Supplementary information for the Plotting Benchmark

Contents

1 Example of the datapoint

(id=22)

{"**plot description**":

" Plot Description: Plot three series from the dataframe on a single graph. Each series ('smiddle', 'slower', 'supper') should be plotted against the 't' column. The three series will be represented as separate lines on the graph, showcasing how each varies over 't'.",

"**plot style**":

" Plot Style Description: The visual representation should clearly distinguish between the three data series by using different colors for each line. The resultant plot should include appropriate axes labels and a legend, if necessary, to ensure clarity in distinguishing between the different data series. The style should be visually appealing and clear, allowing easy comparison of trends across the three series."}

Data description (not included in data point):

The dataframe consists of 200 rows with columns named 't', 'smiddle', 'slower', and 'supper'. The 't' column represents time or a similar numeric axis and the other columns represent different data series which are to be visualized.

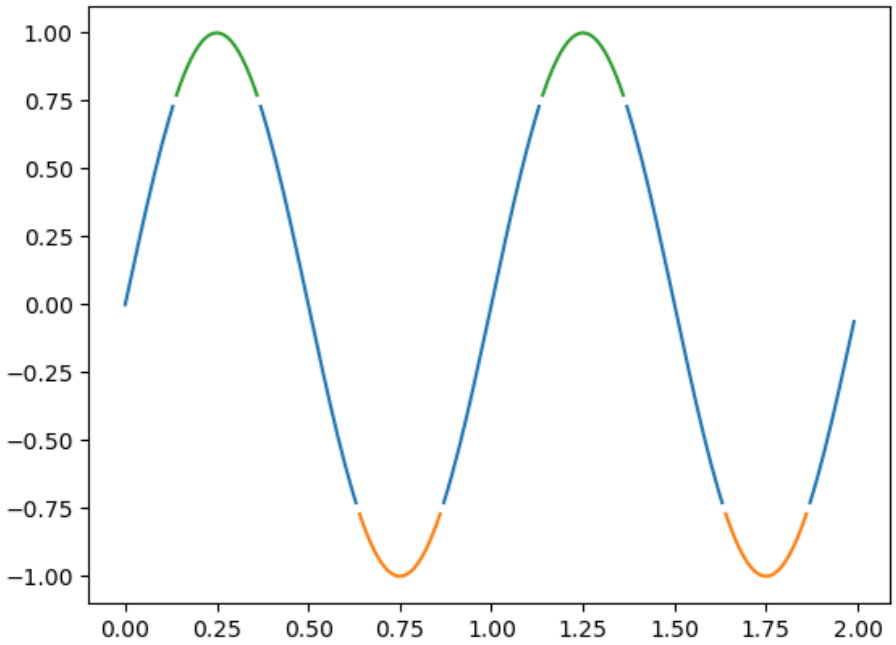

Figure 1: Sample (id=22)

2 Plotting instructions

System prompt

You are a helpful programming assistant proficient in Python, matplotlib and pandas dataframes. All blocks of the Python code must be enclosed in a block “‘python\n SOME CODE“‘. Example for printing \"hello world\": “‘python\n print('hello world')\n“‘

Plot instruct

Write a code to build a plot of dataframe according to following instructions. Write a code that returns plot, not just function declaration.\nLoad df dataframe by single line “‘df = pd.read_csv(\"data.csv\")“‘. DO NOT alter df dataframe columns or add columns. This df dataframe should remain intact.\nDo not write explanations. Just a code enclosed in codeblock. Write important reasoning in comments to the code.\nMake sure that all used libraries and functions are imported.\nUse only visualization libraries listed in the setup section, do not use others.

Setup instruct

Setup. Use Python programming language. Import essential libraries. The essential libraries needed are pandas for managing dataframes and [PLOTLIB] with its subsidiary libraries for plotting. Ensure importing numpy as np and scipy if they are used in the program. DO NOT use or import other visualization libraries. Use latest, not deprecated modules and packages.

