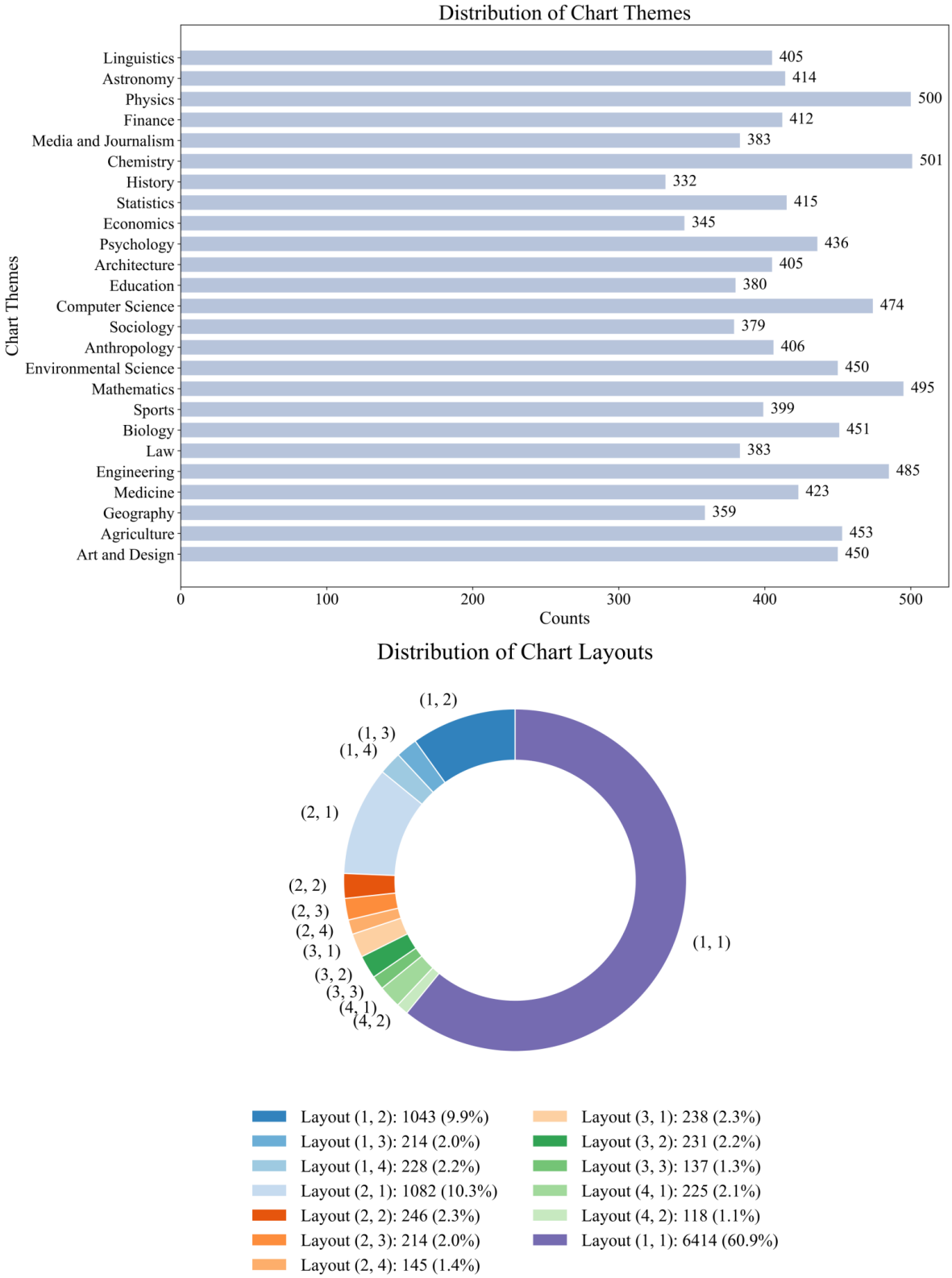

Figure 2. Distribution of themes and layouts in our proposed effective chart dataset (ECD). Our dataset comprises 25 distinct themes and 12 multi-plot layouts. It includes 6,414 single-chart images and 4,121 combined subplot images. Among the multi-plot layouts, 631 subplots consist of combinations of the same chart type, while 3,490 subplots involve combinations of different chart types. Following the pipeline proposed in this paper, our dataset can be further expanded to include more chart types and layouts.

