Discover Data

Research

Evaluating LLMs for visualization generation and understanding

Saadiq Rauf Khan[1] · Vinit Chandak[1] · Sougata Mukherjea[1]

Received: 1 January 2025 / Accepted: 17 April 2025
Published online: 01 May 2025

Abstract
Information Visualization has been utilized to gain insights from complex data. In recent times, Large Language models (LLMs) have performed very well in many tasks. In this paper, we showcase the capabilities of different popular LLMs to generate code for visualization based on simple prompts. We also analyze the power of LLMs to understand some common visualizations by answering questions. Our study shows that LLMs could generate code for some simpler visualizations such as bar and pie charts. Moreover, they could answer simple questions about visualizations. However, LLMs also have several limitations. For example, some of them had difficulty generating complex visualizations, such as violin plot. LLMs also made errors in answering some questions about visualizations, for example, identifying relationships between close boundaries and determining lengths of shapes. We believe that our insights can be used to improve both LLMs and Information Visualization systems.

Keywords Large Language Models · Visualization Generation · Visualization Understanding

1 Introduction

With the amount and complexity of information produced increasing at staggering rates, information visualization is being utilized to enable people to understand and analyze information. Over the years, many techniques have been developed for creating information visualizations of different types of data. Information visualization can be created using various tools (for example, Tableau [1]), libraries in many programming languages (for example, matplotlib [2]), as well as scripts (for example, Vega-lite [3]). However, the complexity of these tools, libraries, and scripts can pose a barrier, especially for people without a strong background in data science or programming. To address this, automation of visualization creation using artificial intelligence techniques has also been explored [4].

Natural language interfaces allow users to generate visualizations using simple and intuitive commands. The integration of natural language processing into data visualization tools significantly improves the efficiency of data analysis. Analysts can now focus more on interpreting the data rather than the technicalities of creating visualizations. This advancement democratizes data analysis, making it more accessible to a broader audience, and facilitating more agile and responsive data-driven decision-making processes.

Large language models (LLMs) like GPT-3 [5] are capable of completing text inputs to produce human-like results. They have revolutionized Natural Language Processing by achieving state-of-the-art results on various tasks. Similarly,

S. R. Khan, V. Chandak and S. Mukherjea contributed equally to this work.

✉ Sougata Mukherjea, sougatam@iitd.ac.in; Saadiq Rauf Khan, saadiq351@gmail.com; Vinit Chandak, defozex47@gmail.com | [1]Indian Institute of Technology, Delhi, India.

deep learning models that are trained on a large amount of existing code can generate new code given some form of specifications such as natural language descriptions or incomplete code [6].

Another important task is the machine understanding of the visualizations. It accelerates data analysis by allowing machines to process and interpret large volumes of visual data quickly, reducing the time needed for manual interpretation. Moreover, it improves accuracy by providing consistent extraction of information from visualizations.

In this paper, we explore whether visualizations can be created or understood by prompting Large language models in natural language. Given the enormous potential of LLMs our aim was to explore whether LLMs are ready for Visualization tasks. Firstly, we evaluated whether popular LLMs like OpenAI's GPT-4 [7], Google's Gemini [8] and Anthropic's Claude [9] could generate code for visualizations based on some simple prompts. Secondly, we investigated whether the LLMs could understand simple visualizations and answer questions about them. Our analysis shows that for some tasks LLMs performed very well; for example, most LLMs could produce code to generate simple visualizations. However, our study has also exposed several limitations of the LLMs - they were incorrect in several tasks - both in generation and understanding.

The two main contributions of the paper are as follows:

1. We have performed an analysis of the capabilities of some popular LLMs to generate Python code and Vega-lite scripts for visualizations based on prompts.
2. We explored the power of LLMs to understand simple visualizations and answer questions about them.

The remainder of the paper is organized as follows. Section 2 cites related work. Section 3 analyzes the LLMs for visualization generation, while Sect. 4 analyzes the LLMs for Visualization Understanding. Finally, Sect. 5 concludes the paper.

2 Related work

2.1 Large language models

Large Language models (LLMs) such as GPT-3 [5] are capable of completing text inputs to produce human-like results. Its text completion capabilities can generalize to other Natural Language Processing (NLP) tasks such as text classification, question-answering, and summarization. Various prompt engineering techniques have been developed to find the most appropriate prompts to allow an LLM to solve the task at hand [10].

On the other hand, Codex, which contains 12 billion model parameters and is trained on 54 million software repositories on GitHub, has demonstrated incredible code generation capability, solving more 70% of 164 Python programming tasks with 100 samples [6].

2.2 Visualization generation

With the popularity of information visualization, many techniques have been developed to create visualizations for different types of data. Information visualization can be created using various tools, libraries in many languages, and scripts.

AI techniques have also been explored to automate the creation of visualizations, for example, using decision trees [11] and sequence-to-sequence recurrent neural networks [12]. ChartSpark [13] is a pictorial visualization authoring tool conditioned on both semantic context conveyed in textual inputs and data information embedded in plain charts. It embeds semantic context into charts based on text-to-image generative model, while preserving important visual attributes.

One significant direction of research is automating the creation of data visualizations based on user natural language queries. Many studies on natural language visualization (NL2VIS) are based on libraries of natural language processing; for example, DeepEye [14] uses OpenNLP [15], while NL4DV [16] uses CoreNLP [17]. These systems either have constraints on user input or cannot understand complex natural language queries [18]. Researchers have also trained neural networks using deep learning-based approaches [19, 20] to process complex natural languages. However, a single approach based on deep learning cannot perform well in various tasks. Benchmarks for the evaluation of NL2VIS systems are also being developed; examples include nvBench [21] and VisEval [22].

With the popularity of LLMs, there is significant interest in their application in various fields, including data visualization. Vázquez [23] investigates the capabilities of ChatGPT in generating visualizations. This study systematically evaluates whether LLMs can correctly generate a wide variety of charts, effectively use different visualization libraries, and configure individual charts to specific requirements. The study concludes that while ChatGPT shows promising capabilities in generating visualizations, there are still areas that need improvement.

Similarly, Li et al. [24] explored the ability of GPT–3.5 to generate visualizations in Vega-Lite from natural language descriptions using various prompting strategies. The key findings reveal that GPT–3.5 significantly outperforms previous state-of-the-art methods in the NL2VIS task. It demonstrates high accuracy in generating correct visualizations for simpler and more common chart types. However, the model struggles with more complex visualizations and tasks that require a deeper understanding of the data structure.

ChartGPT [25] leverages LLMs to generate graphs from abstract expressions. It breaks down the chart generation process into a series of subtasks for the LLM to solve sequentially.

LLMs have been integrated into NL2VIS systems, such as Chat2Vis [26] and LIDA [27], which generate Python code to construct data visualizations. However, a systematic evaluation of how well these LLMs can generate visualizations using different prompt strategies remains a need.

2.3 Visualization understanding

In recent times, various multimodal large language models (MLLMs) have been proposed for the understanding of charts. Examples include ChartLlama [28], UReader [29], and ChartAssistant [30]. Many datasets and benchmarks have also been introduced to test the capabilities of Large Language models and Multi-media Large Language models for chart understanding. Examples include Chart-to-Text [31], ChartQA [32], SciGraphQA [33], HallusionBench [34] and ChartBench [35].

Research has also been done to evaluate the Large language models in different aspects of visualization understanding.

- Bendeck and Stasko [36] evaluated GPT-4 for various visualization literacy tasks, including question-answering and identifying misleading visualizations. The assessment finds that GPT-4 can perform some tasks very efficiently, but struggles with some other tasks.
- Lo and Qu [37] explore the capabilities of several LLMs to detect misleading visualizations and assess the impact of different initiation strategies on model analysis. The evaluation concludes that there is significant potential in the use of MMLLMs to counter misleading information.
- Choe et al. [38] studied how LLMs can be used to help users with low data literacy understand complex visualizations such as Treemaps and Parallel Coordinates. The study found that LLMs helped users interpret charts and improve learning.

Although we used a small dataset, our study shows some interesting and novel insights on the performance of the popular LLMs in understanding visualization.

3 Analyzing LLMs for visualization generation

3.1 Process

To evaluate the capabilities of LLMs in generating information visualizations, we followed a process similar to Vázquez [23]. We prompt the LLM to create a visualization based on a given specification and examine the code generated by the LLM.

The methodology for the analysis involved several key steps:

1. Selection of visualization techniques
2. Selection of visualization methods
3. Creation or acquisition of suitable datasets
4. Selection of LLMs to analyze
5. Design and fine-tuning of prompts
6. Testing

Following these steps, we systematically evaluated the ability of each LLM to generate accurate and diverse visualizations, providing insights into their strengths and limitations.

3.1.1 Selection of visualization techniques

We began by identifying a diverse set of visualization techniques commonly used to ensure a comprehensive evaluation. We selected the following visualizations:

1. Area Chart
2. Bar chart
3. Box Plot
4. Bubble Chart
5. Bullet Chart
6. Choropleth
7. Column Chart
8. Donut Chart
9. Dot Plot
10. Graduated Symbol Map
11. Grouped Bar chart
12. Grouped Column Chart
13. Line chart
14. Locator Map
15. Pictogram Chart
16. Pie Chart
17. Pyramid Chart
18. Radar Chart
19. Range Plot
20. Scatterplot
21. Stacked Bar chart
22. Stacked Column Chart
23. Violin Plot
24. XY Heatmap Chart

These charts cover a broad range of commonly used visualization techniques, excluding hierarchical and network representations. In prompting LLMs to create this extensive set of visualization techniques, our objective was to comprehensively assess their capabilities to generate codes for accurate and diverse visualizations.

3.1.2 Selection of visualization method

Visualizations can be created in various ways. Two common methods are the use of libraries associated with programming languages and the use of scripts. We wanted to test the capabilities of LLMs for both of these methods.

We chose Python to generate visualization code due to its popularity, extensive representation in LLM training datasets, and wide array of visualization libraries. Note that when LLMs are prompted to make code for charts in Python, the models will use *Matplotlib* [2] by default. We also examine the ability of the LLMs to generate Vega-lite [3] scripts.

3.1.3 Creation or acquisition of suitable datasets

The visualization techniques described above require data sets with different types of variables, such as categorical, quantitative, and temporal variables. Some visualizations, such as locator maps, also require GPS location data. Below is a description of these datasets that we utilized:

- myfile.csv: Contains entries with a categorical variable and a quantitative variable, making it suitable for simple charts like bar charts and pie charts.
- heatmapdataOrig.csv: Stores random values ranging from 1 to 100 in a 50×50 matrix, used specifically to generate heatmap plots.

- iowa-electricity.csv: A sample file from the Vega Datasets, used for temporal data visualizations such as line charts and area charts.
- mycarsUnique.csv: A synthetic data set with 53 entries that have categorical and quantitative variables. This data set is suitable for grouped and stacked bar charts.
- carsMod.csv: This data set has multiple quantitative variables, making it ideal for bubble charts and scatterplots.
- myfileDotPlot.csv: A synthetic data set with a categorical variable and two quantitative variables, designed for dot plots.
- population2021.csv: Contains data on the population of countries in 2021 that are used for choropleth maps and graduated symbol maps.
- pyramiddata.csv: A synthetic data set that features age ranges and values for male and female populations used for pyramid charts.

