Computer-Assisted Heuristic Evaluation of Data Visualization

Ying Zhu[1](✉) and Julia A. Gumieniak[2]

[1] Georgia State University, Atlanta, USA
yzhu@gsu.edu

[2] Binghamton University, Binghamton, USA
jgumien1@binghamton.edu

Abstract. Heuristic evaluation has been an important part of data visualization. Many heuristic rules and guidelines for evaluating data visualization have been proposed and reviewed. However, applying heuristic evaluation in practice is not trivial. First, the heuristic rules are discussed in different publications across different disciplines. There is no central repository of heuristic rules for data visualization. There are no consistent guidelines on how to apply them. Second, it is difficult to find multiple experts who are knowledgeable about the heuristic rules, their pitfalls, and counterpoints. To address this issue, we present a computer-assisted heuristic evaluation method for data visualization. Based on this method, we developed a Python-based tool for evaluating plots created by the visualization tool Plotly. Recent advances in declarative data visualization libraries have made it feasible to create such a tool. By providing advice, critiques, and recommendations, this tool serves as a knowledgeable virtual assistant to help data visualization developers evaluate their visualizations as they code.

Keywords: Data visualization · Evaluation · Heuristic rules

1 Introduction

Evaluation is an important part of data visualization [11,18,21], and heuristic evaluation is one of several commonly used evaluation methods. In heuristic evaluation, a small group of evaluators review and critique a data visualization plot based on heuristic rules or guidelines. The heuristic rules and guidelines are proposed by data visualization experts based on their experience. Many heuristic rules have been proposed, tested, and challenged [5,7,10,12–15,17,19,22,28,30–32,36,37,39,41,42,44–46,48]. Heuristic evaluation is a type of formative evaluation that can guide visualization developers to improve visualization design, such as visual encoding and interactions [24]. In addition, some researchers have argued that visualization criticism can be used to great effect in visualization teaching and research [10,20].

Supported in part by NSF Award #1852516.

In theory, heuristic evaluation is considered easy to learn and apply and requires little time and other resources [15], but applying heuristic rules in practice is not trivial. First, the heuristic rules are presented in many different research papers and books, often across different disciplines such as computer science, statistics, psychology, and other fields [28,29]. There is no well-curated central repository of these heuristic rules.

Second, heuristic evaluation is generally difficult for a single individual [25,27]. Ideally, heuristic evaluations should be conducted by a small number of experts who are knowledgeable about the heuristic rules as well as the pitfalls and contexts of these rules [26]. In reality, it is not easy to find such an expert, let alone several of them. Therefore, data visualization developers are usually responsible for evaluating their own data visualizations without external help. In addition, data visualization developers generally lack knowledge about evaluating data visualizations because they work in a wide range of fields (e.g., business, physical science, social science, economy, medicine), and most of them are not formally trained in data visualization. As a result, there are many poorly designed data visualizations on the web, in various publications, and in student works.

To address this issue, we propose a computer-assisted heuristic evaluation method and tool to help data visualization developers conduct heuristic evaluations to improve their designs (Fig. 1). This method is made feasible by the recent advances in declarative data visualization libraries [16,33–35], such as Plotly [3], Altair [1], and d3.js [2,9]. In a declarative data visualization library (e.g., Plotly or Altair), plots are stored in a JSON file (Fig. 3), which can be processed by the proposed computer-assisted heuristic evaluation tool. Based on the characteristics of each plot, the evaluation tool will retrieve a set of internally stored heuristic rules and use them to evaluate the plot (Fig. 1). Since declarative tools like Plotly supports a wide variety of data visualizations, such as statistical charts, maps, graphs, parallel coordinates, and 3D plots, our evaluation tool can be used to evaluate many types of data visualizations.

This tool can provide two types of assistance. For heuristic rules that are too abstract or sophisticated for automatic checking, the tool will display these rules to help the developer conduct heuristic evaluation. In this case, the tool serves as a knowledgeable virtual assistant that reminds developers of the heuristic rules and reduces their mental workload. For heuristic rules that are specifically defined, the proposed tool can automatically check the visualization design and provide specific warnings and recommendations.