Data instruct

Data description. Load df dataframe by single line “‘df = pd.read_csv(\"data.csv\")“‘, DO NOT create it or load. The metadata of the dataframe is following:\n"

The final prompt is constructed as following:

“‘
{Plot instruct}
{Setup instruct}
{Data instruct}
{Data description}
{Plot description}
{Plot stlyle}
“‘

3 Data processing instructions

3.1 Code splitting

Change code, so that all data before plotting is gathered to a single dataframe named "df". Your response MUST contain EXACTLY TWO codeblocks. First for the dataframe construction. Second separate codeblock for the plotting the dataframe.
«CODE»

3.2 Task generation

Write the general TASK to write a code for plotting the given pandas dataframe. Code and dataframe summary is given below. Result of the generated plot image(s) is given below. Split task into four part: 1. Setup (describe language and libs needed for the plot). 2. Some short descriptin of the dataframe. 3. Plot description (omit libraries and specific functions names at this part). 4. Plot style description (omit libraries and specific functions names at this part).
CODE:
«CODE»
Dataframe SUMMARY:
«SUMMARY»
Write the general TASK to write a CODE for plotting the given pandas dataframe (SUMMARY given), to produce plot image (image given at the end). Do not generate dataframe, imply that it is given. Write only task after word TASK. Each part of the task should begin from new line.
«FIGURES»

4 Judge instructions

Visual judge instruction

You are an excellent judge at evaluating visualization plots between a model-generated plot and the ground truth. You will be giving scores on how well it matches the ground truth plot.

The generated plot will be given to you as the first figure.

Another plot will be given to you as the second figure, which is the desired outcome of the user query, meaning it is the ground truth for you to reference.

Please compare the two figures head to head and rate them.

Suppose the second figure has a score of 100, rate the first figure on a scale from 0 to 100.

Scoring should be carried out in the following aspects:

Plot correctness:

compare closely between the generated plot and the ground truth, the more resemblance the generated plot has compared to the ground truth, the higher the score. The score should be proportionate to the resemblance between the two plots. Ignore color matching. If the plots present the same information but are made in different colours, consider them matching. Capture the resemblance of the main idea of the plot.

If the first figure is almost blank, rate it 0.

Only rate the first figure, the second figure is only for reference.

After scoring from the above aspect, please give a final score. Do not write anything else. The final score is preceded by the [FINAL SCORE] token.

For example [FINAL SCORE]: 40"

Task judge instruction

You are an excellent judge at evaluating visualization plots according to the given task. You will be giving scores on how well the plot image matches the ground truth task.

The generated plot will be given to you as an image.

Please score how well the plot matches the task. Score it on a scale from 0 to 100.

Scoring should be carried out in the following aspects:

Task adherence: how the plot corresponds to the task given below (begins from [PLOT TASK] token)

After scoring from the above aspect, please give a final score. Do not write anything else. The final score is preceded by the [FINAL SCORE] token.

For example [FINAL SCORE]: 40

[PLOT TASK]:

«PLOT TASK»

5 Statistical analysis

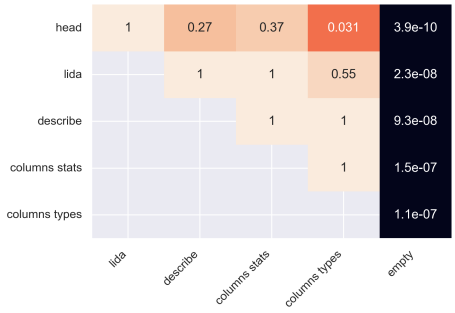

Figure 2: Table. Corrected p-values for statistical significance of differences between DataFrame descriptors score sets.

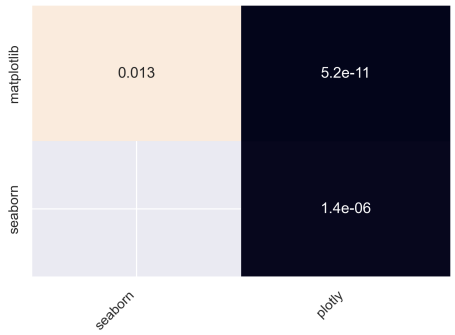

Figure 3: Table. Corrected p-values for statistical significance of differences between libraries score sets.

