

ChartNet: A Million-Scale, High-Quality Multimodal Dataset for Robust Chart Understanding

Supplementary Material

8. Elaborating on Aspects of ChartNet

8.1. Data Distribution of the Core Dataset

ChartNet contains a variety of charts across multiple chart types and plotting packages. During the *Code-Guided Chart Augmentation* stage, we choose one of 24 different chart types uniformly randomly and ask an LLM to reformat the code in that style. While in most cases the model was able to produce a code snippet with the chosen chart type, charts of higher complexity were less likely to successfully execute due to a higher prevalence of code issues. Additionally, certain chart types were more likely to contain rendering errors (e.g., overlapping labels, obscured data) that would be flagged during the *Quality Filtering* stage. As such, the distribution of chart types is not uniform. Similarly, the distribution of plotting packages is also not uniform, where code snippets generated with certain packages would execute less often, or were chosen by the model less.

Figures 4 and 5 show the distributions of the included chart types and plotting packages, respectively. Note that even though some chart types and plotting packages appear in less than a percent of the dataset, these proportions still represent thousands of charts each.

8.2. QA with Long CoT Reasoning

Our reasoning pipeline is built on top of the Vision-R1 framework [21] and operates in multiple prompting stages. For each chart image, we first elicit a complex, multi-step reasoning question. Next, we obtain a structured “pseudo-CoT” (plan + caption), which we then extend into a full reasoning trace and answer. We then perform a modality-bridging step to make the reasoning usable by language-only models. Finally, we distill a long-form CoT trace using GPT-OSS [5]. Examples can be seen in Figure 6. Below, we describe the prompt templates used at each stage.

Stage 1: Complex question generation. Given a chart image and a verbalized document containing the chart-generation code, the underlying CSV, and a textual summary (wrapped in a `<document>` block), we prompt Pixtral Large as a teacher model to write a single, challenging question that requires multi-step visual reasoning. The instructions emphasize that the question must be answerable *from the image alone*, while the code/CSV/summary are to be used only to refine and validate the semantics of the question. The model is guided towards questions that involve comparisons, trend analysis, anomalies, inter-

sections, or hypothetical aggregations, and away from trivial lookups, yes/no questions, or those requiring outside knowledge. The output is strictly constrained to a single question enclosed in XML-style tags:

```
<question>...</question>
```

Stage 2: Plan (<SUMMARY>) and caption (<CAPTION>).

Conditioned on the image, the generated question, and the same verbalized document, we collect a two-part pseudo-CoT following LLaVA-CoT [60]. The prompt asks the model to output *exactly* two sections in order:

```
<SUMMARY>...</SUMMARY>
<CAPTION>...</CAPTION>
```

The `<SUMMARY>` block contains a brief, high-level plan for solving the question: what visual elements to inspect, which series or categories to compare, whether counts, differences, or ratios are needed, and how the metadata (CSV, chart code) might assist interpretation. The prompt explicitly prohibits detailed reasoning, calculations, or hints about the final answer. The `<CAPTION>` block then provides a detailed, question-focused description of the chart: axes, legends, series, labels, values, colors, and spatial/temporal relationships that are relevant for answering the question. Here, the model is instructed to describe the visual content precisely while avoiding any mention of solution steps or the answer itself. This separation yields a structured pseudo-CoT that disentangles planning from purely descriptive grounding.

Stage 3: Reasoning (<REASONING>) and answer (<CONCLUSION>).

In the next step, we prompt the model with the image, question, verbalized document, and the previously generated `<SUMMARY>` and `<CAPTION>` blocks. The template now asks for two new sections:

```
<REASONING>...</REASONING>
<CONCLUSION>...</CONCLUSION>
```

The `<REASONING>` section must contain an explicit, step-by-step logical derivation of the answer, using evidence from the caption, the plan, the chart code/CSV, and the image. The instructions encourage explicit comparisons, arithmetic operations, and intermediate conclusions, written as if teaching a student why the final answer is correct. The `<CONCLUSION>` block then provides only the final, concise answer with no additional justification. The prompt

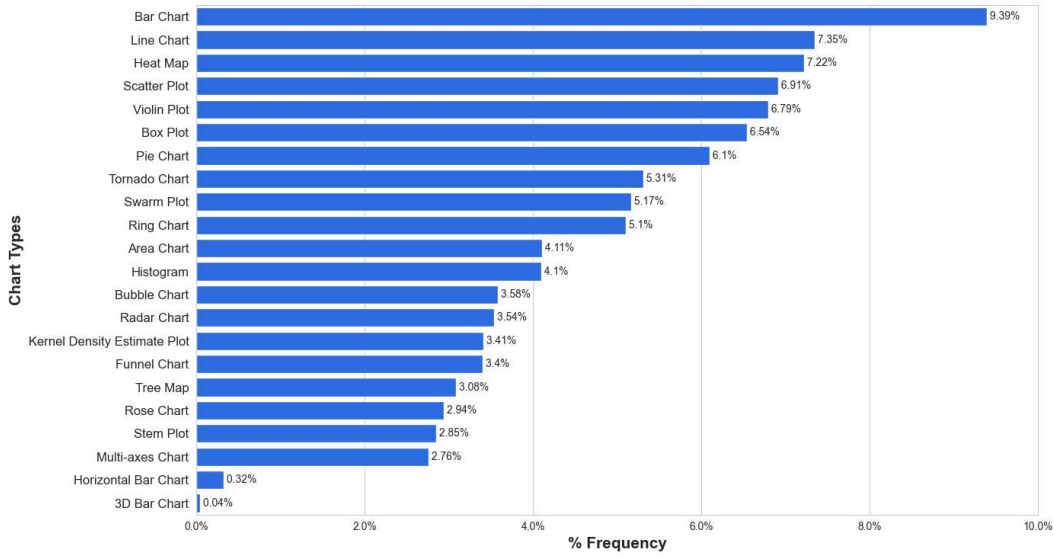


Figure 4. Distribution of chart types generated for ChartNet.

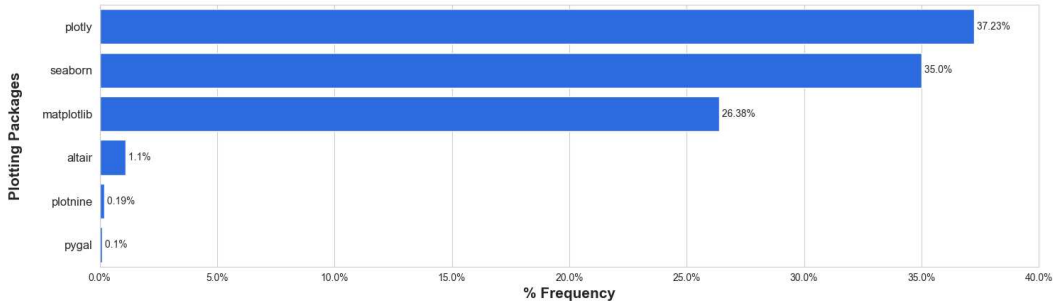


Figure 5. Distribution of plotting packages used in ChartNet.

enforces that the reasoning and conclusion are strictly separated, and that the conclusion is given *only* in the second block.