Figure 3. Compared to the chart images in existing chart QA training datasets (sampled for better visualization, *i.e.*, PlotQA [13] (3 types), ChartQA [11] (3 types), ChartBench [18] (9 types), SimChart9k [17] (3 types), ChartAssistant [12] (9 types), ReachQA [7] (10 types)), it is evident that our ECD (29 types) exhibits a greater diversity of subplot combinations (>250 types of combination) and complexity.

Prompt for Single Plot Generation

Assistant:

Intelligent Assistant Prompt: Input Synthesis Helper

You are a "Function Input Synthesis Helper," and your task is to generate an appropriate function call based on a provided function, filling the call with reasonable parameters.

User-Provided Function:

```python
{user_function}
```

Synthesized Function Call Example:

```python
{function_name}(
    {parameters_and_values}
)
```

Important Considerations:

--Ensure that data_list contains realistic data with variability and noise.
--The pattern for parameters should reflect potential trends, cycles, randomness, and correlations.
--Data should exhibit complexity and noise, avoiding simple linear or polynomial.
--Make sure the elements in data_list match labels.
--Parameters such as colors_list, linestyles_list, and markers_list should match the structure and amount of data in data_list.
--Each parameter should follow a unique potential pattern.
--Users may provide data themes and constraints on certain parameters, and data should be generated accordingly.
--Data values should make sense and fall within appropriate ranges for the given context.
--Labels should be specific and meaningful, avoiding generic terms like "Product One" or "Product Two." The names and chart labels should be specific to the data and context.
--Chart should be clear and visually appealing, with appropriate colors and styles for each line chart. linewidths should larger than 1 for better visibility.
--Adjust the width of any elements (e.g., bars, lines) to ensure they are visible and distinguishable.
--Generate **random and diverse** colors/styles/markers, and don't start with **a fixed color/style/marker**. Explore elegant and unique color tones suitable for research papers, while also considering commonality.
--You MUST add noise to the data to make it more realistic and challenging.
--Do NOT generate linear data.
--DO NOT return function code, just the function call with parameters.

User:

{plot_function}
Example Usage:
{example_usage}
Chart Theme: Imagine a real application scenario under the **{random_theme}** theme and generate the data accordingly. Be specific. For example, if the theme is Finance, you could imagine a company struggling financially but expanding into new business areas. If the theme is Astronomy, you could picture a star nearing the end of its life.
The number of elements:
First level: {random.randint(3, 6)};
If second level make sense in the context, Second level: {random.randint(3, 6)}.
Trend of data: create {trend_string}.
grid: {random.choice([False, True])}.
save_path: {png_save_path}.

Theme Collections:
["Economics", "Psychology", "Sociology", "Biology", "Education", "Engineering", "Law", "Astronomy", "Computer_Science", "Geography", "Physics", "Chemistry", "History", "Environmental_Science", "Anthropology", "Media_and_Journalism", "Mathematics", "Statistics", "Finance", "Medicine", "Art_and_Design", "Agriculture", "Linguistics", "Architecture", "Sports"]

Figure 4. Prompt template for generating a single plot. We define 25 academic themes for generating charts across diverse disciplines.

Prompt for Single Plot Diversification

Assistant 1：

Intelligent Assistant Prompt: Single Plot Code Synthesis and Diversification Helper

You are a 'Single `Plot` Code Synthesis and Diversification Assistant.' Your task is to synthesize an appropriate single `plot` code based on the predefined chart function and calling example provided by the user, ensuring that it follows the user's diversification strategy to make the synthesized code sufficiently varied.

User-Provided Plot Function:
```python
{single_plot_function}
```

User-Provided Calling Example:
```python
{calling_example}
```

Synthesized Code Example:
```python
{code_body}
```

Assistant 2：

Intelligent Assistant Prompt: Code Debug Helper

You are a 'Code Debug Assistant.' Your task is to identify and fix issues in the user's code based on any provided errors, ensuring it works correctly.