These datasets cover a wide range of data types and structures, ensuring a comprehensive evaluation of the LLMs' ability to generate accurate and varied scientific visualizations.

3.1.4 Selection of LLMs to analyze

We utilized 4 LLMs - OpenAI's GPT—3.5 and GPT-4o [7] as well as Google's Gemini—1.5-pro [8] and Anthropics's Claude 3 Opus [9] for our analysis to provide a broad perspective on the capabilities of current models. Our choice of LLMs was based on the following reasoning.

- Established benchmarks: GPT—3.5 is a well-established LLM with a proven track record in various tasks. NL2VIS research has already used GPT—3.5, providing a baseline for comparison with other models and facilitating the interpretation of results within the existing research landscape.
- Cutting-edge advancements: Including GPT-4o allows exploration of the potential benefits offered by the latest advancements in LLM technology. Investigating how GPT-4o performs in visualization tasks compared to its predecessor, GPT—3.5, can provide insight into the impact of these advancements on visualization capabilities.
- Focus on different architectures: Gemini and Claude represent alternative LLM architectures. By including them, we can assess whether the observed performance in visualization tasks is specific to a particular architecture (e.g., GPT—3.5/GPT-4o) or applies across different LLM designs.

Analyzing the results across these models will offer a more comprehensive understanding of the strengths and limitations of LLMs in this domain.

3.1.5 Design and fine-tuning of prompts

We used the zero-shot prompting[1] for this task. We carefully designed and refined the prompts to maximize the effectiveness and accuracy of the LLMs in generating the desired visualizations. Some sample prompts are as follows:

- *Can you write a Python script that generates a Bubble chart using columns mpg (quantitative), disp (quantitative), and hp (quantitative) from the CSV file cars.csv?*
- *Can you write a Vega-lite script that generates a XY Heatmap chart using columns A (categorical), B (categorical) and C (quantitative) from the CSV file heatmapData.csv?*

3.1.6 Testing

We utilized the code or script generated by the LLMs as response to the prompts to create visualizations. We examined the generated charts to determine whether they satisfied all the requirements specified in the prompts.

[1] Zero-shot prompting is a machine learning technique that involves giving an AI model a task or question without providing any specific training or examples [10].

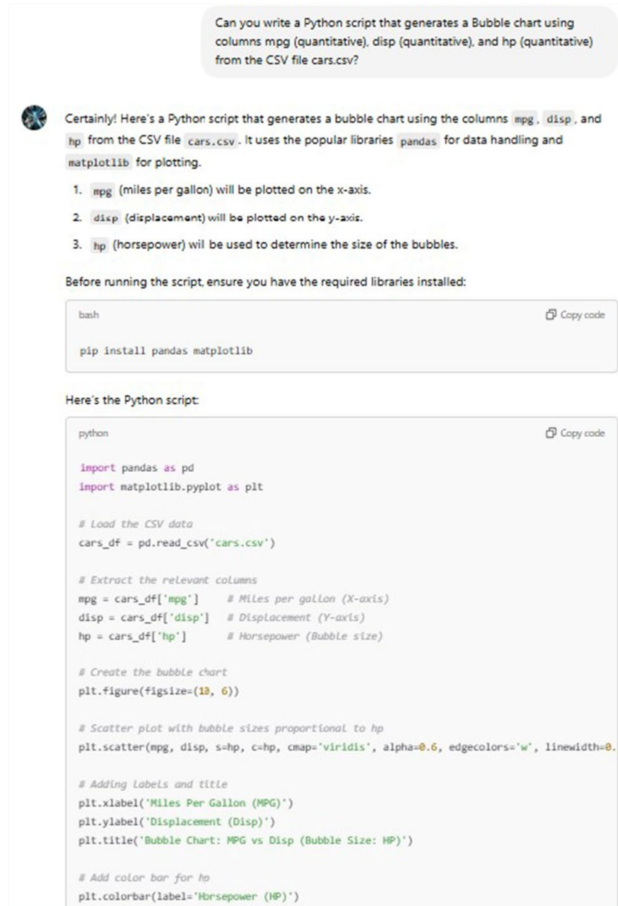

Fig. 1 Example of prompt and (portion of the) corresponding output produced by GPT–3.5

3.1.7 Experimental procedure

The assessment of LLMs focused on the accuracy, efficiency, and versatility of the models to produce effective visual representations of data. For each experiment, we followed the following process to ensure consistency and accuracy:

1. Initialize a New Session: Begin each experiment by creating a fresh session. Given that LLM chat sessions utilize previous prompts as context, it was crucial to start with a new session for each experiment. This approach ensured that each test was conducted independently, preventing any carry-over effects from previous prompts. For example, if multiple prompts requested charts in Vega-lite, subsequent prompts without a specified library or language might default to Vega-lite.
2. Consistent Prompt Input: Enter all prompts within the same session and on the same day to maintain uniform conditions.
3. Execute and Analyze: Utilize the LLM output (either Python code or Vega-lite scripts) to create a visualization and analyze it.

An example of a prompt and part of the output produced by a LLM (GPT–3.5) is shown in Fig. 1.

The generation of charts was divided into three categories:

1. Generation of Python code for charts with default configuration.
2. Generation of Python code for visually modified charts.
3. Generation of Vega-lite scripts for charts.

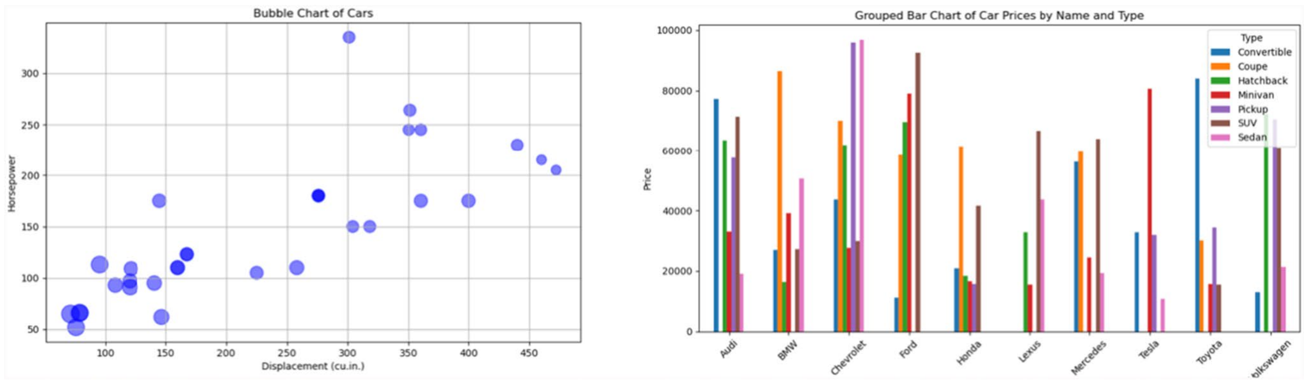

Fig. 2 Examples of some charts produced by GPT–3.5 via Python with default configuration

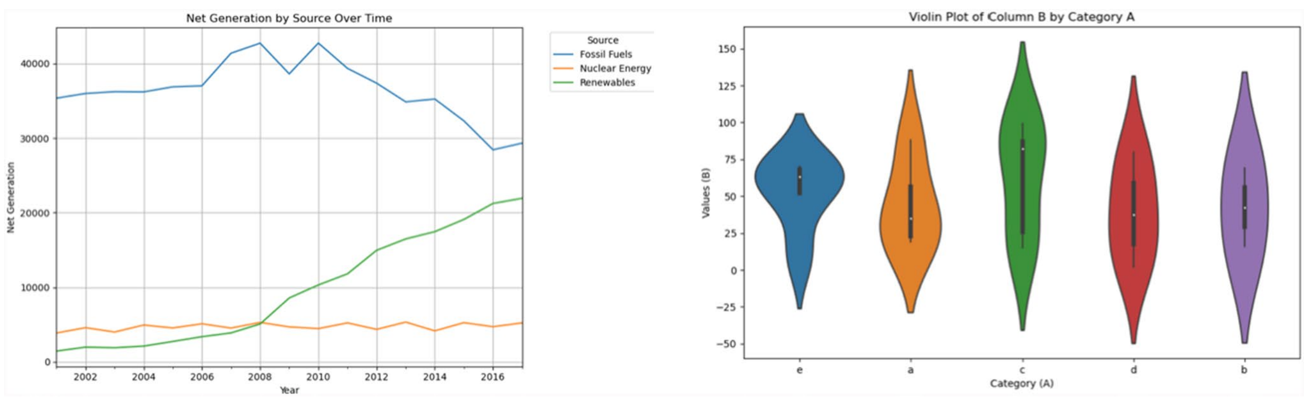

Fig. 3 Examples of some charts produced by GPT-4o via Python with default configuration

3.2 Chart generation with default configuration

In the first analysis, we evaluated the chart generation capabilities of the four LLMs chosen with default configuration, that is, without any other requirement. Each of the LLMs was prompted to generate Python code for the 24 different chart types. Examples of some charts generated by the different LLMs are shown in Figs. 2, 3, 4 and 5.

The performance of LLMs is shown in Table 1. GPT-4o proved to be the best performer with the ability to produce around 95% of the charts followed by GPT–3.5 being able to produce 79% of the charts. Gemini and Claude's were similar to that of GPT 3.5.

Note that correct generation means that the LLM could produce correct code for the visualization based on the requirement specified by the prompt. Since LLMs are known to produce inconsistent results, we tuned the LLM parameters so that randomness in the output is minimized. During the experiments, each prompt is repeated three times, and we accept the output only if they remain the same.

Most of the errors were due to the lack of knowledge of some LLMs on certain types of visualization, especially uncommon ones. Two interesting cases for comparison are the following.

- Range plot: GPT-4o and GPT–3.5 created correct visualizations. Gemini produced violin plots, while Claude produced box plots instead of producing range plots. The comparison is shown in Fig. 6.
- Bullet charts: Only GPT-4o was able to produce the correct chart. GPT–3.5 produced a Pyramid chart instead, whereas Gemini's and Claude's outputs were erroneous. The comparison is shown in Fig. 7.