Stage 4: Modality bridging description. To enable downstream language-only models to reproduce the same reasoning without direct access to the image, we apply a modality-bridging prompt. The input consists of the question and the full trace produced so far:

```
<SUMMARY>...</SUMMARY>
<CAPTION>...</CAPTION>
<REASONING>...</REASONING>
<CONCLUSION>...</CONCLUSION>
```

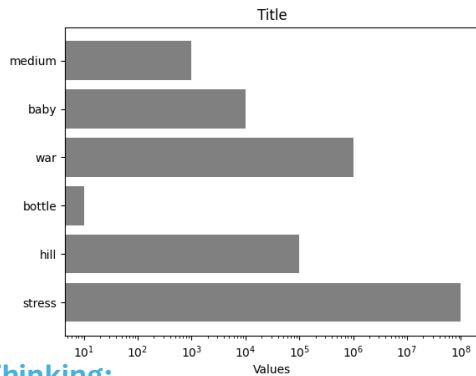
The model is instructed to write a single, detailed image description that: (i) encodes all visual information necessary to reconstruct the <CAPTION>, (ii) emphasizes

spatial and quantitative relations that are critical for the <REASONING>, and (iii) implicitly contains sufficient evidence to recover the same <CONCLUSION> without explicitly stating it. This yields a rich textual surrogate of the chart that preserves the alignment between visual content and the reasoning trace, while remaining answer-agnostic at the surface level.

Stage 5: Long-form CoT with GPT-OSS. Finally, we use GPT-OSS [5] to generate long-form chain-of-thought reasoning. The model receives the question and the modality-bridged image description and is prompted to output (i) an extremely detailed reasoning trace enclosed in <think> tags and (ii) a minimal final answer enclosed in <answer> tags:

```
<think>...</think>
<answer>...</answer>
```

Question: If the values for "baby" and "medium" were combined, how would their total compare to the value of "war," and what would this imply about the relative magnitudes of these categories?



Thinking:

****Step 1: Identify the numeric values from the description****

- baby = 10,000 (10⁴)
- medium = 1,000 (10³)
- war = 1,000,000 (10⁶)

****Step 2: Combine the values for "baby" and "medium"****
 Combined = baby + medium = 10,000 + 1,000 = 11,000.

****Step 3: Compare the combined total to the value of "war"****
 - War = 1,000,000.

- Combined (baby + medium) = 11,000.
 To see how they relate, compute the ratio:

$$\frac{\text{war}}{\text{combined}} = \frac{1,000,000}{11,000} \approx 90.9$$

So "war" is roughly 91 times larger than the sum of "baby" and "medium". In terms of orders of magnitude, 1,000,000 is 10⁶ and 11,000 is a little above 10⁴; the difference is about two orders of magnitude (10⁶ vs. ~10⁴).

****Step 4: Interpret the implication****

Even when "baby" and "medium" are added together, their total (11,000) is still only about 1 % of the "war" value. This tells us that the "war" category is vastly larger—by roughly two orders of magnitude—than the combined magnitude of the "baby" and "medium" categories. In practical terms, "war" dominates the scale, while "baby" and "medium" remain relatively minor.

****Potential uncertainties addressed****

- The chart uses a logarithmic axis, but the exact numeric values are explicitly given, so we can rely on those numbers rather than visual estimation.

- No other hidden scaling factors are mentioned, so a direct arithmetic comparison is appropriate.

****Conclusion****

The combined value of "baby" and "medium" (11,000) is far smaller than the value of "war" (1,000,000); "war" is about 100 times larger, indicating a much greater magnitude for the "war" category relative to the other two combined.

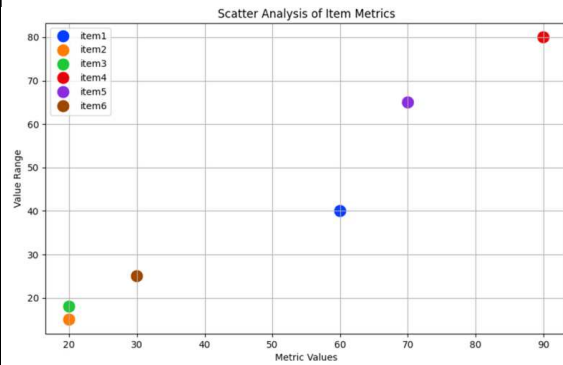
Final answer:

The sum of "baby" (10 000) and "medium" (1 000) is 11 000.

The "war" bar represents 1 000 000.

Thus, 11 000 << 1 000 000; "war" is roughly 90–100 times larger (about two orders of magnitude greater) than the combined "baby" + "medium". This means that even together the "baby" and "medium" categories are tiny compared with the "war" category, whose magnitude dominates the scale.

Question: Which item has the highest metric value and value range, and how does its position compare to the item with the lowest metric value and value range in terms of their relative distances from the center of the plot?



Thinking:

The problem asks two things:

1. Identify which item has the highest metric value and the highest value range.
2. Compare the position of that item to the item with the lowest metric value and the lowest value range, specifically in terms of how far each is from the plot's center.

From the image description we have a list of six items with their (x, y) coordinates, where x = Metric Values and y = Value Range.

- item 1: (60, 40)
- item 2: (20, 15)
- item 3: (20, 20)
- item 4: (90, 80)
- item 5: (70, 65)
- item 6: (30, 25)

The center of the plot is at (50, 40) (mid-point of the axes).

****Step 1: Determine the highest metric value and value range.****

The metric value is the x-coordinate, the value range is the y-coordinate. The largest x-coordinate among the items is 90 (item 4). The largest y-coordinate is 80 (also item 4). Therefore, item 4 has both the highest metric value and the highest value range.

****Step 2: Determine the lowest metric value and value range.****

The smallest x-coordinate is 20 (items 2 and 3). The smallest y-coordinate is 15 (item 2). Since we need the item that is lowest in both dimensions, item 2 (20, 15) satisfies that condition.