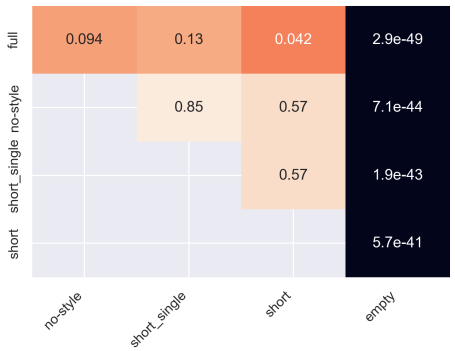

Figure 4: Corrected p-values for statistical significance of differences between the task variants score sets.

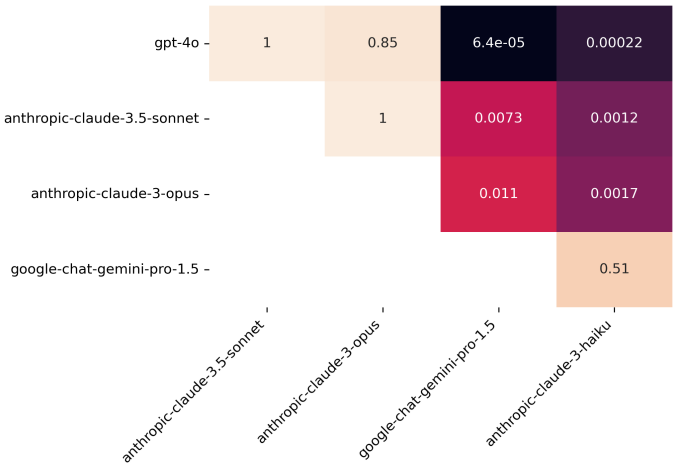

Figure 5: Table. Corrected p-values for statistical significance of differences between models' score sets.

6 CodeBERT score

Score cross-plots demonstrate that the CodeBERT are uncorrelated with both task and visual scores. CodeBERT is almost evenly centred around a 0.8-0.85 score.

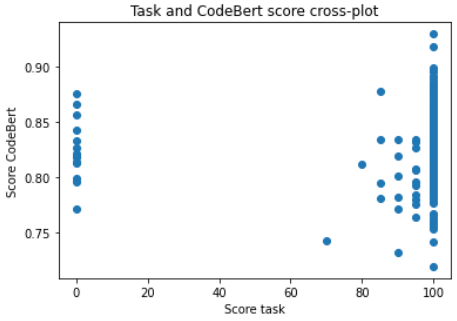

Figure 6: Task and CodeBERT score cross-plot

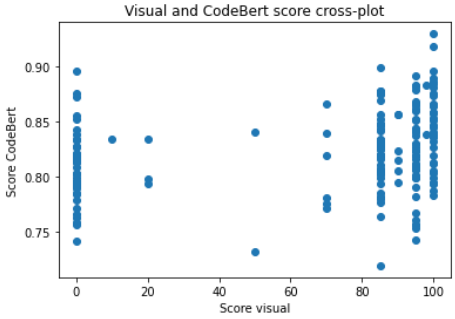

Figure 7: Visual and CodeBERT score cross-plot

7 Dataframe descriptors

To set up model prompt construction, we used a benchmark to determine the best way to include the DF description. We utilized the Matplotlib and OpenAI GPT-4o models with the following descriptors: (i) Columns names and statistics; (ii) Columns names and types (ColTypes). (iii) LIDA [**?**] (columns name, type, samples); (iv) ColTypes + describe() method (stats for each column); (v) ColTypes + head(5) method (samples for each column); (vi) Control - no any DataFrame information. The results are summarized in Table 1.