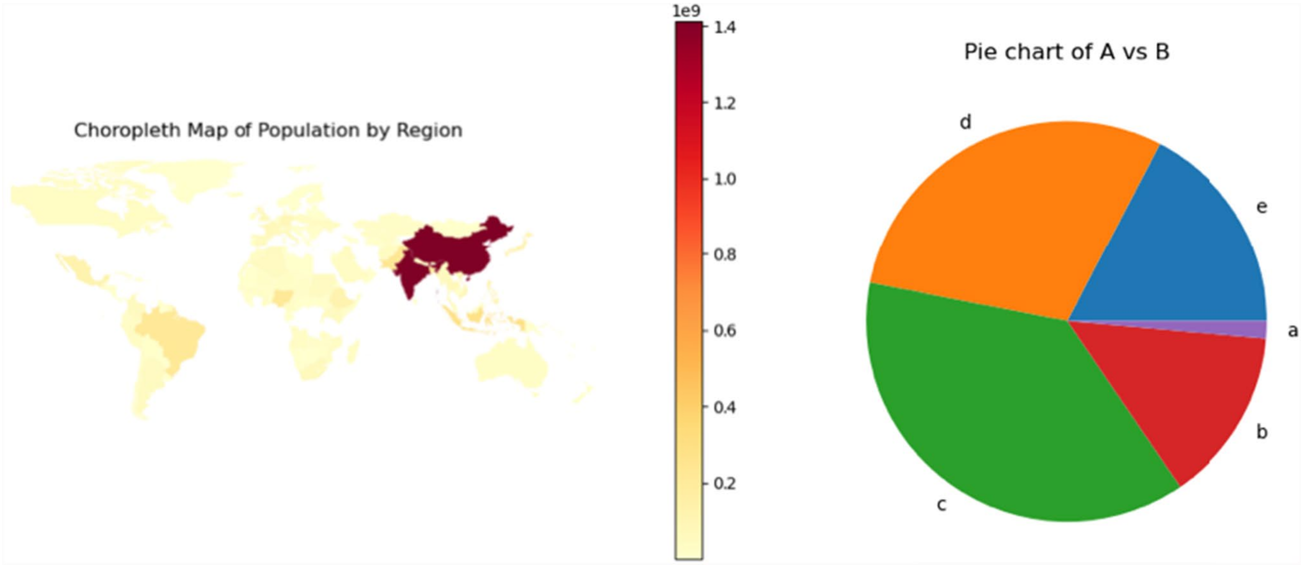

Fig. 4 Examples of some charts produced by Gemini via Python with default configuration

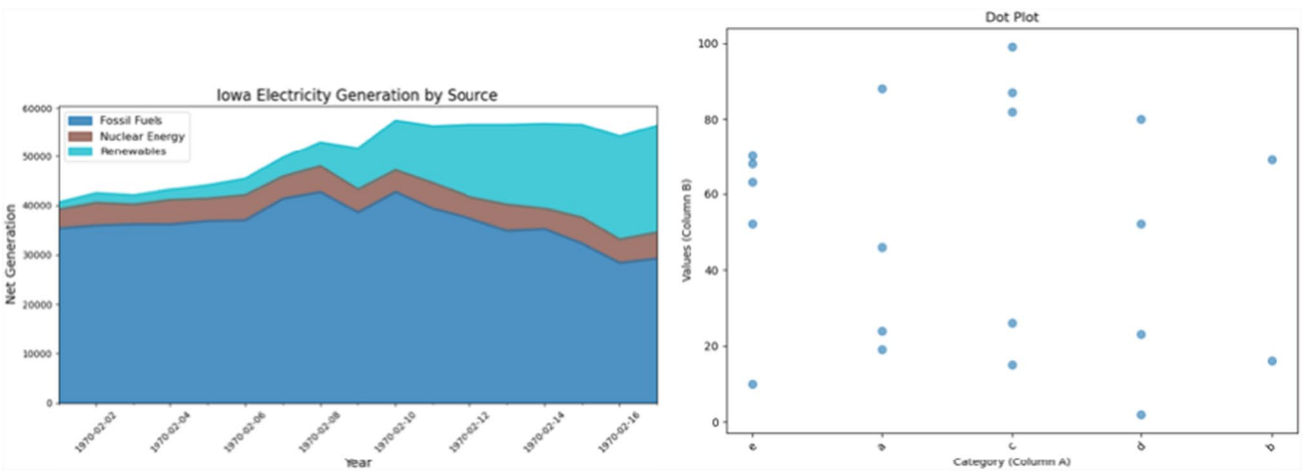

Fig. 5 Examples of some charts produced by Claude via Python with default configuration

3.3 Chart generation with changes in visual appearance

For further experimentation using the Python libraries, we decided to incorporate prompts that, along with the generation of charts, introduced modifications to the visual variables. We selected a subset of plots for this experiment: bar charts, line charts, scatter plots, bubble charts, and pie charts. To generate these charts, we modified the prompts with instructions to alter some default parameters. We tested changing visual variables based on fixed values (for example, a specific width or opacity) and making some variables dependent on others. These changes were related to the visual aspects of the marks, such as direction, size, width, color, shape, opacity, stroke, and ordering of the marks. In addition, we made adjustments to the annotations, including additional labels, their positions, font styles (e.g. boldface), and chart titles.

For each chart type, we tested different configurations based on some common modifications we typically make to our visualizations. Most changes were successful: the code was generated without errors, and the results were satisfactory with the exception of a few chart types. We hypothesize that one of the reasons for the errors is the large number of variables included in the prompts, as these plots required numerous visual variable configurations dependent on input data.

Table 1 Performance Comparison of LLMs in Chart Generation via Python with default configuration

Chart Type	GPT-3.5	GPT-4o	Gemini	Claude
Area Chart	Yes	Yes	Yes	Yes
Bar chart	Yes	Yes	Yes	Yes
Box Plot	Yes	Yes	Yes	Yes
Bubble Chart	Yes	Yes	Yes	Yes
Bullet Chart	No	Yes	No	No
Choropleth	Yes	Yes	Yes	No
Column Chart	Yes	Yes	Yes	Yes
Donut Chart	Yes	Yes	Yes	Yes
Dot Plot	No	Yes	Yes	Yes
Graduated Symbol Map	No	Yes	No	No
Grouped Bar chart	Yes	Yes	Yes	Yes
Grouped Column Chart	Yes	Yes	Yes	Yes
Line chart	Yes	Yes	Yes	Yes
Locator Map	Yes	Yes	Yes	Yes
Pictogram Chart	No	No	No	No
Pie Chart	Yes	Yes	Yes	Yes
Pyramid Chart	No	Yes	No	No
Radar Chart	Yes	Yes	No	No
Range Plot	Yes	Yes	No	No
Scatter Plot	Yes	Yes	Yes	Yes
Stacked Bar chart	Yes	Yes	Yes	Yes
Stacked Column Chart	Yes	Yes	Yes	Yes
Violin Plot	Yes	Yes	Yes	Yes
XY Heatmap Chart	Yes	Yes	Yes	Yes
Total	19(79%)	23(95%)	18(75%)	17(70%)

This time we dropped Claude because its performance was the worst in the previous test (as discussed earlier) and only carried out the experimentation with GPT–3.5, GPT-4o and Gemini.

For the modified charts in Python, the results are shown in Table 2. The table also indicates the type of visual changes we made for each test. Here again, GPT-4o emerged as the best performer and was able to produce more than 92% of the proposed charts. GPT–3.5 and Gemini also performed well with the accuracy of 88% and 84% chart generation.

Some important cases to note are the following.

- Bar chart: The prompt specified that the bars should have a width of 10 pixels. GPT–3.5 and GPT-4o could satisfy the requirement. However, Gemini was trying to reduce the bar width by compressing the width of the chart instead of actually reducing the width of the actual bar. The scenario is shown in Fig. 8.
- Line chart: Here, GPT–3.5 failed to produce the requirement of green lines and purple squares as points. However, GPT-4o could satisfy the requirement. The scenario is shown in Fig. 9.
- Scatter plot: Here, GPT-4o did not satisfy the prompt requirement to use labels or colors to differentiate between the different source types. The comparison is shown in Fig. 10.

3.4 Chart generation via vega-lite scripts

Finally, we wanted to test and compare the performance of the aforementioned LLMs to generate Vega-lite scripts as well. Here again we prompted the LLMs to generate Vega-lite scripts for all the twenty-four selected charts. For this final test, we utilized GPT-4o and Gemini for experimentation. Some examples of charts generated by the LLMs via the Vega-lite scripts are presented in Fig. 11.

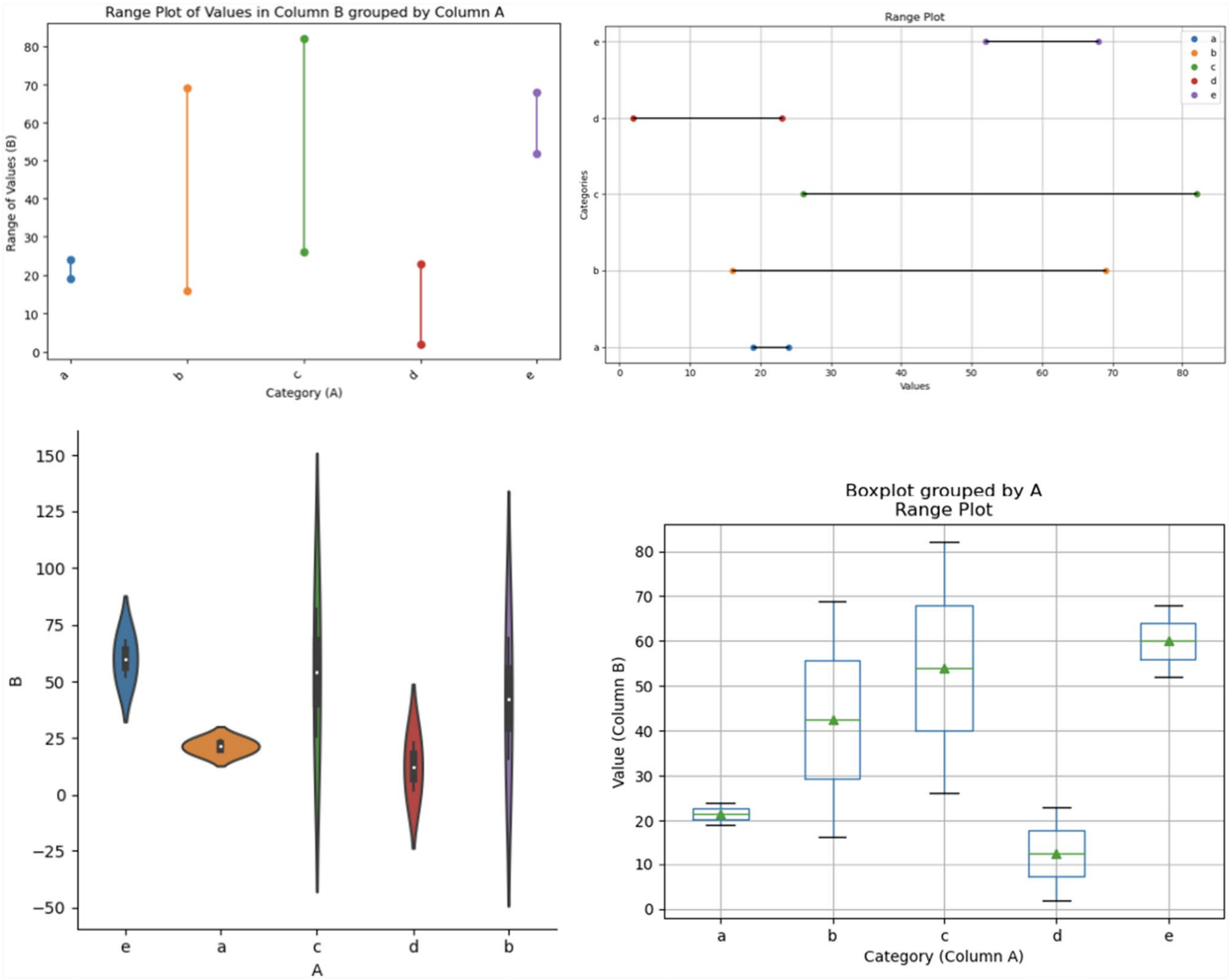

Fig. 6 Comparison of range plots. GPT-4o and GPT−3.5 (on top) created correct visualizations. Gemini (bottom left) produced violin plots while Claude (bottom right) produced box plots

For the Vega-lite scripts, the results are shown in Table 3. Here, the performance of GPT-4o was not as good as in the previous tests; it was only able to produce 70% of the charts. Gemini's performance was worse–it was only able to produce 24% of the charts.