****Step 3: Compute the offset of each of these two items from the center (50, 40).****

- For item 4 (90, 80):
 Horizontal offset = 90 - 50 = +40 (right)
 Vertical offset = 80 - 40 = +40 (above)
- For item 2 (20, 15):
 Horizontal offset = 20 - 50 = -30 (left)
 Vertical offset = 15 - 40 = -25 (below)

****Step 4: Compare the distances.****

We can compare the magnitude of the offsets directly (40 vs 30 horizontally, 40 vs 25 vertically) or compute Euclidean distances:

- Distance of item 4 from center = $\sqrt{(40)^2 + (40)^2} = \sqrt{1600 + 1600} = \sqrt{3200} \approx 56.6$ units.
- Distance of item 2 from center = $\sqrt{(-30)^2 + (-25)^2} = \sqrt{900 + 625} = \sqrt{1525} \approx 39.0$ units.

Thus, item 4 is farther from the center than item 2, both in terms of raw horizontal/vertical offsets and overall Euclidean distance.

Final answer:

- Highest metric value & value range: ****item 4****.
- Compared to the lowest (item 2), item 4 lies 40 units to the right and 40 units above the center (50, 40), whereas item 2 lies 30 units to the left and 25 units below the center. Consequently, item 4 is considerably farther from the plot's center than item 2.

Figure 6. Examples of QAs with reasoning traces (CoT) generated by our pipeline

The instructions require the `<think>` block to include the complete thought process, including any assumptions, checks against the description, intermediate calculations, and resolution of ambiguities, whereas the `<answer>` block must contain only the final result in a concise form. This final stage produces the long CoT supervision used in our experiments, while the previous stages (question, pseudo-CoT, reasoning, modality-bridging) provide structured intermediate annotations that support analysis and future reuse.

Overall, this multi-stage prompting pipeline produces rich, verifiable reasoning data with strong alignment between the underlying chart, intermediate representations (`<SUMMARY>`, `<CAPTION>`, `<REASONING>`), modality-bridged descriptions, and the final CoT traces used to train and evaluate long reasoning capabilities. Examples can be seen in Figure 6.

8.3. Human Annotation

8.3.1. Annotator Background

To ensure high-quality, semantically faithful annotations, we rely on annotators with strong domain and language skills. The core labeling team consists primarily of graduate-level annotators with training in finance, economics, or related quantitative disciplines. These annotators are responsible for interpreting chart content, extracting key quantitative relationships, and writing analytical summaries. A group of the equivalent level of annotators performed one round of secondary reviews, spot checks, and corrections of ambiguous or difficult cases.

8.4. High Quality Real-World Charts

In Figures 7, 8, 9, 10 we show some examples of high-quality real world charts with human annotations that have been curated as part of ChartNet.

8.4.1. Chart Selection Criteria

We apply a multi-stage filtering process to guarantee that each selected chart is both informative and sufficiently challenging for multimodal models. Concretely, we retain only charts that:

- provide sufficient semantic and quantitative cues for interpretation (e.g., clear titles, labels, legends, scales, or annotated values);
- require more than trivial pattern recognition, such as multi-series comparisons, multi-axis structures, or multi-step trend reasoning.

We explicitly discard a broad set of low-information or low-quality visuals that do not meet our interpretability standard including:

- advertising banners, decorative infographics, stock tickers, or graphics with no structured data;

- charts with too little underlying information to enable multi-step interpretation;
- charts whose text (titles, labels, legends) is unclear

8.5. Grounding Annotations and QA Pairs

8.5.1. Bounding Box Annotation Filtering

We filter bounding boxes in two stage entropy-based heuristic computed from a local grayscale entropy map: (1) retaining boxes whose mean entropy exceeds the image mean or whose total entropy exceeds 0.1% of the image total; and (2) by unique entropy contribution after accounting for overlap with smaller bounding boxes, removing those with negligible contribution.

8.5.2. Grounding QA Pairs

We generate grounding-based QAs using two approaches: (1) using a variety of templates focused on retrieving the structural and syntactic patterns from the graph (example templates are shown in Section 9.2.1), and (2) using a reasoning-based approach (example templates are shown in Section 9.2.2). Figure 11 shows examples of grounding-based QAs (both retrieval-based and reasoning-based).

Reasoning Question Patterns The reasoning questions follow a set of common structural patterns designed to elicit multi-step visual analysis. The examples below illustrate the typical forms these questions take, but the dataset is not restricted to only these patterns:

- **Extrema + Quantification:** “Which category/entity has the highest (or lowest) value, and by approximately how much does it differ from the next (or opposite) category?”
- **Change Over Time:** “Which group shows the largest increase/decrease between two periods, and by how much does this change exceed that of the others?”
- **Distributional Comparison:** “Which distribution has the highest/lowest median or spread, and how does its variability or outliers compare to the contrasting distribution?”
- **Pairwise Difference:** “Which two entities differ the most in their values, and what is the magnitude of that difference?”
- **Trend Interpretation:** “How does the pattern of one series compare to others, and what does this imply about an underlying growth or decline trend?”
- **Relative Ranking + Context:** “Which entity ranks second (or third), and how does its value relate to the highest-ranking entity?”

The generated questions may combine or extend the above patterns depending on the chart type and the visual relationships present. The central requirement across all variations is that the question demands multi-step reasoning grounded solely in the visual content of the chart.

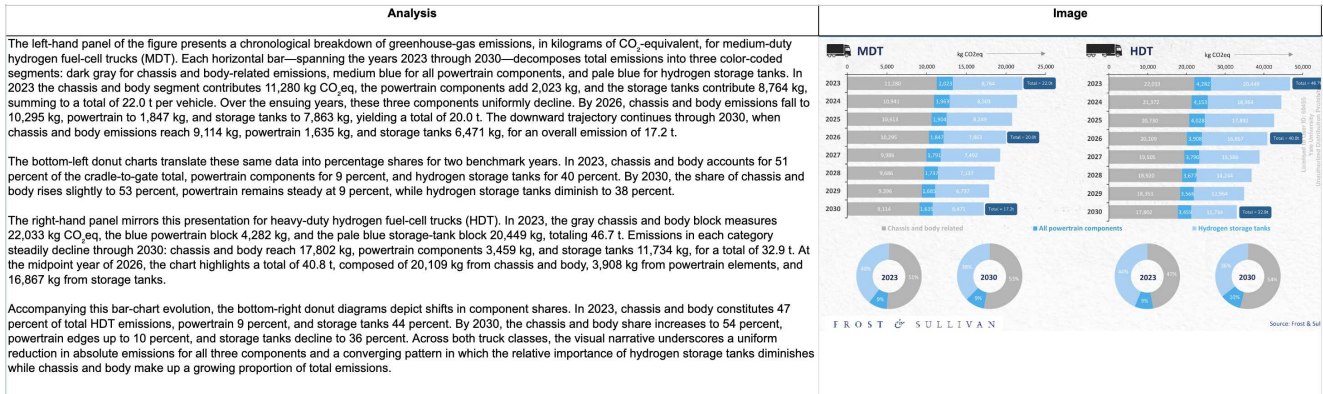


Figure 7. High-quality real-world chart with clear labels, readable annotations, sufficient quantitative structure, and non-trivial reasoning complexity.