The LIDA and head() methods perform best, but the performance differences with the other methods are not significant. Including raw data samples enhances performance, setting these methods apart. However, LIDA's descriptions are 2.5 times longer than the head's and comprise more than half the model's average input length. Without any DF description, 10% of the code is inexecutable, with significantly lower scores, though still relatively high, suggesting that the plotting instructions are quite strict. Future experiments will use the head descriptor due to its high score and shorter task length.

DataFrame descriptor	Task length (symbols)	Incorrect code, %	Mean score visual/task	good ($\geq$75) visual/task
Head	900	1.8	75/89	0.69/0.91
LIDA	2460	2.3	73/87	0.68/0.89
Describe	1490	2.3	74/86	0.69/0.88
Columns statistics	980	2.5	72/86	0.67/0.87
Columns types	230	2.2	72/85	0.66/0.86
No data description	0	10.6	59/69	0.55/0.69

Table 1: Benchmark scores of the DataFrame descriptors.

Example of dataframe descriptions for datapoint (id=22).

data.csv:

```
      t   smiddle  slower  supper
0  0.00  0.000000     NaN     NaN
1  0.01  0.062791     NaN     NaN
...
14  0.14       NaN     NaN  0.770513
15  0.15       NaN     NaN  0.809017
...
164  1.64       NaN -0.770513     NaN
165  1.65       NaN -0.809017     NaN
...
```

head:

```
DataFrame with 4 columns and 200 rows
Column names and types: t: float64, smiddle: float64, slower: float64, supper: float64
Head:
{"t":{"0":0.0,"1":0.01,"2":0.02,"3":0.03,"4":0.04},"smiddle":{"0":0.0,"1":0.0627905195,"2"
```

LIDA:


```
[
  {
    "column": "t",
    "properties": {
      "dtype": "number",
      "std": 0.5787918451395113,
      "min": 0.0,
      "max": 1.99,
      "samples": [
        0.95,
        0.15,
        0.3
      ],
      "num_unique_values": 200
    }
  },
  {
    "column": "smiddle",
    "properties": {
      "dtype": "number",
      "std": 0.4575352482034556,
      "min": -0.7289686274214118,
      "max": 0.7289686274214119,
      "samples": [
        -0.5877852522924728,
        0.1873813145857242,
        0.2486898871648555
      ],
      "num_unique_values": 100
    }
  },
  {
    "column": "slower",
    "properties": {
      "dtype": "number",
      "std": 0.07546722616007479,
      "min": -1.0,
      "max": -0.7705132427757886,
      "samples": [
        -0.9048270524660196,
        -0.7705132427757896,
        -0.9822872507286886
      ],
      "num_unique_values": 34
    }
  },
  {
```

```
    "column": "supper",
    "properties": {
      "dtype": "number",
      "std": 0.07546722616007483,
      "min": 0.7705132427757886,
      "max": 1.0,
      "samples": [
        0.9822872507286886,
        0.8090169943749478,
        0.7705132427757893
      ],
      "num_unique_values": 25
    }
  }
]
```


describe:


```
DataFrame with 4 columns and 200 rows
Column names and types: t: float64, smiddle: float64, slower: float64, supper: float64
Columns stats:
{"t":{"count":200.0,"mean":0.995,"std":0.5787918451,"min":0.0,"25%":0.4975,"50%":0.995,"75
```


columns statistics:


```
Number of rows in DataFrame: 200
DataFrame has the following columns:
t of type float. Count: 200, Mean: 0.995, Std. Deviation: 0.578792, Min: 0.0, Max: 1.99
smiddle of type float. Count: 108, Mean: 1.74757e-17, Std. Deviation: 0.457535, Min: -0.72
slower of type float. Count: 46, Mean: -0.915377, Std. Deviation: 0.0754672, Min: -1.0, Ma
supper of type float. Count: 46, Mean: 0.915377, Std. Deviation: 0.0754672, Min: 0.770513,
```


columns type:


```
DataFrame with 4 columns and 200 rows
Column names and types: t: float64, smiddle: float64, slower: float64, supper: float64
```