Some interesting cases for comparison are the following.

- Pyramid chart: Here, Gemini was not able to produce the required chart. GPT-4o's output was visually correct, but the labels were not correct. The comparison is shown in Fig. 12.
- Violin plot: Both GPT-4o and Gemini could not produce violin plots as shown in Fig. 13.
- Locator map: GPT-4o's output was correct. Gemini could not produce the map. The comparison is shown in Fig. 14.

3.5 Key findings

- Accuracy and Diversity of Visualizations:
  - GPT-4o demonstrated the highest accuracy; the other models, while competent, showed varying degrees of effectiveness in handling different types of visualization.

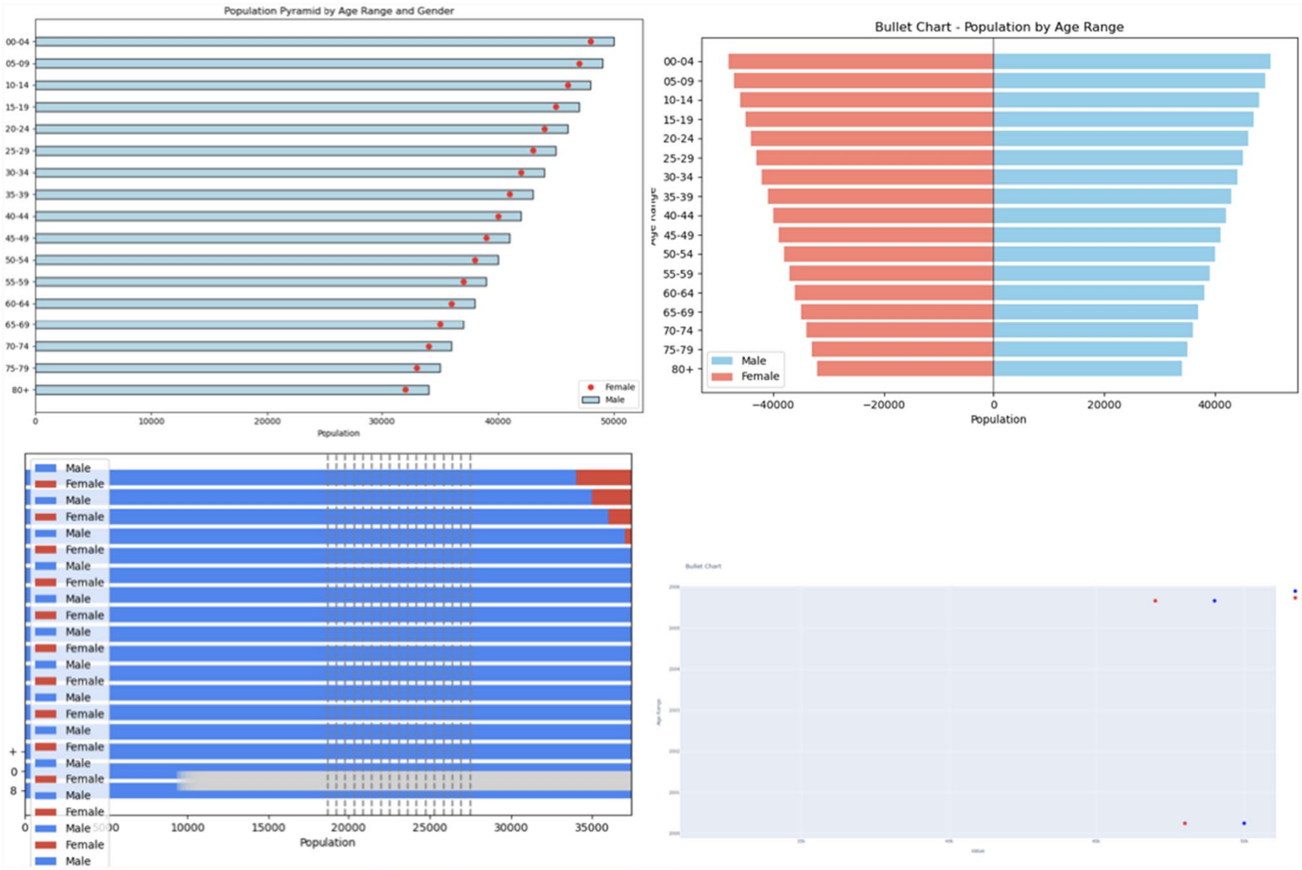

Fig. 7 Comparison of bullet charts. Only GPT-4o (top left) was able to produce the correct chart. GPT–3.5 produced a pyramid chart instead (top right). Gemini's and Claude's outputs were erroneous (bottom)

- The models performed well with common chart types, such as bar charts, line charts, and scatter plots. However, more complex visualizations like bullet charts and radar charts posed challenges for some LLMs, particularly Claude and Gemini.

- Code Variation:
  - All evaluated LLMs effectively utilized visualization libraries such as Matplotlib to produce Python code. There were no significant differences in the models' ability to employ these libraries for the generation of standard visualizations.
  - Although the generated Python code was generally accurate, there were instances of inefficiencies and redundancies. Improving the efficiency and optimization of generated code remains an area for further development
  - Vega-Lite proved to be difficult for the LLMs, for generating correct visualizations. Gemini was rendered almost useless for creating charts via Vega-lite scripts, being only able to create roughly one-fourth of the total charts. The performance of GPT-4o also reduced significantly when switching from Python to Vega-lite.

- Customization and Flexibility:
  - The LLMs exhibited varying degrees of customization capabilities. Although they could adjust basic visual variables such as color, size, and labels, more nuanced customization often required precise and detailed prompts.
  - GPT-4o and GPT–3.5 showed superior flexibility in generating visualizations that closely matched user specifications, indicating their advanced comprehension and execution abilities.

Table 2 Performance comparison of LLMs in visually modified chart generation. The prompts were modified with instructions to alter some default parameters

Chart Type	Modification	GPT-3.5	GPT-4o	Gemini
Bar chart	With horizontal bars	Yes	Yes	Yes
Bar chart	With bars in light green	Yes	Yes	Yes
Bar chart	With bars of width 10 pixels	Yes	Yes	No
Bar chart	With the bars of different colors	Yes	Yes	Yes
Bar chart	With title "This is a Bar chart generated by an LLM"	Yes	Yes	Yes
Bar chart	With labels indicating the quantities on top of the bars	Yes	Yes	Yes
Line chart	With all lines in purple	Yes	Yes	Yes
Line chart	With lines of width 10 pixels	Yes	Yes	Yes
Line chart	With the opacity of the lines as 0.5	Yes	Yes	Yes
Line chart	Marking the data points with circles	Yes	Yes	Yes
Line chart	With green lines and the data points encoded with purple squares	No	Yes	Yes
Line chart	With dashed lines	Yes	Yes	Yes
Pie Chart	Sorting the values from larger to smaller	Yes	Yes	Yes
Pie Chart	Sorting the values from smaller to larger	Yes	Yes	Yes
Pie Chart	With the labels in boldface	No	Yes	Yes
Pie Chart	With the title "Pie"	Yes	Yes	Yes
Pie Chart	With a sequential color palette that depends on the B column	Yes	Yes	Yes
Pie Chart	Sorting the values from larger to smaller, clockwise, and starting at 90 degrees	Yes	No	No
Pie Chart	Sorting the values from larger to smaller, clockwise, and starting at 90 degrees, and not displaying the categorical labels, only the quantities, and outside the pie	No	No	No
Bubble Chart	With points as triangles, whose size depends on column D	Yes	Yes	Yes
Bubble Chart	With the shape of points depending on column B	Yes	Yes	Yes
Scatter plot	With the shape of points depending on column B and their size dependent	Yes	Yes	Yes
Scatter plot	On column D with the opacity encoding the values in E column	Yes	Yes	No
Scatter plot	Using columns C and D for X and Y axis and the size defined by the A column	Yes	Yes	Yes
Scatter plot	With the shape of points depending on column B and their size dependent on column D	Yes	Yes	Yes
Total		22(88%)	23(92%)	21(84%)

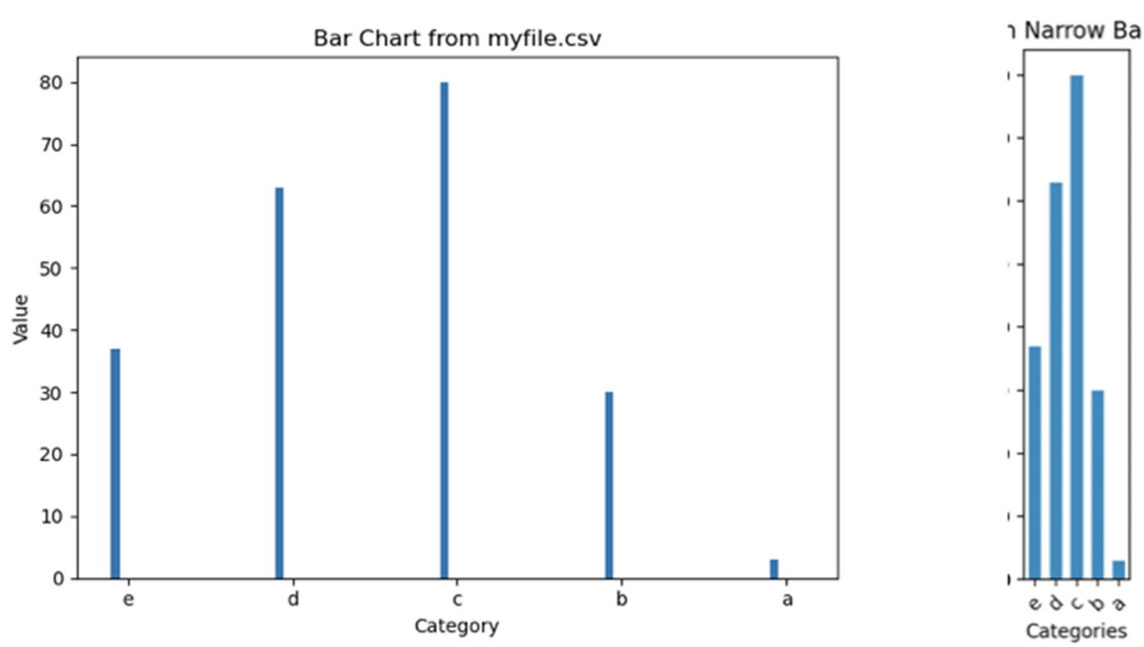

Fig. 8 Comparison of visually modified bar charts. GPT-4o (on left) could satisfy the requirement of reducing the width of the bar. However, Gemini was trying to reduce the bar width by compressing the width of the chart instead of actually reducing the width of the actual bar (as shown on the right)