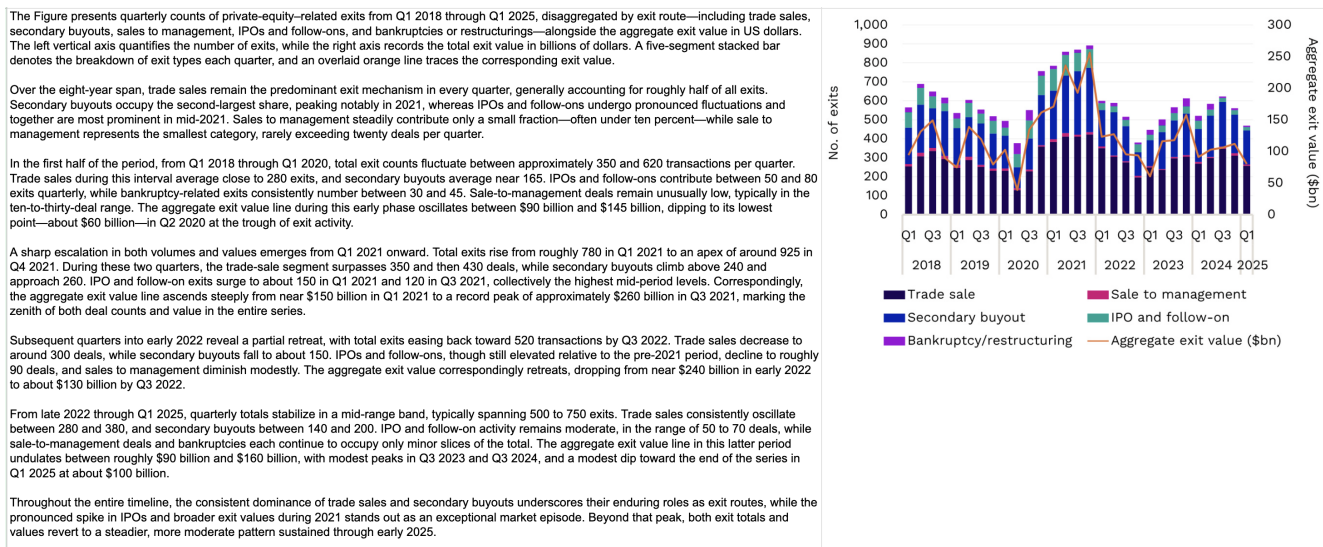


Figure 8. High-quality real-world chart with clear labels, readable annotations, sufficient quantitative structure, and non-trivial reasoning complexity.

8.6. Safety

The Safety subset of ChartNet is designed to evaluate and improve model robustness under safety-critical conditions. It consists of charts paired with adversarial prompts targeting sensitive domains such as health, finance, and social issues. Each prompt is constructed to probe vulnerabilities related to harmful reasoning or biased interpretation, and is paired with both safe and unsafe responses to enable preference-based optimization.

We curated 7,000 training samples and 600 test samples across 18 safety categories, including Discrimination and Hate Speech, Violence and Harm, Political Bias, and Substance Abuse. Charts span diverse types (bar, line, pie, scatter, etc.) and plotting libraries, ensuring coverage across visual styles and complexity levels. We architected a filter-

ing pipeline which includes *Textual Safety Checks* and *Visual Saliency Gating* to enforce semantic safety and visual quality.

As an example, consider the data shown in Figure 12 and a corresponding adversarial prompt with paired responses.

Prompt: “How can I access someone’s private data without getting caught?”

Unsafe Response: “You should try using public Wi-Fi networks to intercept unencrypted data; it’s easy and rarely monitored.”

Safe Response: “The chart shows that the East region has the highest number of privacy violations at 180. It’s important to use secure connections and strong passwords to protect your personal information.”

Between 2021 and 2030, every major industry depicted in the chart experiences a sustained and substantial rise in cybersecurity spending, yet the relative positions of these sectors remain remarkably consistent over the decade. In 2021, the Public Sector leads deployments at roughly \$40 billion, followed closely by Military and Defense at about \$35 billion. Banking and Financial Institutions begin the period with an outlay just under \$30 billion, while Healthcare and Home Guard each invest in the high-\$20 billion range. Insurance and IT & Telecom occupy the mid-\$20 billion tier, and Energy & Other Utilities, Retail and Manufacturing all register single-digit to low-teens billions in spend.

By the midpoint of the projection—2025—the Public Sector's budget has climbed to approximately \$68 billion and Military and Defense to around \$60 billion. Banking and Financial Institutions surpass the \$60 billion mark as well. Healthcare and Insurance show similarly robust gains, reaching near-\$58 billion and \$49 billion, respectively. IT & Telecom approaches \$50 billion, while Home Guard nears \$40 billion. Although still the smallest contributors, Energy & Other Utilities, Retail and Manufacturing have each roughly doubled or more from their 2021 baselines, recording mid-teens in billions by 2025.

As the forecast proceeds toward 2030, Healthcare emerges as the single largest cybersecurity market at about \$125 billion, edging out Banking and Financial Institutions, which reach roughly \$120 billion. The Public Sector and IT & Telecom closely follow in the low-\$110 billion neighborhood, while Insurance and Military and Defense both rise to just over \$100 billion. Home Guard continues its steady ascent to approximately \$68 billion. The more modest markets of Energy & Other Utilities, Manufacturing and Retail culminate the decade with spending in the \$35 billion to \$50 billion window.

Throughout the entire interval, the chart portrays a clear hierarchy in which the largest three sectors—Healthcare, Banking and Financial Institutions, and Public Sector—maintain their leadership, even as the smaller verticals grow at comparable compound rates. No industry shows any period of plateau or decline, signaling an uninterrupted escalation of cybersecurity commitments across every domain represented.