User-Provided Code:
```python
{user_code}
```

Error Message:
{error_message}

Returned Code:
```python
{whole_code_after_fix}
```

User:
{single_plot_function}
Provided calling example for single plot: {calling_example}. The data/labels/legends in the chart must be preserved at the diversification stage.
Diversification Strategy for Single Plot Chart Code: {random.choice{Diversification_Strategy_Collections}}.
save_path: {div_png_save_path}.
Important Considerations:
--Code Diversity: Provide multiple styles and techniques for modifying the chart, using different libraries such as Seaborn to enhance the code and visual presentation. Must avoid returning results as **functions like 'def' or function calls**, and instead, provide complete executable code.
--Layout and Organization: You must follow the **save_path and diversification strategy** and ensure clear separation (without any overlap), alignment and labeling for readability. Additionally, feel free to adjust the figsize to ensure that all elements are fully visible and can be displayed.
--Original Data Preservation: You are required to create the code, without modifying or ignoring the original chart data. It is worth noting that for the line_num and bar_num charts, the number annotations should not be modified or removed.
Note: Be sure not to use any interactive elements that cannot be saved. If chart type is Funnel, use plotly lib for implementation.

Diversification_Strategy_Collections:
["Add text labels, annotations, arrows, uncertainty bars, threshold lines (which should have specific names, not just 'threshold'; these could include curves like exponential/logarithmic curves, and not always be straight lines), or highlights (e.g., using a circle or highlighting a range in a specific color) to emphasize key data points, trends, or regions, ensuring that these annotations are contextually relevant and not generic. The number of items added can exceed one.",
"Modify the font styles, colors or sizes significantly for titles, labels or ticks.",
"Use gradient fills, area shading (typically along the line itself, within a defined range above and below it, must not between the line and the x-axis), or transparency effects to enhance depth. Additionally, fine-tune grid lines, background colors, and shadow effects to improve visual appeal.",
"(If Applicable) Remove axis borders for a cleaner, modern look.",
"(If Applicable) Incorporate a **zoomed-in inset** of a particular section, ensuring the area is appropriately sized (usually a very small size) and placed, making sure that the elements are visually separated without overlapping."]

Figure 5. Prompt template for single plot diversification. We define 5 diversification strategies and randomly select one for each plot.

Prompt for Combined Subplots Generation and Diversification – (a)

Assistant 1:

Intelligent Assistant Prompt: Input Synthesis Helper

You are a "Function Input Synthesis Helper," and your task is to generate an appropriate function call based on a provided function, filling the call with reasonable parameters.

User-Provided Function:
```python
{user_function}
```

Synthesized Function Call Example:
```python
{function_name}(
    {parameters_and_values}
)
```

Important Considerations:
--Ensure that data_list contains realistic data with variability and noise.
--The pattern for parameters should reflect potential trends, cycles, randomness, and correlations.
--Data should exhibit complexity and noise, avoiding simple linear or polynomial.
--Make sure the elements in data_list match labels.
--Parameters such as colors_list, linestyles_list, and markers_list should match the structure and amount of data in data_list.
--Each parameter should follow a unique potential pattern.
--Users may provide data themes and constraints on certain parameters, and data should be generated accordingly.
--Data values should make sense and fall within appropriate ranges for the given context.
--Labels should be specific and meaningful, avoiding generic terms like "Product One" or "Product Two." The names and chart labels should be specific to the data and context.
--Chart should be clear and visually appealing, with appropriate colors and styles for each line chart. linewidths should larger than 1 for better visibility.
--Adjust the width of any elements (e.g., bars, lines) to ensure they are visible and distinguishable.
--Generate **random and diverse** colors/styles/markers, and don't start with **a fixed color/style/marker**. Explore elegant and unique color tones suitable for research papers, while also considering commonality.
--You MUST add noise to the data to make it more realistic and challenging.
--DO NOT generate linear data.
--DO NOT return function code, just the function call with parameters.

Assistant 2:

Intelligent Assistant Prompt: Combined Subplot Chart Code Synthesis Helper

You are a 'Combined Subplot Chart Code Synthesis Assistant.' Your task is to synthesize an appropriate combined subplot chart code based on the individual chart functions and calling examples provided by the user, ensuring that all functions and examples integrate seamlessly.