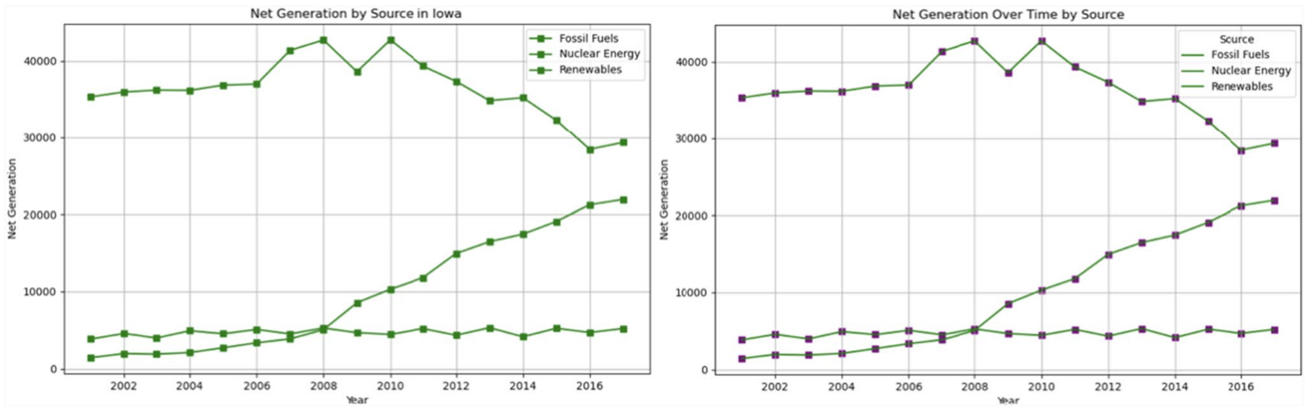

Fig. 9 Comparison of visually modified line charts. Here GPT–3.5 (on left) failed to satisfy the requirement of green lines and purple squares as points. However, GPT-4o could satisfy the requirement (on right)

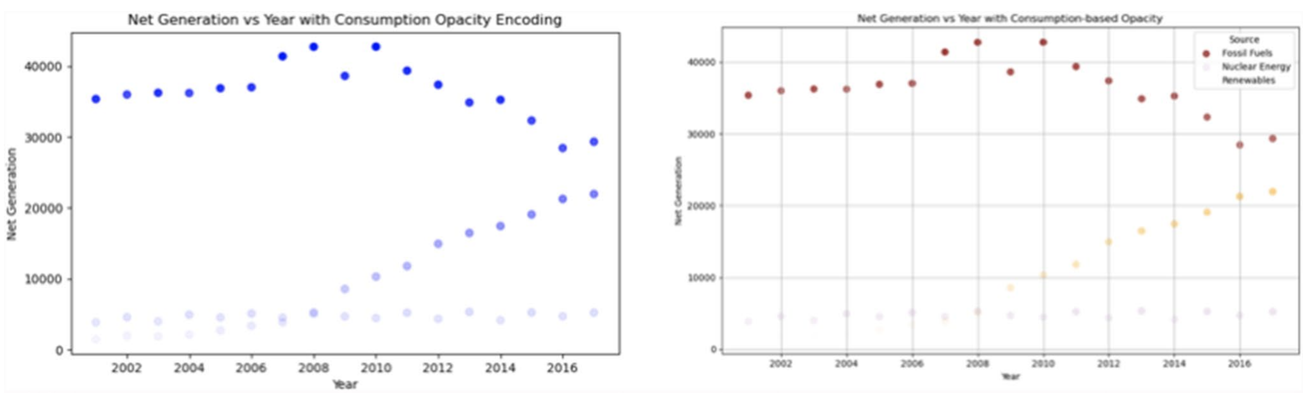

Fig. 10 Comparison of visually modified scatter plots. GPT-4o (on left) did not follow the prompt to use labels or colors to differentiate between the different source types. GPT–3.5 (on right) satisfied the requirement

4 Analyzing LLMs for visualization understanding

4.1 Data set

For analyzing the capabilities of LLMs for understanding visualization, we have used the FigureQA dataset [39]. We chose this data set to evaluate the basic capabilities of LLMs for visualization understanding since it consists of some very common charts for tabular data accompanied by questions and answers concerning them. The corpus is synthetically generated on a large scale and has five common visualizations for tabular data, namely, horizontal and vertical bar graphs, continuous and discontinuous line charts, and pie charts. These figures are produced with a white background, and the colors of the plot elements (lines, bars, and pie slices) are chosen from a set of 100 colors. Figures also contain common plot elements such as axes, gridlines, labels, and legends. A sample chart and the corresponding questions are shown in Fig. 15.

The question-answer pairs for each figure have been generated from its numerical source data according to predefined templates. There are 15 types of questions, as shown in Table 4, which compare quantitative attributes of two plot elements or one plot element versus all others. In particular, the questions examine properties such as the maximum, minimum, median, roughness, and greater than/less than relationships. All are posed as a binary choice between yes and no.

4.2 Automated analysis on FigureQA

To evaluate the ability of LLMs to understand and answer questions about information visualization, we randomly chose 100 images from the data set and the corresponding 1, 342 questions. Our random choice of the images will hopefully

Fig. 11 Example of charts generated through Vega-lite scripts

lead to variations in the chart types. We evaluated 3 LLMs - Google's Gemini–1.5-pro [8], OpenAI's GPT-4o [7] and Anthropics's Claude 3 Opus [9]. The results of the evaluation are shown in Table 5.[2]

As we can see, GPT-4o is the best performer, followed by Gemini–1.5-pro which is slightly behind and then Claude 3 Opus, which is much worse when compared to the other two models.

4.3 Need for manual analysis

While this initial automated test with the FigureQA dataset provided quantitative metrics for evaluating the performance of the selected LLMs, we know that relying solely on binary questions does not offer a comprehensive assessment of the model's true comprehension abilities. The binary nature of the FigureQA questions introduces a significant limitation: susceptibility to random guessing. Models can achieve approximately 50% accuracy by making random choices without truly understanding the content of the figure/chart. This might lead to misleading performance evaluations, as the models might appear to perform well despite not understanding the underlying charts at all.

Binary questions, while convenient for initial testing, lack the depth and complexity necessary to fully evaluate a model's reasoning capabilities. They oversimplify the issue by restricting the options to a binary "yes" or "no", which fails to account for the complex relationships that must be understood in visual data. This oversimplification can mask underlying flaws in the models' reasoning processes and provide an inflated estimation of their true capabilities.

[2] Unfortunately, we could not use a larger data set since there is a cost associated with each invocation of the LLM APIs.

Table 3 Performance Comparison of LLMs in Chart Generation via Vega-lite scripts

Chart Type	GPT-4o	Gemini	Remarks
Area Chart	Yes	Yes	Both similar and correct
Bar chart	Yes	Yes	Both similar and correct
Box Plot	Yes	Yes	Both similar and correct
Bubble Chart	Yes	No	Gemini threw error
Bullet Chart	No	No	Both created incorrect charts
Choropleth	Yes	No	Gemini created a blank chart
Column Chart	Yes	Yes	Both similar and correct
Donut Chart	Yes	Yes	Both similar and correct
Dot Plot	Yes	No	Gemini threw error
Graduated Symbol Map	Yes	No	Gemini created a blank chart
Grouped Bar chart	No	No	GPT-4o: not grouped but separate, Gemini: stacked not grouped
Grouped Column Chart	No	No	GPT-4o: not grouped but separate, Gemini: stacked not grouped
Line chart	Yes	No	Gemini threw error
Locator Map	Yes	No	Gemini threw error
Pictogram Chart	No	No	Gemini created a blank chart, GPT-4o created incorrect chart
Pie Chart	Yes	Yes	Both similar and correct
Pyramid Chart	Yes	No	Gemini created incorrect chart, GPT-4o created correct chart
Radar Chart	No	No	Both created incorrect charts
Range Plot	No	No	Both created incorrect charts
Scatter Plot	Yes	Yes	Both similar and correct
Stacked Bar chart	Yes	Yes	Both similar and correct
Stacked Column Chart	Yes	Yes	Both similar and correct
Violin Plot	No	No	Both created incorrect charts
XY Heatmap Chart	Yes	Yes	Both similar and correct
Total	17(70%)	10(41%)	

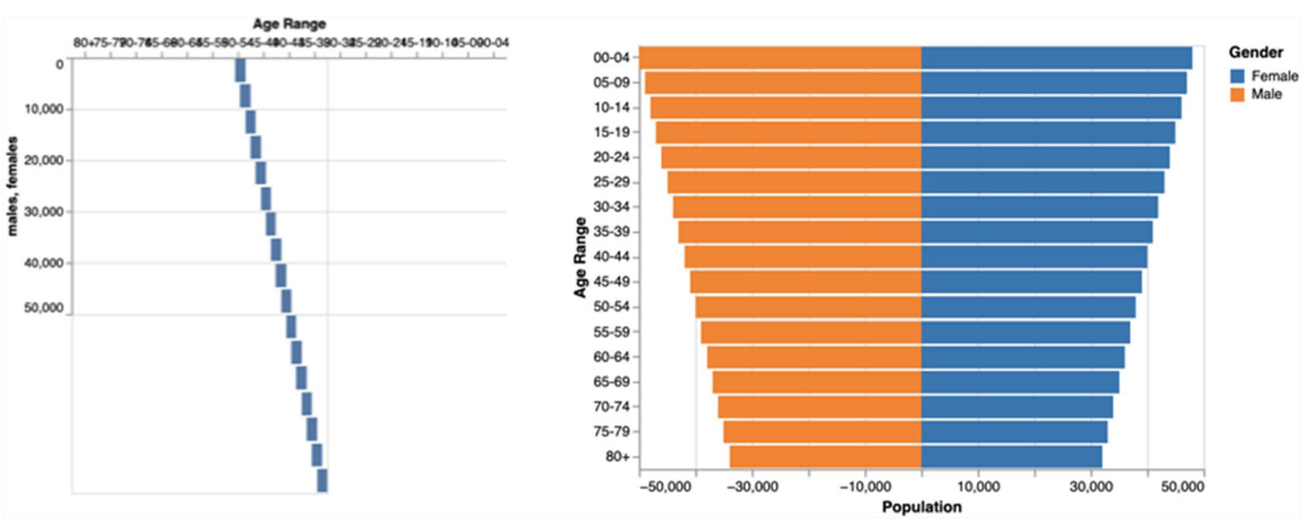

Fig. 12 Comparison of pyramid charts produced via Vega-lite scripts. Gemini (on left) was totally incorrect. GPT-4o's output was visually correct, the labels were not proper (on right)

To address this limitation, we went beyond automated binary questioning and incorporated manual analysis as a crucial step in our methodology. This involved developing custom, non-binary questions aimed at probing deeper into the visual reasoning abilities of the models. By using questions that require more elaborate responses, we aim