Our target audience is data visualization developers who create plots through coding. This API-based tool is easy to use and fully integrated with the current data visualization programming environments such as Jupyter Notebook and Google Colab. For example, a developer only needs to import a Python library and make one API call to start the evaluation. It is particularly useful for individual data visualization developers conducting heuristic evaluations on their own. As far as we know, this is the first computer-based heuristic evaluation tool for data visualization.

2 Background and Related Work

Heuristic evaluation was first proposed by Jacob Nielsen [25–27] for usability inspection of user interfaces, and it has become popular in the field of human-computer interaction (HCI). According to Nielsen and Molich [27], "Heuristic evaluation is an informal method of usability analysis where a number of evaluators are presented with an interface design and asked to comment on it." Their experiments showed that individual evaluators were less effective than a small group of evaluators in finding usability issues [27]. Nielsen [26] also showed that usability specialists were better than non-specialists at performing the heuristic evaluation. Our work directly addresses the issues identified by Nielsen. Our computer-assisted heuristic evaluation tool is designed to help individual data visualization developers who are not experts in data visualization evaluation.

The standard evaluation method for data visualization is controlled user studies, but such studies are complex and time-consuming [4,23]. Many researchers have advocated using heuristic evaluation as a valuable alternative or supplement to the controlled user studies. Tory and Moller [40] argued that expert review could generate valuable feedback on visualization tools, and they recommended developing heuristics based on visualization guidelines and usability guidelines. Munzner [24] proposed a nested model for visualization design and validation, and classified heuristic evaluation as a type of formative evaluation. She further pointed out that heuristic evaluation and expert review are helpful for improving the design of visual encoding and interaction techniques. Lam et al. [21] identified seven guiding scenarios for information visualization evaluation. They pointed out that heuristic evaluation is a useful tool for the design stage and for evaluating collaborative data analysis. Kosara, et al. [20] and Brath and Banissi [10] argued that the visualization design and education could benefit from using more and better critiques. All the previous works assumed that the heuristic evaluations were carried out by human evaluators. However, our work showed that a computer program could be a virtual evaluator for some heuristic evaluation tasks.

Many heuristic rules and guidelines have been proposed for data visualization. Seminal works by Bertin [7], Cleveland and McGill [13], Chambers, et al. [12], Tufte [41,42], and Shneiderman [38] are highly influential in the data visualization field. In 1990s, Senay and Ignatius [37] collected many heuristic rules and principles for the purpose of developing a visualization tool assistant to help scientists and engineers in selecting and creating effective data visualizations. Many scholars continued to develop, analyze, and classify heuristic rules for data visualization [5,6,8,14,15,28,30–32,36,39,44–46,48].

Some researchers have tried to develop software tools to assist the evaluation of data visualizations [4,23]. However, these tools were primarily for assisting user studies, not heuristic analysis. Our evaluation assistant tool is different from the previous work in that our tool focuses on heuristic analysis through code and data analysis.

3 Methodology

3.1 Overview

Figure 1 shows the workflow of computer-assisted heuristic evaluation. A data visualization developer creates a visualization plot using a declarative visualization library, such as Plotly, Altair, or d3.js. The plot is internally stored in a JSON file (see Fig. 3), which is passed to the heuristic evaluation program via a single API call. The heuristic evaluation program processes the JSON file and analyzes the plots based on selected heuristic rules. The outputs of the evaluation program are a set of rules, analyses, warnings, and recommendations. Based on this information, the developer continues to revise and evaluate the visualization plots.

The heuristic evaluation library can be loaded as an external package in a standard programming environment such as Jupyter Notebook, Google Colab, or Observable. The developer only needs to make one API call to start the heuristic evaluation. Therefore, the evaluation tool is easy to use and integrates seamlessly with the current visualization programming practice. This workflow is similar to a writer using a tool like Grammarly to help detect grammar errors and improve the quality of writing.