User-Provided Functions:
```python
{user_function}
```

User-Provided Calling Examples:
```python
{user_calling_examples}
```

Synthesized Code Example:
```python
{code_body}
```

Assistant 3:

Intelligent Assistant Prompt: Code Debug Helper

You are a 'Code Debug Assistant.' Your task is to identify and fix issues in the user's code based on any provided errors, ensuring it works correctly.

User-Provided Code:
```python
{user_code}
```

Error Message:
{error_message}

Returned Code:
```python
{whole_code_after_fix}
```

Figure 6. Prompt template for combined subplots generation and diversification - Part (a).

Prompt for Combined Subplots Generation and Diversification – (b)

==========Stage I：Synthesize the data of all individual Subplots==========

User:
{current_plot_function}
Example Usage of single chart: {current_example}.
Subplot Data generated previously: {before_data}.
Color Scheme: Ensure that the color scheme across subplots is consistent, balancing both aesthetic appeal and clarity in data presentation.
Chart Theme: Imagine a real application scenario under the **{random_theme}** theme and generate the data accordingly. Be specific. For example, if the theme is Finance, you could imagine a company struggling financially but expanding into new business areas. If the theme is Astronomy, you could picture a star nearing the end of its life.
The number of elements:
First level: {random.randint(3, 6)};
If second level make sense in the context, Second level: {random.randint(3, 6)}.
Trend of data: create {trend_string}.
grid: {random.choice([False, True])}.
save_path: {png_save_path}.
Important Considerations: You must generate parallel sub-graphs that maintain strong internal cohesion while introducing meaningful variations, ensuring both similarity and distinctiveness within the data structure.

==========Stage II：Combine Subplots and Diversification==========

User:
{all_plot_functions}
Provided chart data for each subplot: {combined_data}. The data/labels/legends in the chart must be preserved at the combining stage.
Layout: A {random_layout[0]}x{random_layout[1]} grid of subplots, totaling {random_layout[0] * random_layout[1]} subplots. You must ensure the elements in the data_list match the number of subplots in the layout.
save_path: {combined_png_save_path}.
Diversification Strategy for Combined Subplots Chart Code: {random.choice{Diversification_Strategy_Collections}}.
Important Considerations:
--Code Diversity: Provide multiple styles and techniques for modifying the chart, using different libraries such as Seaborn to enhance the code and visual presentation. Must avoid returning results as **functions like 'def' or function calls,** and instead, provide complete executable code.
--Layout and Organization: You must follow the **requested layout, save_path and change strategy** and ensure clear separation (without any overlap), alignment and labeling for readability. Additionally, feel free to adjust the figsize to ensure that all elements are fully visible and can be displayed.
--Original Data Preservation: You are required to create the code, without modifying or ignoring the original chart data. It is worth noting that for the line_num and bar_num charts, the number annotations should not be modified or removed.
Note: Be sure not to use any interactive elements that cannot be saved. Besides, {random.choice{Subtitles_Strategy_Collections}}.

Diversification_Strategy_Collections:
["Add the title of the whole combined chart in a specific and meaningful way, avoiding general terms.",
"Add text labels, annotations, arrows, uncertainty bars, threshold lines (which should have specific names, not just 'threshold'; these could include curves like exponential/logarithmic curves, and not always be straight lines), or highlights (e.g., using a circle or highlighting a range in a specific color) to emphasize key data points, trends, or regions, ensuring that these annotations are contextually relevant and not generic. The number of items added can exceed one.",
"Modify the font styles, colors or sizes significantly for titles, labels or ticks.",
"Use gradient fills, area shading (typically along the line itself, within a defined range above and below it, must not between the line and the x-axis), or transparency effects to enhance depth. Additionally, fine-tune grid lines, background colors, and shadow effects to improve visual appeal.",
"(If Applicable) Remove axis borders for a cleaner, modern look.",
"(If Applicable) Incorporate a **zoomed-in inset** of a particular section, ensuring the area is appropriately sized (usually a very small size) and placed, making sure that the elements are visually separated without overlapping."]

Subtitles_Strategy_Collections:
["leave the subfigure titles as they are without any letters",
"retain the original subfigure titles and add letters before the titles, such as (a), (b), (c), (d)",
"choose to remove the original subfigure titles and use only the letters"]

Figure 7. Prompt template for combined subplots generation and diversification - Part (b). The combined subplots introduce an additional diversification strategy on top of the single plot (i.e., add the title of the whole chart). Additionally, the diversification of subtitles is also randomly selected from the collection.