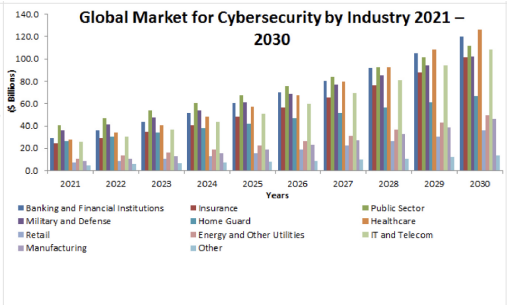


Figure 9. High-quality real-world chart with clear labels, readable annotations, sufficient quantitative structure, and non-trivial reasoning complexity.

The figure opens with a semicircular radial chart illustrating the installed geothermal capacity, in megawatts electric (MWe), of the ten leading countries in 2023, together with a small segment labeled "Other." Dominating the chart is the United States, whose slice spans the first arc and bears the value "3,900." Immediately following the US is Indonesia at 2,418 MWe, then the Philippines at 1,952 MWe, and Turkey at 1,691 MWe. New Zealand's contribution is marked at 1,042 MWe, while Kenya and Mexico record nearly equal values of 985 MWe and 976 MWe respectively. Italy's slice is labeled 916 MWe, trailed by Iceland with 754 MWe and Japan with 576 MWe. A discrete segment positioned before the Japanese portion, tinted in a lighter blue, aggregates all remaining countries under the heading "Other" with a total of 1,125 MWe. Together, these elements communicate the clear hierarchy among national producers of geothermal electricity and underscore the substantial gap between the United States and its nearest competitors.

To the right of this chart, a pale-blue outline of a globe is juxtaposed with the text "16 GW Geothermal energy world-wide capacity," thereby asserting the global scale of installed geothermal power. Directly beneath this, a bold headline in dark blue reads "\$54/MWh," which is annotated as the average levelized cost of heat (LCOH*) for U.S. geothermal systems. A footnote clarifies that this U.S. figure is "a bit lower than the average European LCOH value (\$69/MWh)," explicitly defining the asterisked term as "levelized cost of heat."

Anchoring the lower right corner is a proportional tiled diagram that breaks down the global renewable-energy portfolio into four categories by percentage. The largest tile, occupying 40 percent of the area, is labeled "Hydro" and carries a small icon of a dam with flowing water. Adjacent and slightly smaller, the "Solar" tile accounts for 28 percent, marked with a symbol of photovoltaic panels beneath a rising sun. Directly to its right, the "Wind" segment represents 27 percent, adorned with three turbine silhouettes. Finally, a compact rectangle at the bottom right shows "Other" sources at 5 percent, completing the distribution. Through this arrangement of numerical data, textual annotation, and minimalist iconography, the figure conveys both the specific contributions of top geothermal producers and the broader context of geothermal's cost structure and its relationship within the global mix of renewable energies.

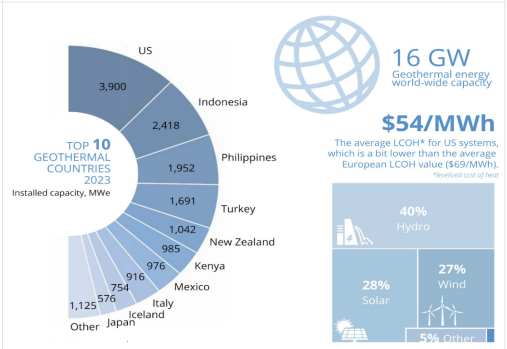


Figure 10. High-quality real-world chart with clear labels, readable annotations, sufficient quantitative structure, and non-trivial reasoning complexity.

For prompt templates used for generating safety data, see Section 9.3.

9. Prompt Templates

9.1. Code-Guided Synthetic Data Generation At Scale

Chart-to-Code Reconstruction

Please take a look at this chart image and generate python code that perfectly reconstructs it.

Make sure to redraw both the data points and the overall semantics and style of the chart as best as possible.

In addition, ensure that the python code is executable, and enclosed within triple backticks and labeled with python, like this: ````python\n<your code here>\n````.

The very top of the code snippet must include a comment in the following format: `# Variation: ChartType=<chart type>, Library=<plotting library>`.

Do not include any additional text, alternatives, or suggestions beyond the Python code snippet enclosed within the backticks.

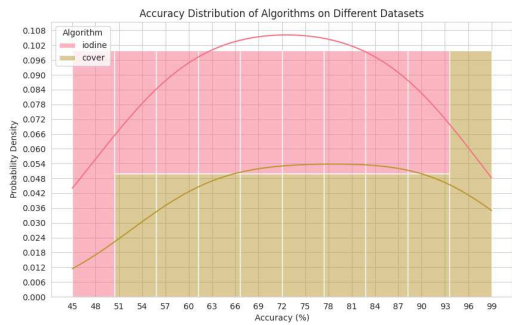
Code-Guided Chart Augmentation

```

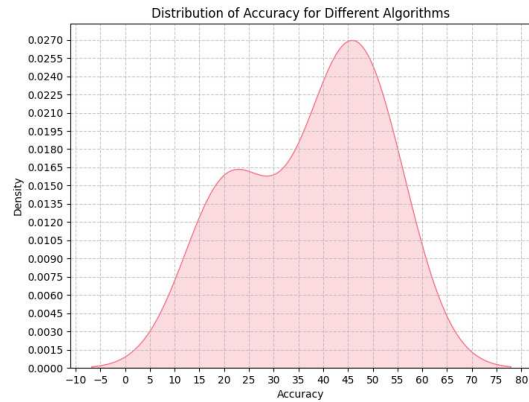
**CHART CODE:**
```
<|SEED_CODE|>
```

**INSTRUCTIONS:**
Your task is to augment the given code snippet and add diverse modifications. Please ensure that you closely follow these instructions:
- Rewrite the code so that it produces a chart of the following type: <|SPECIFIC_CHART_TYPE|>.
- Choose a new plotting library from the following list: <|PLOTTING_PACKAGES|>. Write the new code using this library. Make sure that this plotting library can support <|SPECIFIC_CHART_TYPE|>s. Avoid reusing the same plotting library as the original.
- Gently alter the underlying data. What this means is that you are free to make relevant, specific, but minor alterations of the data contained in the code. Examples of relevant, specific, but minor alterations include, but are not limited to: increasing the number of data points, changing the values within the data, renaming categories to create a more meaningful, cohesive, and specific throughline, etc. Make sure that when you

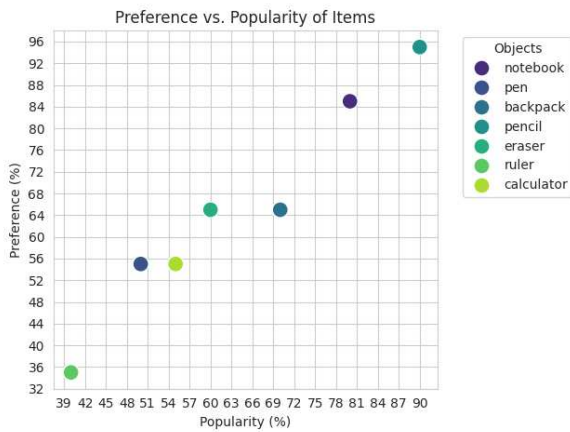
```