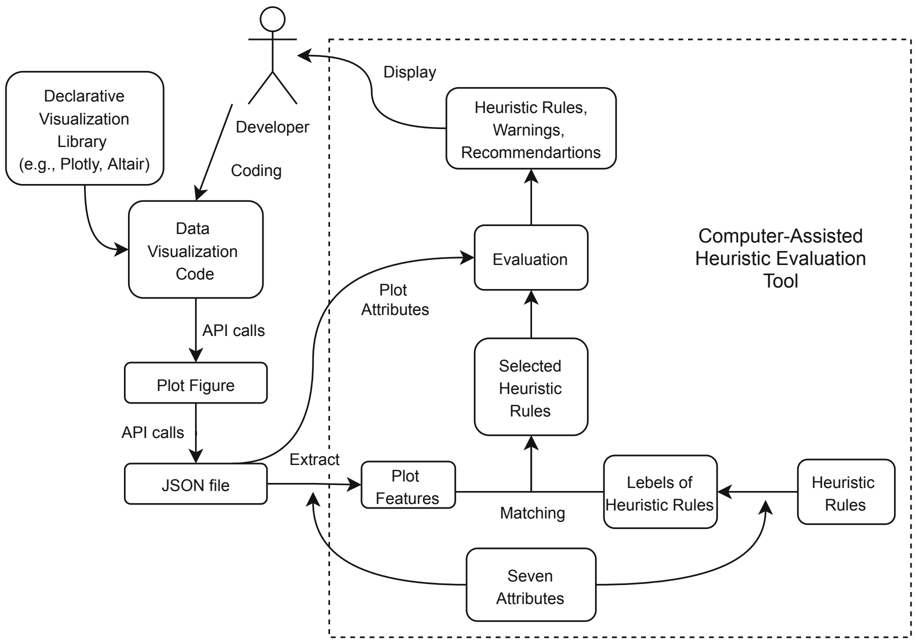


Fig. 1. Computer-assisted heuristic evaluation of data visualization

3.2 Declarative Visualization Libraries

Declarative programming is a programming paradigm in which a developer specifies the logic of a computation without describing its control flow. On the other hand, in imperative programming, a developer implements the control flow to create the desired outcome. One of the best-known examples of declarative programming is the JavaScript library React.

Data visualization libraries can also be divided into imperative libraries and declarative libraries. Imperative libraries include Matplotlib, Seaborn, Bokeh, R/ggplot, Google Charts, etc. Declarative libraries include Plotly [3], Altair [1], d3.js [2], etc. Altair and d3.js are based on the declarative language and visualization grammar research by Heer's lab [16,33–35]. Their visualization grammars Vega and Vega Lite provide a declarative JSON syntax to create an expressive range of data visualizations. The Plotly library is based on d3.js and uses a similar JSON syntax to store its plots. Plotly also includes a set of imperative API called Plotly Express. Therefore, Plotly developers can choose to use either the declarative Plotly API, the imperative Plotly Express API, or both. In this paper, we will use Plotly in our examples.

In Plotly, a developer creates a plot figure by assigning values to some attributes of a JSON data structure, such as specifying chart type, mapping data variables to X and Y axes, adding labels, etc. Many more attributes are filled with default values. The internal Plotly rendering engine processes the JSON file to draw the plot figure. A developer can also create a figure by calling the imperative Plotly Express APIs in a way similar to that of Matplotlib or Seaborn. Even if the figure is created with Plotly Express, the figure is still expressed in a JSON data structure and can be retrieved with a simple API call.

Declarative libraries such as Plotly and Altair make it feasible to analyze data visualization plots in their entirety. Every detail of the plot is stored in the JSON file and can be retrieved and analyzed based on certain heuristic rules. In contrast, it is much more difficult to analyze a plot created by an imperative visualization library (such as Matplotlib or Seaborn) because one has to write a static code analyzer to process the API calls in the source code and the many default attributes of the plot are difficult to locate. Therefore, the recent advances in declarative visualization libraries provide an opportunity to apply certain heuristic evaluations algorithmically.

3.3 Selecting and Classifying Heuristic Rules

We collected over one hundred heuristic rules and guidelines by searching through various academic papers, books, and websites [5–8,12–15,28,30–32,36–39,41,42,44–46,48]. While some rules are specific, many rules are loosely defined and difficult to be checked by a computer program. Therefore, we divide the rules into three categories: *difficult-to-check, advice, and automatic-check*.

The rules in the *difficult-to-check* category are considered too difficult to be checked by a computer program for various reasons, such as too general, too abstract, or too short. For example, the rules "expose uncertainty" [5], "relate"

[38], "flexibility" [15] would fall into this category. We do not include these rules in our program.