Prompt for Figure Size Post-Processing

You are an experienced Python developer specializing in data visualization and chart rendering. The following Python code generates and saves charts. Some charts may have unnecessarily large or inappropriate sizes, while others may already be optimal. Modify the code as follows:

1. If the chart size is inappropriate (e.g., unnecessarily large or excessively small), adjust the figsize to a **reasonable resolution** suitable for typical use cases.
2. If the chart size is already appropriate, leave it unchanged.
3. Saves with dpi **no higher than 150** (e.g., `dpi=100`).
4. Guarantee that axis labels, titles, legends and tick labels **do not overlap**:
   - Use `fig.tight_layout()` or `constrained_layout=True`.
   - Dynamically scale `figsize` or reduce `plt.rcParams["font.size"]` when needed.
   - Verify that every text element remains readable and non-overlapping.
6. Do not alter the chart content, style, or any other functionality of the code.
7. Headless: set backend "Agg" before pyplot and remove all **.show()** calls.
8. Ensure the modified code is efficient, readable, and adheres to best practices for saving visualization outputs.

Here is the original code:
```python
{original_code}
```
Respond with the complete modified code, enclosed in Python code block tags for clarity.

Figure 8. Prompt template for figure size post-processing. We enable GPT-4o to dynamically adjust figure size and visual elements during post-processing, ensuring better clarity and visualization.

Prompt for Visual Clarity Rating

<Chart_Image>

You're an expert evaluating the **visual clarity** of a chart. Assign a rating from 1 to 5, adhering to the detailed descriptions below:

1 point: The chart is essentially unreadable due to overwhelming issues, such as excessive clutter, severe text overlap, missing legends, or large blank areas, making it impossible to interpret the data accurately.
2 points: The chart has significant design flaws that hinder understanding. Issues like noticeable clutter, text overlap, or missing elements complicate the data's presentation and make it difficult to extract meaningful insights.
3 points: While the chart communicates the data to some extent, it suffers from moderate issues. Mild clutter, misleading scales, or slight text overlap may confuse viewers, affecting overall clarity.
4 points: The chart is largely clear and conveys the data effectively, though minor issues like slight text overlap, small misalignments, or minor clutter might cause slight distractions without greatly affecting comprehension.
5 points: The chart is visually excellent, with precise and well-organized data presentation. It uses space effectively, ensuring that the information is easy to interpret and completely free of any errors.

Layout for current chart: **{layout}**, you should consider whether the chart image matches the provided layout. Give your rating for the provided chart image now.

Provide your rating in the format without any explanation:
Rating: (integer value within 1 to 5)

Figure 9. Prompt template for visual clarity rating. We apply a 1∼5 point scale for GPT-4o to evaluate and select chart images with excellent visual clarity. The underlined portions are applicable only to combined subplot charts with layout that are not 1 by 1.

Prompt for Semantic Coherence Rating

<Chart_Image>

You're an expert evaluating the **semantic coherence** of a chart. Assign a rating from 1 to 5, adhering to the detailed descriptions below:

1 point: The chart is irrelevant to its intended message or theme. The data is disjointed or disconnected, with no clear connection between elements or subplots. There is no recognizable story, and the chart does not support the overall narrative in any meaningful way.

2 points: The chart's content partially aligns with its theme, but there are significant gaps in how the data is presented. The relationships between elements are weak or unclear, and subplots may seem out of place, making it difficult to extract a clear narrative or insight from the chart.

3 points: The chart generally supports its theme or message, but the connections between data points or subplots are inconsistent or ambiguous. The overall story is somewhat discernible, but there are sections where the data feels disjointed or hard to follow.

4 points: The chart aligns well with its theme, presenting a mostly coherent story. There are clear connections between the main plot and subplots, though some minor ambiguities or distractions might slightly affect the clarity of the narrative. The data supports the theme effectively.

5 points: The chart is highly coherent, with a clear, logical progression of ideas. The data is directly relevant to the theme, and the relationships between the main plot and any subplots are seamless. The chart presents a compelling and easy-to-follow narrative, making the insights immediately apparent.