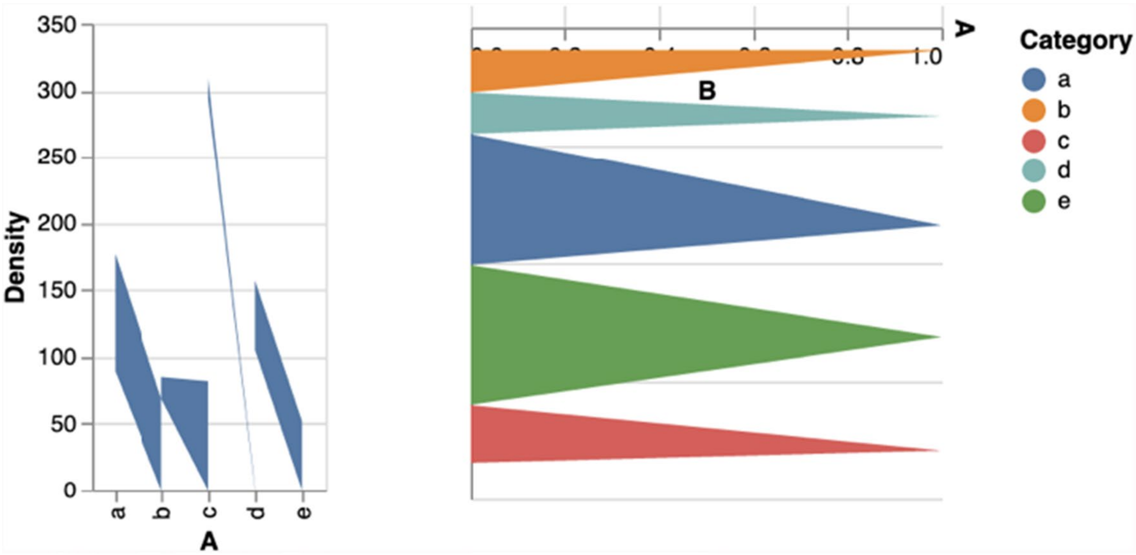

Fig. 13 Comparison of violin charts produced via Vega-lite scripts. Both GPT-4o and Gemini could not produce Violin charts

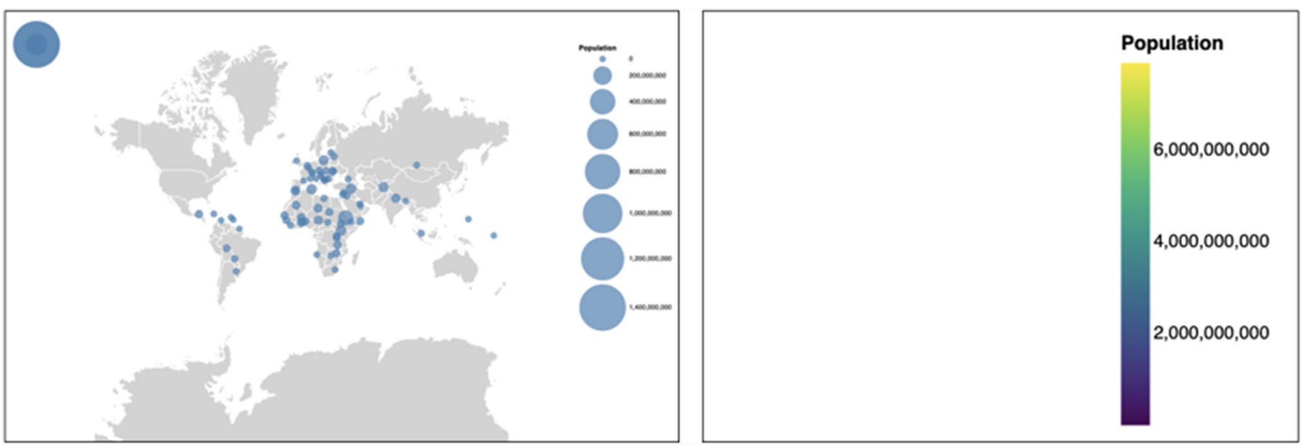

Fig. 14 Comparison of locator maps produced via Vega-lite scripts. GPT-4o's output was correct (on left). Gemini couldn't produce the map (on right)

to provide a more complete and accurate analysis of whether the models truly understand the underlying charts rather than merely guessing the answers.

4.4 Data for manual analysis

For the manual analysis, we have selected 20 random charts for each of chart type in the FigureQA dataset, namely,

1. Vertical Bar charts
2. Horizontal Bar charts
3. Line charts
4. Pie Charts

Since the images were randomly chosen, we hope the chosen charts have different levels of complexity. We have introduced new non-binary questions for each chart type that are useful to evaluate the level of understanding a model has of a chart. For each of the charts, we created a group of questions whose answers we thought would be a better indicator

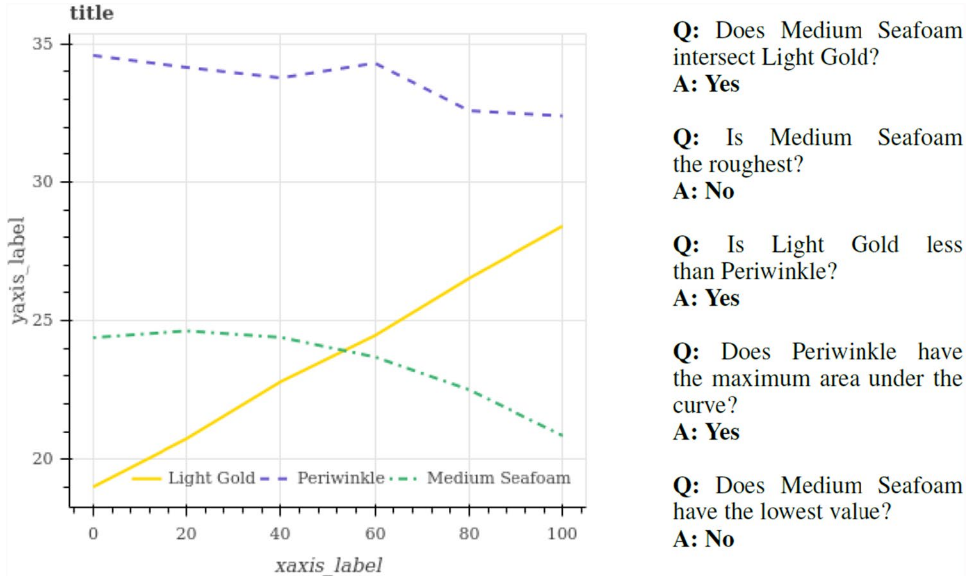

Fig. 15 Sample line plot figure with the corresponding question-answer pairs in FigureQA

Table 4 Question templates and applicable figure types in FigureQA

Index	Question	Figure types
1	Is X the minimum?	Bar, pie
2	Is X the maximum?	Bar, pie
3	Is X the low median?	Bar, pie
4	Is X the high median?	Bar, pie
5	Is X less than Y?	Bar, pie
6	Is X greater than Y?	bar, pie
7	Does X have the minimum area under the curve?	Line
8	Does X have the maximum area under the curve?	Line
9	Is X the smoothest?	Line
10	Is X the roughest?	Line
11	Does X have the lowest value?	Line
12	Does X have the highest value?	Line
13	Is X less than Y?	Line
14	Is X greater than Y?	Line
15	Does X intersect Y?	Line

Table 5 Comparison of Performance Metrics between LLMs to answer Yes-No (binary) questions

Metric	Gemini-1.5-pro	GPT-4o	Claude 3 Opus
Total Questions	1342	1342	1342
Total Correct Answers	863	886	733
Total Wrong Answers	479	456	609
Accuracy (%)	64.31%	66.02%	54.61%

of the understanding of those charts by LLMs. A subset of these questions was selected to evaluate the LLMs for that particular chart. Examples of these questions are as follows:

- Vertical bar graphs:
  - How many bars are there?
  - What are their colors?

- Which color has the maximum/minimum value?
- Is the bar with color X greater/larger than the bar with color Y?
- Is the value for the bar with color X the same as that of the bar with color Y?
- What is the value of the bar with color X?
- Which color bars are greater/larger than the bar with color X?
- Are there X bars? If yes, which color bar has the height that is in the middle?
- Do bars with colors X and Y have the same value?
- How many shades of X color are there in the chart?

- Horizontal bar graphs:
  - How many bars are there?
  - What are their colors?
  - Which color has the maximum/minimum value?
  - What is the value of the bar with color X?
  - Is the value for the bar with color X the same as that of the bar with color Y?
  - What are the values of bars with colors X and Y, respectively?
  - Is there a bar with the value 'X'? If yes, what is the color of that bar?
  - Which bar is bigger among colors X and Y?
  - Which color bars are bigger than the bar with color X?
- Line charts:
  - How many lines are there?
  - What are their colors?
  - What is the maximum/minimum value on the X-axis/Y-axis?
  - Do lines with colors X and Y intersect?
  - How many dotted/non-dotted lines are there?
  - Do any of the lines intersect?
  - Is there a straight line/point where all the lines intersect?
  - Do any of the dotted and non-dotted lines intersect?
  - How many intersection points are there?
  - Does the non-dotted line intersect all the dotted lines? (Not necessarily at the same point)
  - What is the maximum number of lines that intersect at a single point?
  - Are there any straight dotted lines?
  - Is there an intersection point in the chart?
- Pie charts:
  - How many pies are there?
  - What are their colors?
  - Which color pie has the largest area?
  - Does the lower half of the pie look like a pyramid?
  - What colors surround the pie with the color X?

4.5 Manual analysis results

For each LLM, for each chart, we uploaded the image and then asked all the questions. All answers were checked against the correct answers for the same questions. This, along with the inter-LLM comparison, will inherently compare the LLMs' performances with a human baseline. We have also compared the models' performances with and without a simple system prompt given below: *Analyse the following chart carefully and answer the following questions correctly.*

Table 6 Performance comparison of different models for vertical bar graphs

	Gemini 1.5 Pro	GPT-4o	Claude 3 Opus
Images for which all questions were answered correctly without system prompt	55%	55%	15%
Images for which all questions were answered correctly with system prompt	60%	70%	15%
Questions answered without system prompt	86.2%	91.1%	50.9%
Questions answered correctly with system prompt	88.2%	94.1%	54.9%

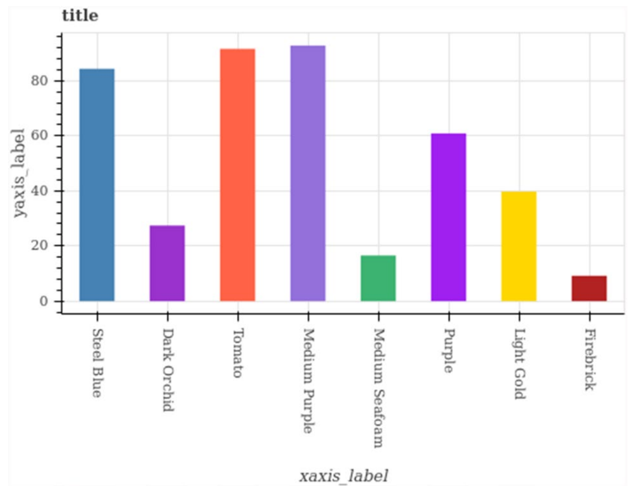

Fig. 16 Example of a chart with close bars where all the models struggled

4.5.1 Vertical bar graphs

The performance comparison of the different LLMs for vertical bar graphs is shown in Table 6. Some observations from this analysis are as follows.