The rules in the *advice* category are more specific but still not specific enough to be checked automatically. In this category, a rule might be associated with a specific plot (e.g., a line chart) or a specific feature of the plot (e.g., axis), but it is difficult to conduct a specific evaluation. This type of heuristic rules will be presented to the developer so the developer can conduct the heuristic evaluation herself. In this case, our program would serve as a virtual assistant that advises the developer with heuristic rules relevant to the current plot.

Due to the space limit, we will only list a few examples of the rules in the *advice* category here.

– Maximizing data-ink ratio, but within reason.
– Remove the extraneous ink.
– Choose colormaps that match the nature of the data

The rules in the *automatic-check* category are mostly low-level, specific rules that can be automatically checked by a program. Table 1 shows 20 rules in the *Automatic-Check* category. Additional rules are implemented in our program, and we continue to add more rules.

3.4 Codifying Heuristic Rules

Given a particular plot, how does our program select the relevant heuristic rules for evaluation? We have developed a novel mechanism to match heuristic rules with visualization plots. The basic idea is to label each heuristic rule based on seven attributes. Similarly, each plot is also labeled based on the seven attributes. The heuristic rules with the same or similar labels as the plot are selected for evaluation. We will describe this mechanism in this section and the next two sections.

Each rule in the *advice* and *automatic-check* category is classified based on its connections to the following seven attributes: visual frames, visual structures, visual unities, visual primitives, labeling, interaction, and data attributes. A visualization plot can also be classified based on these attributes. The following list shows the attributes and their possible values.

– Visual Frames: single or multiple visual structures
– Visual Structures: scatter plot, line plot, bar plot, pie chart, map, graph, and other chart types supported by the visualization libraries
– Visual Unities: points, lines, 2D shapes, texts, tooltips, etc.
– Visual Primitives: X or Y position, shape, size, color, and orientation
– Labeling: chart title, X or Y-axis label, ticks on axes, labels, legend, background color, and other style or layout features supported by the visualization libraries
– Interaction: tooltip, zoom, pan, filter, etc.
– Data Attributes: range, dimension, multivariant, categorical, ordinal, nominal, numerical, continuous, etc.

Table 1. Examples of heuristic rules and their classifications

Heuristic rules	Classification
A plot should have a title	Labeling (title)
Each axis should have a label	Visual Primitives (color)
Avoid using more than four different colors	Visual Primitives (color)
Avoid using color to encode ordinal data	Visual Primitives (color)
Local contrast affects color and gray perception	Visual Primitives (color)
Color perception varies with the size of the colored item	Visual Primitives (color, size)
Use color blind friendly palettes	Visual Primitives (color)
Use a lighter color for secondary elements such as frames, grids, and axes	Visual Primitives (color)
A reasonable number of reference values on a coordinate axis might be between four and twelve	Labeling (axis)
If data squeezes toward zero, use a log scale	Labeling (axis), Data (range)
Zoom, filter, and details on demand	Visual Unities (text), Interaction (zoom, filter, tooltip)
Ensure visual variable has sufficient length. For example, it is difficult to tell small differences in size	Visual Primitives (all)
Quantitative assessment requires position or size variation	Visual Primitives (position, size)
Preserve data to graphic dimensionality. For example, avoid representing one or two-dimensional data in 3D visualizations	Visual Structure (3D), Data (dimension)
In general, the position is the most accurate and effective visual variable. Use position for visual encoding when possible	Visual Primitive (position)
Integrate text wherever relevant	Visual Unities (text)
Consider replacing a pie chart with a bar chart	Visual Structure (pie chart)
Do not use line plots if wild points are at all common	Visual Structure (line plot)
Multivariate data calls for multivariate representation. One implication is to consider replacing a stacked bar chart with multiple bar charts or replacing a grouped bar chart with a scatter plot with color-coding	Visual Frame (multiple views), Visual Structure (scatter plot, bar chart), Data (multivariate)
A line plot can only be used to express a continuous function	Visual Structure (line plot), Data (continuous)

The seven attributes are partially based on the visual hierarchy and visual actions proposed by Zhou and Feiner [47]. We removed the visual object "visual discourse" from Zhou and Feiner's original visual hierarchy because it is too abstract for a computer program to check, and we added labeling and data. Visual frame, visual structure, visual unit, and visual primitives are primarily related to visual perception. Labeling and interaction are primarily related to usability. Therefore, the focus of our heuristic analysis is on visual perception or usability, depending on the specific rules.