Theme for current chart: **{theme}**, you should consider whether the chart image matches the provided theme. Give your rating for the provided chart image now.

Provide your rating in the format without any explanation:
Rating: (integer value within 1 to 5)

Figure 10. Prompt template for semantic coherence rating. We apply a 1∼5 point scale for GPT-4o to evaluate and select chart images with clear, logical semantic coherence.

Prompt for Descriptive QA Pairs Generation

User：
You are a careful and precise observer of the chart. You are provided with the original chart code as follows:

```python
{chart_code}
```

Your task is to carefully analyze the chart code details and generate **Basic Descriptive-based** question-answer pairs. Additionally, a chart image is provided to assist in verifying whether the questions can be fully addressed based on the visual information it contains.

Areas to Focus On:
1. **Textual Content**: This includes elements like titles, subtitles, axis label (e.g., x-axis or y-axis), tick marks (e.g., labeled ticks at extremes, like **the leftmost/rightmost/highest/lowest**, etc.), annotations, or legends (e.g., list of legend labels).

2. **Numerical Content**: This refers to the count of variables (e.g., lines/bars/instances), the number of ticks across all axes, **differences between numerical tick values on the x- and y-axes**, label count in legends, and the maximum/minimum (or **difference between the maximum and minimum**) and intervals in continuous colorbar, etc.

3. **Graphical Content**: This includes visual characteristics such as:
   - Chart type
   - Point positions
   - Line intersections and thresholds
   - Overall orientation (e.g, "veritical" or "horizontal", not the angle of specific elements)
   - **Subplot layouts** (e.g., answer in format like "1 by 2", "2 by 4", "3 by 3")
   - Number of subplots
   - Highlights and zoom-in inset (e.g., emphasizing specific features for clarity)
   - General trends and patterns (e.g., "increase," "decrease," "stabilize")

Response Format:
Please use the following JSON format to ensure clarity:
```json
{
   {
     "image_id": "{img_id}_1",
      "rating": "Rate your confidence in this question and answer [0-5], Integer",
      "question": "Provide your question here",
      "answer": "Provide your corresponding complete answer here"
    },
    {
      "image_id": "{img_id}_2",
        ...
     },
        …
}
```

Notes:
1. Focus only on aspects that can be inferred visually from the chart image. Do not refer to facts or terms that directly suggest the content is based on the code rather than the chart image itself.
2. The total number of questions should be **10 to 15,** and feel free to rephrase your questions to enhance diversity.
3. The QA pairs should include questions involving the enumeration and counting of elements such as legends, ticks (e.g., total across all axes, like x_axis + y_axis), variables, etc. Furthermore, simple calculations are necessary for identifying differences.
4. Do not include questions about **alpha value, transparency, opacity, padding, spacing, grid, width, viewing angle, rotation, function (e.g.,'np.random'), font size, figsize, save filename and save_path** in the code. Focus only on the meaningful visual elements in the chart and avoid mentioning 'code' in the questions and answers.
5. Ensure the image_id numbers increment sequentially, such as _1, _2, _3, _4, and so on. Keep the ```json``` formatting intact.

Figure 11. Prompt template for descriptive QA pairs generation. We prompt GPT-4o to focus on various aspects of descriptive queries, such as textual content, numerical content, and graphical content, and generate multiple question-answer pairs based on these dimensions.

Prompt for Reasoning QA Pairs Generation

User：
As an expert in chart mathematical analysis, you are provided with the original chart code as follows:

```python
{chart_code}
```

You are tasked with examining a given chart code and generating a series of in-depth **Analytical Reasoning-based** question-answer pairs. Additionally, you have been given the chart image. The chart image will help you verify if the questions can be fully answered from it.

Generation Requirements:
1. Generate a diverse set of questions based on the chart data. The questions should cover a variety of aspects, such as comparisons, ratios, rankings, trends, range, overlaps, extreme values, relationships, highlights, annotations, shading area or calculations. You can freely combine any of above aspects to create a challenging question.