- Claude 3 did not use the actual given color names for most of these charts. If it had some association of a color with a name, it used those.
- All three models responded incorrectly when the heights of the bars were close (for example, Fig. 16).
- In some of the charts, Claude 3 Opus hallucinated and stated things that are not present anywhere in the image. An example is shown in Fig. 17.[3]
- System prompts significantly improved the performance of Gemini–1.5-Pro and GPT-4o but did not have much effect on the performance of Claude 3 Opus.
- From our limited testing, it seems that GPT-4o is the best model for tasks involving such charts.

4.5.2 Horizontal bar graphs

The performance comparison of the different LLMs for horizontal bar graphs is shown in Table 7. Some observations from this analysis are as follows.

- Again, when the bar lengths are close, all the models struggle in determining the larger/smaller bar (for example, Fig. 18).

[3] Hallucinations [40] are events in which LLMs produce outputs that are coherent and grammatically correct, but factually incorrect or nonsensical.

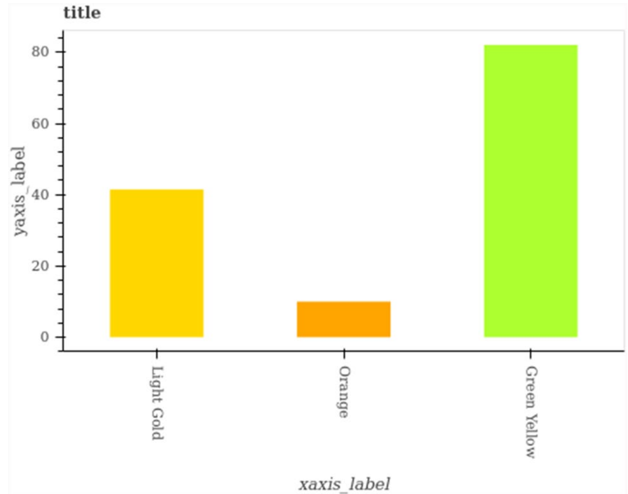

Fig. 17 Claude 3 Opus hallucinated and answered "South Africa" to the question "Which color has min value?" on this chart

Table 7 Performance comparison of different models for horizontal bar graphs

	Gemini 1.5 Pro	GPT-4o	Claude 3 Opus
Images for which all questions were answered correctly without system prompt.	55%	50%	25%
Images for which all questions were answered correctly with system prompt.	70%	55%	30%
Questions answered without system prompt.	86.7%	84.6%	64.2%
Questions answered correctly with system prompt.	93.8%	86.7%	68.3%

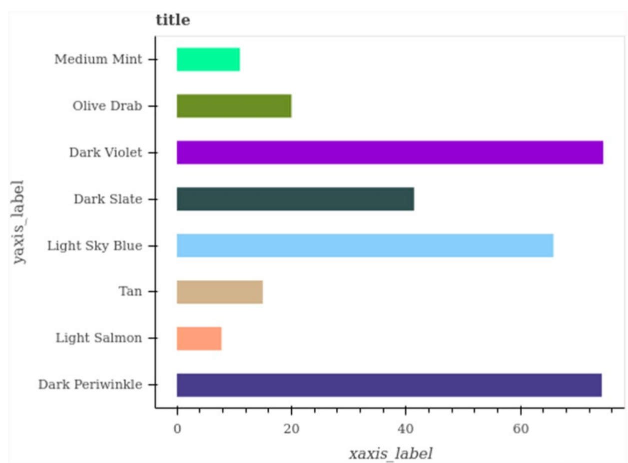

Fig. 18 All three models could not determine the color of the longest bar

- The system prompts significantly improved the performance of all three models, especially the Gemini–1.5-pro.
- All models were not able to determine the actual bar length. Gemini is comparatively better, while Claude 3 almost always got the length wrong (for example, Fig. 19).
- Claude did not use the given color names.
- From our limited testing, it seems that Gemini–1.5-Pro is the best model for tasks involving such charts.

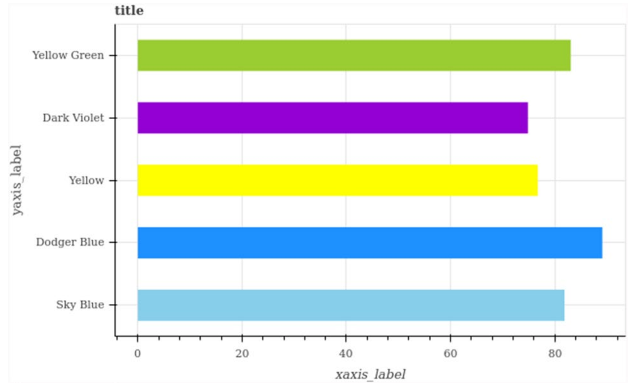

Fig. 19 Claude 3 Opus and GPT-4o could not determine the length of yellow bar

Table 8 Performance comparison of different models for line charts

	Gemini 1.5 Pro	GPT-4o	Claude 3 Opus
Images for which all questions were answered correctly without system prompt.	30%	15%	40%
Images for which all questions were answered correctly with system prompt.	40%	20%	45%
Questions answered without system prompt.	78.5%	83.1%	84.1%
Questions answered correctly with system prompt.	84.1%	84.1%	88.7%

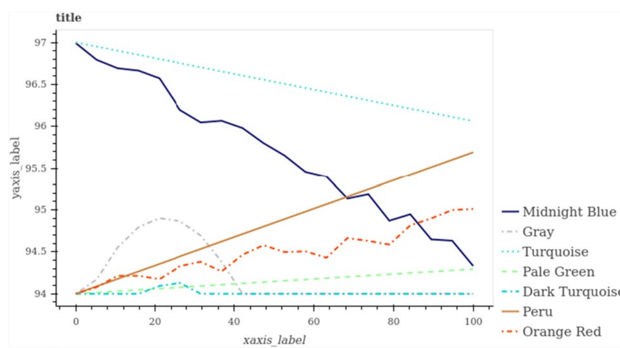

Fig. 20 Gemini–1.5-Pro did not find any dotted line in this chart

4.5.3 Line charts

The performance comparison of the different LLMs for the Line charts is shown in Table 8. Some observations from this analysis are as follows.

- Sometimes Gemini–1.5-Pro did not recognize the dotted lines at all. This always happened when there is a mixture of dotted and non-dotted lines (for example, Fig. 20).
- For the questions involving the counting of lines and points, the models gave different responses when asked again 15% of the time (see, for example, Fig. 21).
- All models performed poorly on line charts, as compared to other types of chart.
- Claude 3 Opus is the best performer in our limited testing.
- The system prompt significantly improved the performance of Gemini–1.5-Pro.

4.5.4 Pie charts

The performance comparison of the different LLMs for pie charts is shown in Table 9. Some observations from this analysis are as follows.

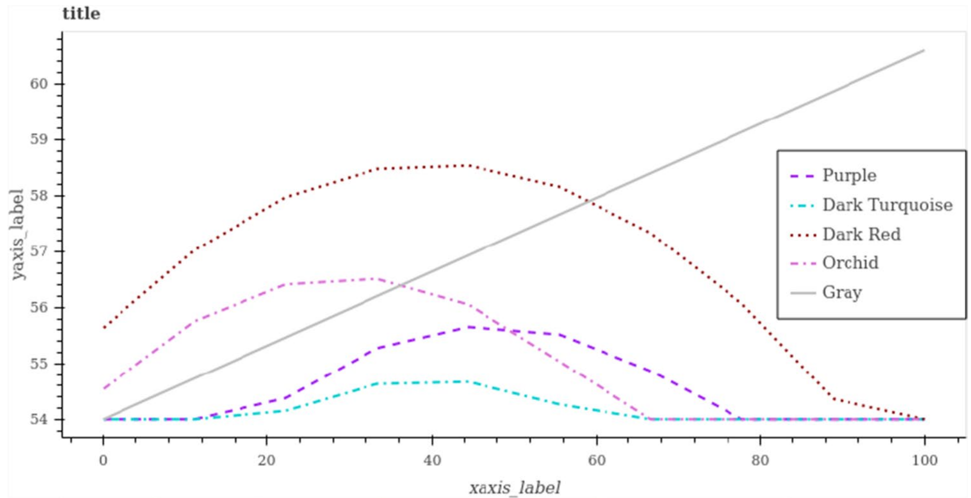

Fig. 21 When asked about the number of dotted lines, the models gave different answers in different invocations

Table 9 Performance comparison of different models for pie Charts

	Gemini 1.5 Pro	GPT-4o	Claude 3 Opus
Images for which all questions were answered correctly without system prompt.	75%	85%	55%
Images for which all questions were answered correctly with system prompt.	80%	85%	65%
Questions answered without system prompt.	90.3%	95.1%	82.2%
Questions answered correctly with system prompt.	91.9%	95.1%	85.4%

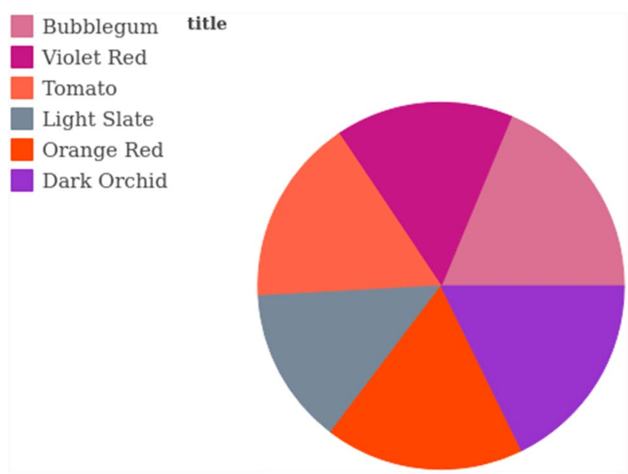

Fig. 22 Only Gemini–1.5-pro answered correctly the question "Which color pie has the most area?"

- Most of the models performed much better on pie charts as compared to other chart types. The best performing model was GPT-4o.
- The system prompts did not help much in this, as the accuracy was already pretty high.
- For the most part, Claude 3 Opus used its own color names and not the ones given in the chart.
- GPT-4o and Claude 3 Opus again got answers to the questions that involved close things. wrong (for example, Fig. 22 on "Which color pie has the most area?").

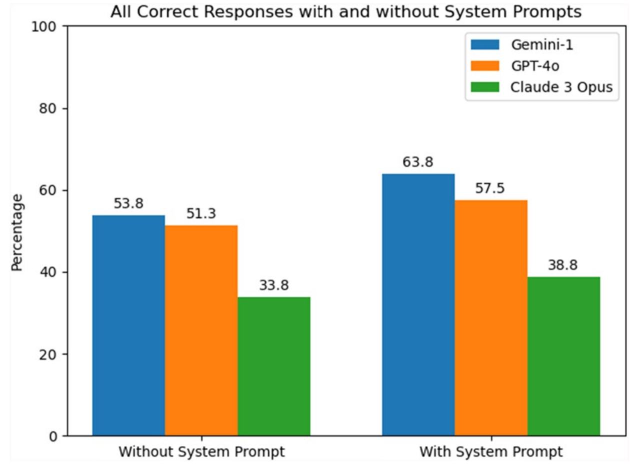

Fig. 23 % of images with all questions answered correctly

4.5.5 Overall Statistics:

We also calculated the results in all four types of graphs. Figure 23 shows the percentage of images for which all the questions were correctly answered. On the other hand, Fig. 24 shows the percentage of questions answered correctly. GPT-4o and Gemini performance were almost identical and better than Claude's performance.