Table 1 shows the classification of the rules discussed in Sect. 3.3 based on the seven attributes.

3.5 Extracting Plot Features

With declarative visualization library Plotly (or Altair), every attribute of a plot is stored in a JSON file. Our program will parse the JSON file and retrieve relevant information based on the seven attributes discussed in Sect. 3.4. For example, from the JSON file (Fig. 3), our program can identify whether the plot is a single frame or multi-frame plot (Visual Frames), the chart type (Visual Structures), what Visual Units are used, how different data variables are mapped to different Visual Primitives, and whether the plot has titles and labels. Since the JSON file for the figure created by Plotly contains the data, it is also possible to retrieve and analyze data attributes based on the JSON file. Therefore, each plot can be labeled based on the values for the seven attributes. Several examples will be discussed in Sect. 4.

3.6 Computer-Assisted Evaluation

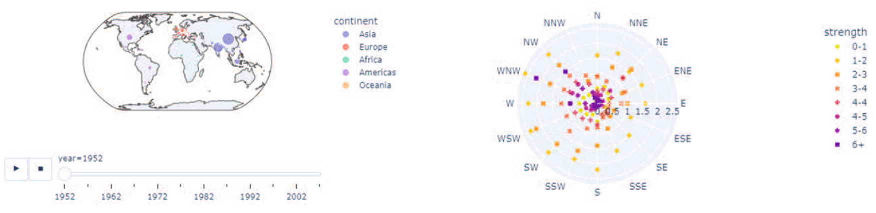
Since both the heuristic rules and plots are labeled based on the same seven attributes, it is now possible to match them by the labels. For a particular plot, our program searches for the heuristic rules with labels that match the labels of the plot. Once the heuristic rules are selected for a particular plot, the nature of the assistance provided to the developer depends on the type of rules. For the rules in the *advice* category, our program displays the rules and lets the developer evaluate the plot. Our program can also display a more detailed description of the rules, such as the sources of the rules, context, potential pitfalls, and counterpoints. In this case, our program serves as a knowledgeable assistant that advises the developer in evaluating the visualization design.

For the rules in the *automatic-check* category, our program will analyze the plot based on the values extracted from the JSON file. Since the logic of the evaluation varies from rule to rule, each heuristic rule is implemented separately in a function. The output of the automatic evaluation is a set of rules, analyses, warnings, and recommendations.

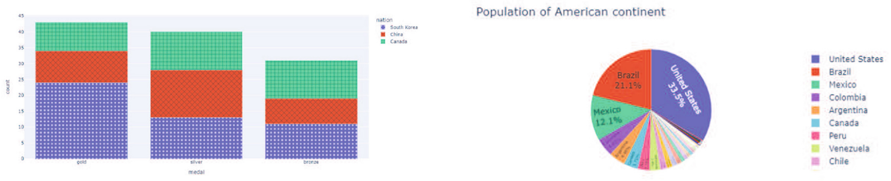
We implemented this computer-assisted evaluation tool in Python for Plotly-generated charts. This is an API-based tool with no graphical user interface. Our goal is to make it integrate seamlessly with the existing coding workflow. It can work in any Python programming environment such as Jupyter Notebook and Google Colab.

4 Case Studies

Figure 2 shows four plots created with Plotly. All four examples are taken from the official Plotly programming guide [3], which unfortunately contains many data visualizations that need improvement.



(a) ScatterGeo Chart (b) Polar Chart



(c) Stacked Bar Chart (d) Pie Chart

Fig. 2. Charts created with Plotly

For each chart (or figure), a program calls the Plotly API `fig.to_json()` or `fig.to_plotly_json()` to retrieve the JSON file (see Fig. 3) and call our API `heustic_eval(fig.to_json())` to start the heuristic evaluation.