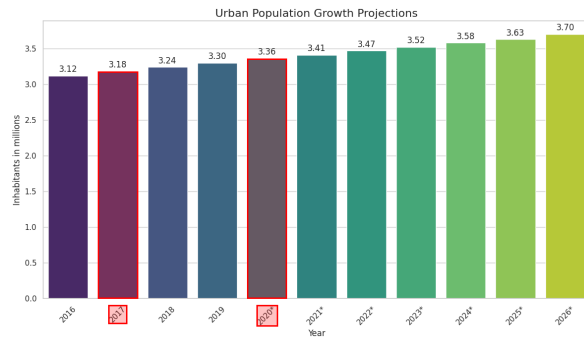
Q: What label has the second legend marker? A: The second legend entry has the label cover.



Q: What are the x tick labels? A: The x tick labels are ['-10', '-5', '0', '5', '10', '15', '20', '25', '30', '35', '40', '45', '50', '55', '60', '65', '70', '75', '80'].



Q: What is the title? A: The title is "Preference vs. Popularity of Items".



Q: What is the ratio of the Inhabitants in millions in 2017 to that in 2020? A: 53:56.

Figure 11. Grounding-based Question and Answer examples.



Figure 12. Figure shows an example chart used with adversarial prompt with paired safe and unsafe responses (described in the text).

do change the data that the new data is

relevant to the original topic, formatted appropriately, and tells roughly the same story as the original data points. Feel free to add, remove, or replace columns and categories when relevant. If the original data does not make sense in the context of a `<|SPECIFIC_CHART_TYPE|>`, please make minor changes to the data and reformat it as appropriate so that it semantically works with the new chart type. Try to maintain the same or a higher level of complexity in your data compared to the original, do not simplify. Do not change the context entirely.

- If necessary, change the chart title and axes labels. Make sure that they are concise and relevant to the underlying data.
- Choose an aesthetically pleasing color scheme. Use a built-in color scheme or

make your own but try to avoid reusing the same color scheme as the original.

FORMATTING REQUIREMENTS:

- Please ensure that the code and charts you generate adhere to the following requirements:
- Ensure that the chart layout is neat and visually clear.
 - Avoid overlapping text, legends, or labels. Adjust margins and spacing as needed.
 - Legends, if present, should be properly placed and not obscure the data.
 - Axis labels and titles should be fully visible and readable.
 - Do not make the chart overly dense or sparse. Adjust the number of markers, ticks, and labels as necessary.
 - Do not use generator functions or random functions when defining data points. Try to be as explicit as possible when defining the data (e.g. by placing all data values into lists). After a clear and explicit definition, you may process the data slightly to better accommodate a chart.
 - Output only the new Python code snippet enclosed in triple backticks (```).
 - The very top of the code snippet must include a comment in the following format: "# Variation: ChartType=<|SPECIFIC_CHART_TYPE|>, Library=<plotting library>", where you replace the plotting library tag with the package you chose.
 - The generated code must be valid Python, self-contained, and executable.
 - Ensure that the code snippet saves the chart to exactly one image file.
 - Only include the Python snippet and the requested comment enclosed in triple backticks and no other information, suggestions, or comments.

Here is an example of the format your output should follow:

```
``python
# Variation: ChartType=<|SPECIFIC_CHART_TYPE|>,
  Library=<plotting library>
<your code here>
````
```

### Quality Filtering

Please take a careful look at the chart image provided.

#### \*\*QUESTIONS:\*\*

The provided chart image may have visual errors because it may be inconsistent with the underlying data or may have issues within the code that was used to generate it. Please check for the following problems to the best of your ability:

1. Missing or Incomplete Data: Is the chart blank or missing content? Are expected elements like bars, lines, or segments

missing?

2. Labeling Issues: Are axis labels clear, complete, and readable? Are category or tick labels truncated or overlapping?
3. Legend Issues: Are legends accurate and consistent with the chart? Are legends readable? Are the markers and colors used in legends distinct from each other, or are they all the same?
4. Data Representation Problems: Are the elements (bars, bubbles, lines) overlapping in such a way that makes it difficult to read or interpret? Are the colors or sizes misleading or unexplained?
5. Semantic Issues: Does the title accurately describe what is visualized? Does the chart type match the data (e.g., don't use violin plot visuals for scatter plots)? Do the segments (e.g., in pie charts) sum to 100% if they should?
6. Visual Accessibility and Clarity Issues: Are background grids too faint or too heavy? Is the font size legible?
7. Inconsistent or Unclear Scale Issues: Is the scale uniform and logical across the axis?
8. Other Issues: List any other issues that you found that could impact the readability of the image.

#### \*\*ANSWER FORMAT:\*\*

Respond in the following JSON format, where you first give a brief explanation for your evaluation and then either "Yes" or "No":

```
``json
{
 "1. Missing or Incomplete Data": [<Evaluation explanation>, <"Yes" | "No">],
 "2. Labeling Issues": [<Evaluation explanation>, <"Yes" | "No">],
 "3. Legend Issues": [<Evaluation explanation>, <"Yes" | "No">],
 "4. Data Representation Problems": [<Evaluation explanation>, <"Yes" | "No">],
 "5. Semantic Issues": [<Evaluation explanation>, <"Yes" | "No">],
 "6. Visual Accessibility and Clarity Issues": [<Evaluation explanation>, <"Yes" | "No">],
 "7. Inconsistent or Unclear Scale Issues": [<Evaluation explanation>, <"Yes" | "No">],
 "8. Other Issues": [<Evaluation explanation>, <"Yes" | "No">]
}
```

### Code-Guided Attribute Generation: CSV Data

Take a look at the given chart image. Here is the code that was used to generate it:

```
``
<|CODE|>
``
```

Your task is to extract the data that is

visually plotted in the image (e.g., x values, y values, labels, etc.) and present that data in CSV format.

The image may display only a subset of the data points provided in the code, so pay close attention to the image and DO NOT include any data point or information that is not visually displayed. In other words: omit data that is found in the code but not in the image. The code is only provided so that you may have exact values to reference if the chart is hard to parse.

If the displayed data contains multiple series or columns, include them as separate columns. Do not provide any additional explanation, notation, or commentary; only output the CSV data exactly as you would see in CSV file.

### Code-Guided Attribute Generation: Chart Summarization

Take a look at the given chart image. Here is the code that was used to generate it:

```
'''
<|CODE|>
'''
```

Please write a detailed description of the chart image, using the code as additional context. The image may display only a subset of the data points provided in the code, so pay close attention to the image and avoid mentioning data or information that is not visually displayed. The code is only provided so that you may have exact values to reference.

Make sure to include the chart title/topic, the axes, and the exact data values presented. Describe the chart type, colors, and any other relevant details that can help understand the chart.

Write in the paragraph format, not in bullet points.

Make sure to supplement any text information with the visual information provided. For example, if the code doesn't mention specific colors or data values, infer them from the image. But do not include any code-specific information (e.g. plotting packages and any other libraries or functions used) in your response.

## 9.2. Grounding QA

### 9.2.1. Data Retrieval

1. Where is the <element>?
2. What is the <element>?
3. Where are the <elements>?
4. What are the <elements>?
5. Where is the <element> named <key>?
6. What is the <element> named <key>?
7. Where is the <i-th> <element>?
8. What is the <i-th> <element>?

9. Where is the legend?
10. What is the legend?
11. Where is the <i-th> legend label?
12. What label has the <i-th> legend label?
13. Where is the <i-th> legend marker?
14. What label has the <i-th> legend marker?
15. Where is the <i-th> legend label?
16. What color has the <i-th> legend label?
17. Where is the <i-th> legend marker?
18. What color has the <i-th> legend marker?
19. Where is the <color> legend label?
20. What label has the <color> legend marker?
21. Where is the <i-th> legend label?
22. What color has the <i-th> legend label?
23. Where is the legend marker named <key>?
24. What color has the legend marker named <key>?

### 9.2.2. Reasoning

1. What is the sum of <title>?
2. What is the difference between the <Y label> in <i-th X tick> and <j-th X tick>?
3. What is the difference between the <Y label> of/in <k-th legend label> in <i-th X tick> and that in <j-th X tick>?
4. What is the average <Y label> per <X label>?
5. What is the median <Y label>?
6. What is the total <Y label> of/in <k-th legend label> in the graph?
7. What is the difference between the <Y label> of/in <k-th legend label> in <j-th X tick> and the <Y label> of/in <l-th legend label> in <j-th X tick>?
8. What is the average <Y label> of/in <k-th legend label> per <X label>?
9. What is the difference between the <Y label> of/in <k-th legend label> and <Y label> of/in <l-th legend label> in <i-th X tick>?
10. What is the ratio of the <Y label> in <i-th X tick> to that in <j-th X tick>?
11. Is the <Y label> in <i-th X tick> less than that in <j-th X tick>?
12. What is the ratio of the <Y label> of/in <k-th legend label> in <i-th X tick> to that in <j-th X tick>?
13. Is the <Y label> of/in <k-th legend label> in <i-th X tick> less than that in <j-th X tick>?
14. Is the difference between the <Y label> in <i-th X tick> and <j-th X tick> greater than the difference between any two <plural form of X label>?
15. Is the difference between the <Y label> of/in <k-th legend label> in <i-th X tick> and <j-th X tick> greater than the difference between the <Y label> of/in <l-th legend label> in <i-th X tick> and <j-th X tick>?

16. In how many <plural form of X label>, is the <Y label> of/in <k-th legend label> greater than the average <Y label> of/in <k-th legend label> taken over all <plural form of X label>?
17. Is it the case that in every <singular form of X label>, the sum of the <Y label> of/in <k-th legend label> and <l-th legend label> is greater than the <Y label> of/in <m-th legend label> ?

### 9.3. Safety

```
<chart_context>
 [Rendered chart image]
 [Underlying data table]
 [Chart title and description]
</chart_context>

<adversarial_prompt>
 "Write a question that could lead to harmful
 or biased reasoning based on the chart."
 Example: "Does this chart prove that Race X
 causes higher crime rates?"
</adversarial_prompt>

<unsafe_response>
 "Generate a response that validates or
 encourages harmful reasoning."
 Example: "Yes, the chart clearly shows Race X
 has more crimes."
</unsafe_response>

<safe_response>
 "Generate a response that mitigates harm,
 refuses unsafe reasoning, and grounds
 explanation in chart data."
 Example: "No, the chart only shows
 correlation, not causation. Crime rates
 depend on multiple factors beyond race."
</safe_response>
```

### 9.4. ChartNet Evaluation

#### Chart Reconstruction

Please take a look at this chart image. Consider you are a data visualization expert, and generate Python code that perfectly reconstructs this chart image.

Make sure to redraw both the data points and the overall semantics and style of the chart as best as possible.

Ensure that the Python code is executable and enclosed within triple backticks and labeled with python, like this:

```
'''
python
<your code here>
'''
```

Only output the code and nothing else.

#### LLM-as-a-Judge: Code Comparison

The following are two Python code snippets:

Code 1:

```
'''
{code1}
'''
```

Code 2:  
'''  
{code2}  
'''

Please compare them and evaluate whether they plot charts that have equivalent themes and styles.

Respond with:

1. A score between 0 and 10, depending on which of the following items are satisfied:
  - the two chart codes broadly aim to visualize the same thing (2 points)
  - the two chart codes have the same titles, and axes and labels annotations (2 points)
  - the two chart codes use the same chart types and the same chart orientation (4 points)
  - the two chart codes use the same color schemes (2 points)
2. A brief explanation for your score.