From the analysis, we gained several key insights as follows:

- Performance Across Different Chart Types
  - The performance of LLMs varied significantly among different types of charts. For example, with system prompts GPT-4o had 85% accuracy on pie charts and a mere 20% accuracy on line charts. This suggests that certain visualization types may be easier for LLMs to interpret than others.
  - All of the models performed very poorly on the line charts. This might be because of the presence of dotted lines, which might be treated as some kind of noise by the models. Most of them got the questions related to the dotted lines wrong.
  - All models struggled with identifying relationships between close boundaries and lengths of shapes.
- Impact of System Prompts
  - In all the cases, the use of system prompts improved the performance of the models. The amount of improvement varied with the models and chart types.
  - The accuracy of Gemini–1.5-Pro improved significantly with the use of system prompts.

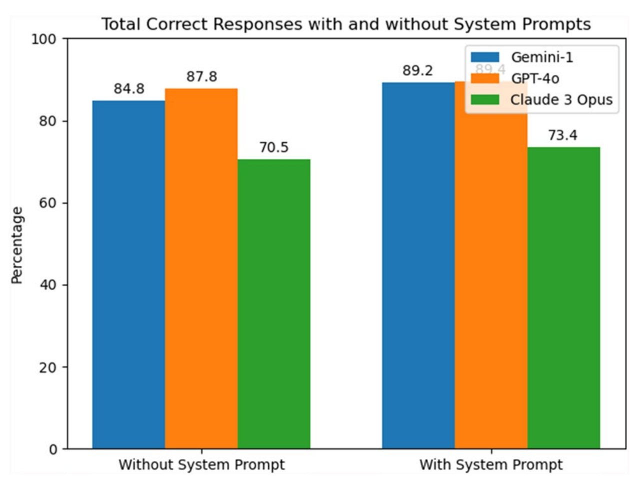

Fig. 24 % of questions answered correctly

- The fact that in all the cases we had improvements highlights the importance of context and guidance in enhancing model outputs.

5 Conclusion

In this paper, we explored the capabilities of large language models (LLMs) in generating visualizations from natural language commands. Our primary focus was on evaluating the performance of four prominent LLMs: OpenAI's GPT–3.5 and GPT-4o, Google's Gemini–1.5-pro, and Anthropics's Claude 3 Opus in creating various types of chart using Python and Vega-Lite scripts. Our evaluation shows that while most LLMs can generate simpler charts, many have difficulties to generate correct code for more complex charts and providing non-trivial customizations to charts. Moreover, we explored the capabilities of 3 LLMs - GPT-4o, Gemini, and Claude - to understand and answer questions about some common information visualizations. We determined that LLMs can have difficulty interpreting some features in charts such as dotted lines as well as distinguishing between bar graphs of similar length.

This paper extends the prior art to explore the capabilities of LLMs for visualization generation and understanding. The findings of our research provide valuable insight into the current state of LLMs in the field of data visualization. The results of this paper can be utilized to address the limitations of LLMs to enable them to generate and understand more sophisticated visualizations in the future.

Some of these limitations highlighted in the paper may be removed in future versions of the LLMs. Some other areas of future work include:

- We want to explore whether more advanced prompting techniques like Chain-of-Thought [41] can improve the results.
- We need to expand the analysis to other types of information visualization - for example, the generation and understanding of visualizations of graphs and trees.
- We also need to create robust evaluation metrics and use more comprehensive datasets that better capture the complexities of real-world information visualizations.
- Combining the capabilities of LLMs and visualization tools to generate interactive visualizations from natural language input is another promising research direction.

Authors contributions All authors Saadiq Rauf Khan, Vinit Chandak, and Sougata Mukherjea contributed equally.

Funding None

Data availability The data sets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Ethics approval and consent to participate Not applicable

Consent for publication Not applicable

Competing interests None

References

1. Tableau. https://www.tableau.com/
2. Matplotlib. https://matplotlib.org/
3. Vega-Lite. https://vega.github.io/vega-lite/
4. Wu A, Wang Y, Shu X, Moritz D, Cui W, Zhang H, Zhang D, Qu H. AI4VIS: survey on artificial intelligence approaches for data visualization. IEEE Transact on Visualizat Comput Graph. 2022;28(12):5049–70.
5. Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, et al. Language models are few-shot learners. In: Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS), 2020;pp. 1877–1901.
6. Chen M, Tworek J, Jun H, Yuan Q, Ponde H, et al. Evaluating large language models trained on code. 2021. arxiv: 2107.03374
7. GPT-4. https://openai.com/index/gpt-4
8. Gemini. https://gemini.google.com
9. Claude. https://claude.ai
10. Liu P, Yuan W, Fu J, Jiang Z, Hayashi H, Neubig G. Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing. ACM Comput Surv. 2023;55(9):1–35.
11. Wang Y, Sun Z, Zhang H, Cui W, Xu K, Ma X, Zhang D. DataShot: automatic generation of fact sheets from tabular data. IEEE Transact on Visualizat & Comput Graph. 2020;26(1):895–905.
12. Dibia V, Demiralp C. Data2Vis: automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. IEEE Comput Graph Applicat. 2019;39(5):33–46.
13. Xiao S, Huang S, Lin Y, Ye Y, Zeng W. Let the chart spark: embedding semantic context into chart with text-to-image generative model. IEEE Transact Visualizat Comput Graph. 2024;30(1):284–94.
14. Luo Y, Qin X, Tang N, Li G, Wang X. DeepEye: creating good data visualizations by keyword search. In: International ACM Conference on Management of Data (SIGMOD); 2018.
15. Apache OpenNLP. http://opennlp.apache.org
16. Narechania A, Srinivasan A, Stasko JT. NL4DV: a toolkit for generating analytic specifications for data visualization from natural language queries. IEEE Transact Visualizat Comput Graph. 2021;27(2):369–79.
17. Manning CD, Surdeanu M, Bauer J, Finkel JR, Bethard S, McClosky D. The Stanford CoreNLP natural language processing toolkit. In: Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations, 2014;pp. 55–60.
18. Shen L, Shen E, Luo Y, Yang X, Hu X, Zhang X, Tai Z, Wang J. Towards natural language interfaces for data visualization: a survey. IEEE Transact Visualizat Comput Graph. 2023;29(6):3121–44.
19. Liu C, Han Y, Jiang R, Yuan X. ADVISor: automatic visualization answer for natural-language question on tabular data. In: IEEE Pacific Visualization Symposium (PacificVis) 2021; (2021)
20. Luo Y, Tang N, Li G, Tang J, Chai C, Qin X. Natural language to visualization by neural machine translation. IEEE Transact Visualizat Comput Graph. 2022;28(1):217–26.
21. Luo Y, Tang N, Li G, Chai C, Li W, Qin X. Synthesizing natural language to visualization (NL2VIS) benchmarks from NL2SQL Benchmarks. In: Proceedings of SIGMOD. 2021.
22. Chen N, Zhang Y, Xu J, Ren K, Yang Y. VisEval: a benchmark for data visualization in the era of large language models. IEEE Transact Visualizat Comput Graph. 2025;31(1):1301–11.
23. Vázquez P-P. Are LLMs ready for visualization? In: IEEE PacificVis 2024 workshop - Vis Meets AI. 2024; 343–352.
24. Li G, Wang X, Aodeng G, Zheng S, Zhang Y, Ou C, Wang S, Liu CH. Visualization generation with large language models: an evaluation. 2024. arxiv: 2401.11255
25. Tian Y, Cui W, Deng D, Yi X, Yang Y, Zhang H, Wu Y. ChartGPT: leveraging LLMs to generate charts from abstract natural language. In: IEEE Vis 2024; (2024)
26. Maddigan P, Susnjak T. Chat2Vis: Generating data visualizations via natural language using chatgpt, codex and GPT-3 large language models. IEEE Access. 2023;11:45181–93.
27. Dibia V. LIDA: a tool for automatic generation of grammar agnostic visualizations and infographics using large language models. In: Proceedings of the 61st annual meeting of the association for computational linguistics (System Demonstrations), 2023; 3: pp. 113–126.
28. Han Y, Zhang C, Chen X, Yang X, Wang Z, Yu G, Fu B, Zhang H. ChartLlama: a multimodal LLM for chart understanding and generation; 2023. arxiv: 2311.16483
29. Ye J, Hu A, Xu H, Ye Q, Yan M, et al. UReader: Universal OCR-free visually-situated language understanding with multimodal large language model. In: findings of the association for computational linguistics: EMNLP 2023; (2023)
30. Meng F, Shao W, Lu Q, Gao P, Zhang K, Qiao Y, Luo P. ChartAssisstant: a Universal chart multimodal language model via chart-to-table pre-training and multitask instruction tuning. In: findings of the association for computational linguistics: ACL 2024; (2024)
31. Kanthara S, Leong RTK, Lin X, Masry A, Thakkar M, Hoque E, Joty SR. Chart-to-text: a large-scale Benchmark for chart summarization. In: Proceedings of the 60th annual meeting of the association for computational linguistics (Volume 1: Long Papers), 2022;pp. 4005–4023.
32. Masry A, Do XL, Tan JQ, Joty S, Hoque E. ChartQA: a benchmark for question answering about charts with visual and logical reasoning. In: findings of the association for computational linguistics: ACL, Dublin, Ireland. 2022.
33. Li S, Tajbakhsh N. SciGraphQA: a large-scale synthetic multi-turn question-answering dataset for scientific graphs; 2023. arxiv: 2308.03349
34. Guan T, Liu F, Wu X, Xian R, Li Z, et al. HallusionBench: an advanced diagnostic suite for Entangled language Hallucination and visual illusion in large vision-language models. In: computer vision and pattern recognition (CVPR). 2024.
35. Xu Z, Du S, Qi Y, Xu C, Yuan C, Guo J. ChartBench: a Benchmark for complex visual reasoning in charts; 2024. arxiv: 2312.15915
36. Bendeck A, Stasko J. An empirical evaluation of the GPT-4 multimodal language model on visualization literacy tasks. IEEE Transact Visualizat Comput Graph. 2025;31(1):1105–15.
37. Lo LY-H, Qu H. How good (Or Bad) are LLMs in detecting misleading visualizations. IEEE Transact Visualizat Comput Graph. 2025;31(1):1116–25.

38. Choe K, Lee C, Lee S, Song J, Cho A, Kim NW, Seo J. Enhancing data literacy on-demand: LLMs as guides for Novices in chart interpretation. In: IEEE Vis 2024; (2024)
39. Kahou SE, Michalski V, Atkinson A, Kadar A, Trischler A, Bengio Y. FigureQA: an annotated figure dataset for visual reasoning. 2018. arxiv: 1710.07300
40. Ji Z, Lee N, Frieske R, Yu T, Su D, Xu Y, Ishii E, Bang YJ, Madotto A, Fung P. Survey of Hallucination in natural language generation. ACM Comput Survey. 2023;55(12):1–38.
41. Wei J, Wang X, Schuurmans D, Bosma M, Chi EH, Xia F, Le Q, Zhou D. Chain of thought prompting elicits reasoning in large language models. Neural Informat Process Syst. 2022;35:24824–37.