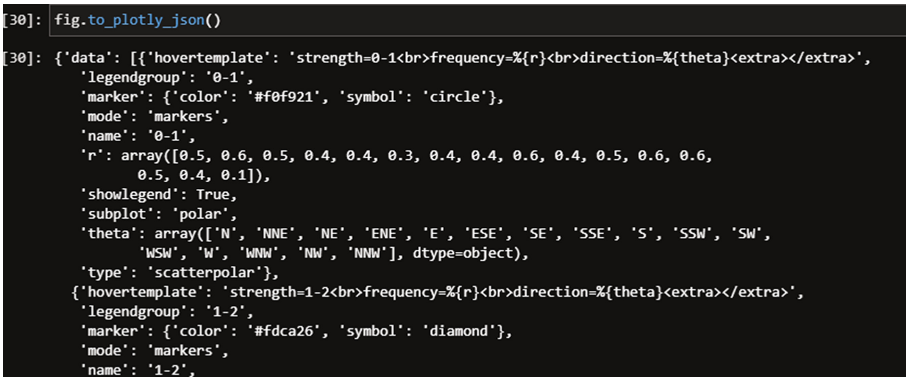
For Fig. 2(a), from the JSON file, our program detects five *geo* traces, each with two markers: *color* and *size*. Based on the heuristic rules, a warning will be displayed to inform the developer that he or she used more than four colors. Our program also detects *buttons* with *'method': 'animate'* in the JSON file. This means the plot has an animation button. Based on this information, the heuristic rule about animation will be displayed, advising the developer about the pros and cons of animation [17,43].

For Fig. 2(b), from the JSON file, our program detects eight *scatterpolar* type traces, each with two markers: *color* and *symbol*. Based on the heuristic rules, two warnings will be displayed. The first one warns the developer that too many colors are used. The second one warns the developer about the drawbacks of using shapes as visual variables [7].

For Fig. 2(c), from the JSON file, our program detects three *bar* traces, each with two markers: *color* and *pattern*. Based on the heuristic rules, two warnings will be displayed. One warns the developer about the drawbacks of stacked bar charts and suggests an alternative: a grouped bar chart. The second warns about the pitfalls of using patterns as a visual variable [7].

For Fig. 2(d), from the JSON file, our program detects a *pie* chart with 25 *values* and *labels*. A warning will be displayed to remind the developer about the drawbacks of pie charts and suggest an alternative: bar chart.

Our program also detects the missing *title* property in the JSON files of Figs. 2(a), 2(b), and 2(c) and generates warnings.



```
[30]: fig.to_plotly_json()

[30]: {'data': [{'hovertemplate': 'strength=0-1<br>frequency=%{r}<br>direction=%{theta}<extra></extra>',
         'legendgroup': '0-1',
         'marker': {'color': '#f0f921', 'symbol': 'circle'},
         'mode': 'markers',
         'name': '0-1',
         'r': array([0.5, 0.6, 0.5, 0.4, 0.4, 0.3, 0.4, 0.4, 0.6, 0.4, 0.5, 0.6, 0.6,
                0.5, 0.4, 0.1]),
         'showlegend': True,
         'subplot': 'polar',
         'theta': array(['N', 'NNE', 'NE', 'ENE', 'E', 'ESE', 'SE', 'SSE', 'S', 'SSW', 'SW',
                'WSW', 'W', 'WNW', 'NW', 'NNW'], dtype=object),
         'type': 'scatterpolar'},
        {'hovertemplate': 'strength=1-2<br>frequency=%{r}<br>direction=%{theta}<extra></extra>',
         'legendgroup': '1-2',
         'marker': {'color': '#fdca26', 'symbol': 'diamond'},
         'mode': 'markers',
         'name': '1-2',
```

Fig. 3. Part of the JSON file for the Polar Chart in Fig. 2(b)

5 Limitations

This computer-assisted heuristic evaluation method has its limitations. First, it only works for heuristic rules that are more specific. Many heuristic rules are too abstract to be checked automatically. This tool can help visualization developers find some design flaws, but much of the evaluation still needs to be done by a human. Second, this tool works with declarative visualization libraries such as Plotly, Altair, d3.js, or d3.js based tools. It does not work with imperative visualization libraries such as Matplotlib, Seaborn, Bokeh, or R/ggplot. So far, we have only implemented this tool in Python for Plotly, but it can be adapted to support Altair and d3.js.