2. Ensure that the phrasing of the questions is varied and complex. Avoid using the same simple question structure or repetitive wording. Use a variety of question types, including but not limited to 'Is/Are', 'What', 'How', 'How many (times)', 'Which', 'Where', 'when' and 'If'. Incorporate different sentence structures, such as direct, indirect, conditional (e.g., 'If/Assuming ...', 'Excluding/Discarding ...', 'As ....'), rhetorical, **selective** (e.g., '... or ...?') and **negated** questions (e.g., 'Which is not ...?').

3. Include nested questions that require layered reasoning, as well as declarative sentences that prompt identification or clarification (e.g., 'Compared to/Comparing/Considering/Focusing on ...', 'In terms of/Given ...', 'Referring to ...', 'In/On/At/Across/Around/For/Of/Over/From/Between/By/Among/With/During/Towards ...', 'Based on ...', etc.). You can use a combination of one declarative instruction and one question, but must not to combine two questions.

4. Be sure to ask questions that address different dimensions of the data, such as time, categories, labels, names, values, and relative changes, rather than focusing only on the largest or smallest values (consider including **middle-ranked** values/labels such as 'the second', 'the third' etc., for a more balanced analysis).

5. If the chart contains subplots, prioritize questions that **involve reasoning across multiple subplots**, or focus on a specific subplot (by indicating its location, name, or ID) to explore detailed aspects of the data.

6. The generated questions should **exhibit clear visual references**, with an emphasis on the nuanced variations in details. **Be sure to ask how the legend labels or colorbar interact with the data.** The corresponding answers can be text or numbers, but not always number of values.

Response Format:
Please use the following JSON format to ensure clarity:

```json
{
  {
    "image_id": "{img_id}_1",
    "question": "Provide your question here",
    "rationale": "Provide your concise analysis and calculation procedures (MUST enumerate **all relevant elements** with **concrete values** and show **detailed calculations with specific values**)",
     "answer": "Provide complete final answer here",
     "rating": "Rate your confidence in this question, rationale and answer [0-5], Integer",
   },
  {
    "image_id": "{img_id}_2",
     ...
   },
     ...
}
```

Notes:
1. Focus only on aspects that can be inferred visually from the chart image. Do not refer to facts or terms that directly suggest the content is based on the code rather than the chart image itself.
2. Frame questions to encourage mathematical reasoning and logical deductions, avoiding sole reliance on chart data.
3. **MUST ENSURE** the rationale provides **numerical/text recognition or calculations**, converting visual questions into recognition- and calculation-based questions to support clear and logical reasoning.
4. The total number of questions should be **10 to 15**, and feel free to rephrase your questions to enhance diversity.
5. **MUST INCLUDE** one instance of "Not Applicable" for answer (different from "No") when appropriate. If a question relates to specific content that is missing in the image (especially for legend and colorbar, such as when asking about legend (e.g., no legend), values of colorbar (e.g., no colorbar)), answer "Not Applicable". Questions categorized as 'Not Applicable' should begin with 'What' and be generated based on your judgement.
6. Do not include questions about **alpha value, transparency, opacity, padding, spacing, grid, width, viewing angle, role, rotation, function (e.g.,'np.random'), font size, figsize, interpretation, visualization, significance, impact, save filename and save_path** in the code. Focus only on the meaningful visual elements in the chart and avoid mentioning 'code' in the questions, rationales and answers.
7. Ensure the image_id numbers increment sequentially, such as _1, _2, _3, _4, and so on. Keep the ```json``` formatting intact.

Figure 12. Prompt template for reasoning QA pairs generation. Prior to directly generating the final answer, we generate a rationale for in-depth analysis.

Prompt for Evaluating Model Predictions on ECDBench

Your task is to rigorously evaluate whether the VLM's prediction aligns with the expected answer for a question regarding a chart (note: the chart image itself is not provided here). The evaluation should focus solely on factual alignment between the prediction and the ground truth. Minor differences in wording or phrasing are acceptable as long as the core meaning remains consistent.

For numerical answers, **a margin of error up to 5% is acceptable** unless explicitly stated otherwise in the question. However, partial correctness or incomplete responses should not be considered correct.

- Question: {question}
- Expected Answer: {answer}
- Prediction: {prediction}

Please respond using the following format:
Correctness: (Yes or No)

Figure 13. Prompt template for evaluating model predictions on our constructed ECDBench. For numerical responses, we allow a tolerance range of 5% during evaluation.