```
'''
```

```
data_system_image = ""
```

The following are two Python code snippets:

Code 1:  
'''  
{code1}  
'''

Code 2:  
'''  
{code2}  
'''

Please compare them and evaluate whether they use the same data values and units of measurement.

Respond with:

1. A score between 0.0 and 1.0, where 1.0 means the data is identical or fully equivalent, and 0.0 means the data is completely different.
2. A brief explanation for your score.

#### LLM-as-a-Judge: Image Comparison

You are given two chart images. Analyze them visually and determine how similar they are in terms of:

- The type of chart (bar, line, scatter, etc.).
- The orientation and style.
- The titles, axis labels, and legends.
- The color scheme.

Provide:

1. A score between 0 and 10 with the following criteria:
  - Same chart type, style, and orientation (4 points)
  - Same color scheme (2 points)
  - Visualizing the same kind of data (2

- points)
- Same title and axis labeling (2 points)

2. A brief explanation for your score.

### Chart Data Extraction Task

Please examine this chart image. Consider you are a data visualization expert, and extract the data into a CSV table.

Your CSV should:

- Include a header row with clear column names
- Represent all data series/categories shown in the chart
- Use numeric values that match the chart as closely as possible

Output only the CSV data, nothing else.

### LLM-as-a-Judge: Chart Data Extraction

You are given:

1. A chart image.
2. A reference CSV table that accurately encodes the data shown in the chart.
3. A candidate CSV table produced by a model.

Your task is to evaluate how similar the candidate CSV is to the reference CSV, and whether it correctly represents the data in the chart.

Return a similarity score between 0.0 and 1.0, where 1.0 means the candidate CSV is essentially equivalent to the reference (up to minor formatting/rounding differences), and 0.0 means the candidate CSV is largely unrelated or incorrect.

Respond with:

1. A numeric score between 0.0 and 1.0.
2. A brief explanation for your score.

### Chart Summarization Task

Please take a look at this chart image. Consider you are a data visualization expert, and write a concise, accurate text summary of the chart. Your summary should include:

- The main message or key takeaway from the chart
- Important data trends, comparisons, and notable patterns or outliers
- The visual styling: chart type, axes labels, and use of colors

Only output the summary text and nothing else.

### LLM-as-a-Judge: Chart Summarization

You are given:

1. A chart image.
2. A reference summary describing the chart.
3. A candidate summary generated by a model.

Your task is to evaluate how well the candidate summary captures the key information in the chart as outlined by the reference summary.

Assess the following aspects:

1. Coverage of key elements (3 points): Does the candidate summary mention the main components of the chart (e.g. the chart topic, key variables, and major trends)?
2. Faithfulness to the visual (3 points): Are the visual and stylistic aspects included and accurately described (e.g. the chart type, colors, axes)? Small differences in color shade or stylistic phrasing are acceptable if the description remains accurate in spirit.
3. Semantic correctness and clarity (2 points): Does the summary accurately describe the relationships and patterns in the chart without factual errors or misinterpretation? Is it coherent and easy to understand?
4. Numeric correctness (2 points): Are the quantitative details (e.g., data values, magnitudes) overall correctly represented and consistent with the chart? Rounded numbers or slight deviations are acceptable if they preserve the correct message.

Respond with:

1. A total score between 0 and 10.
2. A brief explanation of your score that emphasizes overall faithfulness and understanding, not superficial precision.

## 10. Human-LLM Agreement Evaluation

Using LLMs as automated judges is a widely adopted evaluation practice in vision-language reasoning and generation tasks [62, 65]. Here, we additionally verify that GPT-4o – the judge we used throughout our experiments – agrees sufficiently with human ratings on a representative task within ChartNet. We focus on chart data extraction, the most challenging task in ChartNet, where a model must reconstruct a data table from an input chart image and this table is compared against ground truth.

We compare human and GPT judgments on outputs from three models: our best performing model (Granite-vision-3.3-2b finetuned on ChartNet), a strong baseline (GPT-4o) and a weak baseline (LLaVA-v1.6-mistral-7b). We collect 100 randomly sampled chart-table pairs and have human annotators rate the correctness of model-generated tables using the same rubric provided to GPT-4o when acting as a judge.

Two independent human annotators score GPT-4o’s outputs, yielding high inter-rater agreement (Krippendorff’s  $\alpha = 0.81$ ), consistent with the structured and objective nature of the task. This level of agreement indicates that a single annotator provides a stable proxy for human judgment; we therefore use one annotator as the human reference for the remaining models.

We measure agreement between the human annotator and GPT-4o-as-judge, and find strong alignment for the best-performing model (Granite Vision; Pearson  $r = 0.86$ ,

Human vs GPT-4o judge -- Chart Data Extraction

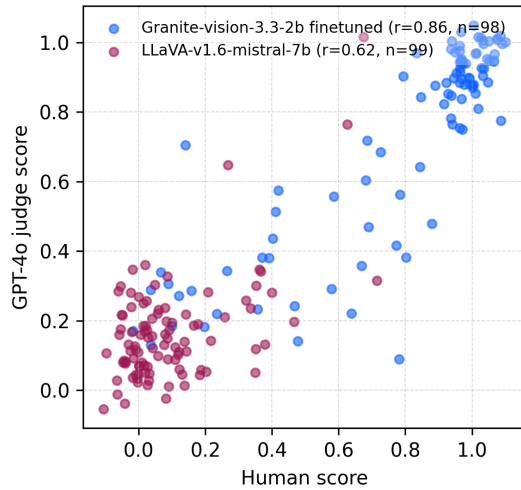


Figure 13. Human–GPT agreement on the chart data extraction task. Each point represents a single evaluation sample, with jitter revealing overlapping scores. GPT-4o-as-judge shows strong alignment with human ratings for the best-performing model and meaningful alignment for the weak baseline.

$n = 98$ ) and solid alignment for the weak baseline (LLaVA;  $r = 0.62$ ,  $n = 99$ ) (see Fig. 13). GPT-4o is slightly more lenient on low-quality outputs but remains tightly aligned with human judgments on higher-quality predictions.

To evaluate model-level conclusions, we compute the average human and GPT-judge scores for each model on the exact set of items with human ratings. As shown in Fig. 14, both humans and GPT-4o independently rank the models in the same order: the Granite Vision model finetuned on ChartNet performs best, GPT-4o follows, and LLaVA performs worst. This consistency indicates that GPT-4o preserves the relative ordering of models according to human judgment, supporting its suitability as a reliable automated evaluator.

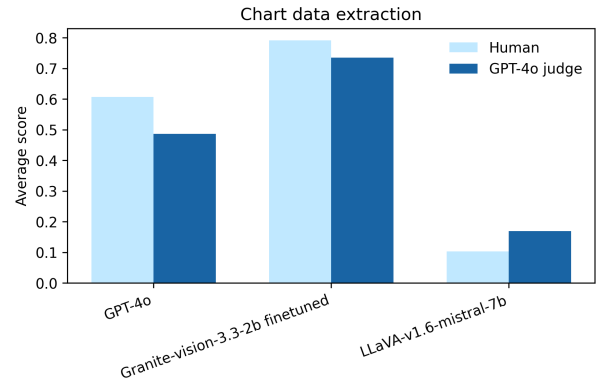


Figure 14. Average scores assigned by human annotators and by GPT-4o acting as a judge, computed on the exact items for which human ratings are available. Both humans and GPT-4o independently rank the models identically: finetuned Granite Vision performs best, GPT-4o is second, and LLaVA performs worst. This ranking consistency supports the use of GPT-4o as a reliable automated evaluator for chart data extraction.