6 Conclusion and Future Work

We have presented a method and tool for computer-assisted heuristic evaluation of data visualization. Heuristic evaluation is a valuable method for improving visualization designs. However, heuristic evaluation is often difficult for individual visualization developers who are often unfamiliar with the wide range of heuristic rules and their pitfalls. The proposed tool addresses this issue by providing virtual assistance during the coding process, informing developers of the relevant heuristic rules, and in some cases, automatically checking the visualization plots for possible design flaws or improvements. Although the tool is still limited to certain specifically defined heuristic rules and only works for declarative visualization libraries, it is a step toward AI-enabled automatic evaluation of data visualizations.

We are improving the tool by adding and testing more heuristic rules and adding support for Altair and d3.js based visualization libraries.

References

1. Altair: Declarative visualization in python. https://altair-viz.github.io/, Accessed 07 Aug 2021
2. d3.js. https://d3js.org/, Accessed 07 Aug 2021
3. Plotly graphing libraries. https://plotly.com/python/, Accessed 07 Aug 2021
4. Aigner, W., Hoffmann, S., Rind, A.: Evalbench: a software library for visualization evaluation. Comput. Graph. Forum **32**, 41–50 (2013)
5. Amar, R., Stasko, J.: A knowledge task-based framework for design and evaluation of information visualizations. In: Proceedings of the IEEE Symposium on Information Visualization, pp. 143–149 (2004)
6. Barcellos, R., Viterbo, J., Bernardini, F., Trevisan, D.: An instrument for evaluating the quality of data visualizations. In: 2018 22nd International Conference Information Visualisation (IV), pp. 169–174 (2018)
7. Bertin, J.: Semiology of Graphics, 1st edn. ESRI Press, Noida (2010)
8. Borland, D., Taylor, R.M., II.: Rainbow color map (still) considered harmful. IEEE Comput. Graph. Appl. **27**(02), 14–17 (2007)
9. Bostock, M., Ogievetsky, V., Heer, J.: D3 data-driven documents. IEEE Trans. Vis. Comput. Graph. **17**, 2301–2309 (2011)
10. Brath, R., Banissi, E.: Evaluation of visualization by critiques. In: Proceedings of the Sixth Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization (BELIV), pp. 19–26. ACM (2016)
11. Carpendale, S.: Evaluating information visualizations. In: Kerren, A., Stasko, J.T., Fekete, J.-D., North, C. (eds.) Information Visualization. LNCS, vol. 4950, pp. 19–45. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70956-5_2
12. Chambers, J.M., Cleveland, W.S., Tukey, P.A., Kleiner, B.: Graphical Methods for Data Analysis. Duxbury Press, Pacific Grove (1983)
13. Cleveland, W.S., McGill, R.: Graphical perception: theory, experimentation, and application to the development of graphical methods. J. Am. Stat. Assoc. **79**, 531–554 (1984)
14. Forsell, C.: Evaluation in information visualization: Heuristic evaluation. In: Proceedings of the International Conference on Information Visualisation, pp. 136–142. IEEE (2012)
15. Forsell, C., Johansson, J.: An heuristic set for evaluation in information visualization. In: Proceedings of the International Conference on Advanced Visual Interfaces, pp. 199–206. ACM (2010)
16. Heer, J., Bostock, M.: Declarative language design for interactive visualization. IEEE Trans. Vis. Comput. Graph. **16**(6), 1149–1156 (2010)
17. Hullman, J., Adar, E., Shah, P.: Benefitting infovis with visual difficulties. IEEE Trans. Vis. Comput. Graph. **17**, 2213–2222 (2011)
18. Isenberg, T., Isenberg, P., Chen, J., Sedlmair, M., Moller, T.: A systematic review on the practice of evaluating visualization. IEEE Trans. Vis. Comput. Graph. **19**, 2818–2827 (2013)
19. Kosara, R.: An empire built on sand: reexamining what we think we know about visualization. In: Proceedings of the Sixth Workshop on Beyond Time and Errors on Novel Evaluation Methods in Visualization, pp. 162–168. ACM (2016)
20. Kosara, R., Drury, F., Holmquist, L.E., Laidlaw, D.H.: Visualization criticism. IEEE Comput. Graph. Appl **28**, 13–15 (2008)
21. Lam, H., Bertini, E., Isenberg, P., Plaisant, C., Carpendale, S.: Empirical studies in information visualization: seven scenarios. IEEE Trans. Vis. Comput. Graph. **18**, 1520–1536 (2012)

22. Larkin, J.H., Simon, H.A.: Why a diagram is (sometimes) worth ten thousand words. Cogn. Sci. **11**, 65–100 (1987)
23. Lücke-Tieke, H., Beuth, M., Schader, P., May, T., Bernard, J., Kohlhammer, J.: Lowering the barrier for successful replication and evaluation. In: 2018 IEEE Evaluation and Beyond - Methodological Approaches for Visualization (BELIV), pp. 60–68 (2018)
24. Munzner, T.: A nested model for visualization design and validation. IEEE Trans. Vis. Comput. Graph. **15**, 921–928 (2009)
25. Nielsen, J.: Heuristic evaluation. In: Nielsen, J., Mack, R.L. (eds.) Usability Inspection Methods. Wiley, Hoboken (1994)
26. Nielsen, J.: Finding usability problems through heuristic evaluation. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 373–380. ACM (1992)
27. Nielsen, J., Molich, R.: Heuristic evaluation of user interfaces. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 249–256. ACM (1990)
28. Parish, C.M., Edmondson, P.D.: Data visualization heuristics for the physical sciences. Mater. Des. **179**, 107868 (2019)
29. Rougier, N.P., Droettboom, M., Bourne, P.E.: Ten simple rules for better figures. PLOS Comput. Biol. **10**(9), 1–7 (2014)
30. Santos, B.S., Ferreira, B.Q., Dias, P.: Heuristic evaluation in information visualization using three sets of heuristics: an exploratory study. In: Kurosu, M. (ed.) HCI 2015. LNCS, vol. 9169, pp. 259–270. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20901-2_24
31. Santos, B.S., Ferreira, B.Q., Dias, P.: Using heuristic evaluation to foster visualization analysis and design skills. IEEE Comput. Graph. Appl **36**, 86–90 (2016)
32. Santos, B.S., Silva, S., Dias, P.: Heuristic evaluation in visualization: an empirical study. In: 2018 IEEE Evaluation and Beyond - Methodological Approaches for Visualization (BELIV), pp. 78–85 (2018)
33. Satyanarayan, A., Moritz, D., Wongsuphasawat, K., Heer, J.: Vega-lite: a grammar of interactive graphics. IEEE Trans. Vis. Comput. Graph. **23**(1), 341–350 (2017)
34. Satyanarayan, A., Russell, R., Hoffswell, J., Heer, J.: Reactive vega: a streaming dataflow architecture for declarative interactive visualization. IEEE Trans. Vis. Comput. Graph. **22**, 659–668 (2016)
35. Satyanarayan, A., Wongsuphasawat, K., Heer, J.: Declarative interaction design for data visualization. In: Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology, pp. 669–678. ACM (2014)
36. Scholtz, J.: Developing guidelines for assessing visual analytics environments. Inf. Vis. **10**, 212–231 (2011)
37. Senay, H., Ignatius, E.: Rules and principles of scientific visualization. Technical report, Department of Electrical Engineering and Computer Science, George Washington University (1990). http://www6.uniovi.es/hypvis/percept/visrules.htm, Accessed 07 Aug 2021
38. Shneiderman, B.: The eyes have it: a task by data type taxonomy for information visualizations. In: Proceedings of IEEE Symposium on Visual Languages, pp. 336–343 (1996)
39. Tarrell, A., Forsell, C., Fruhling, A., Grinstein, G., Borgo, R., Scholtz, J.: Toward visualization-specific heuristic evaluation. In: Proceedings of the Fifth Workshop on Beyond Time and Errors: Novel Evaluation Methods for Visualization. ACM (2014)

40. Tory, M., Möller, T.: Evaluating visualizations: do expert reviews work? IEEE Comput. Graph. Appl. **25**, 8–11 (2005)
41. Tufte, E.R.: Visual Explanations: Images and Quantities, Evidence and Narrative. Graphics Press, Cheshire (1997)
42. Tufte, E.R.: The Visual Display of Quantitative Information, 2nd edn. Graphics Press, Cheshire (2011)
43. Tversky, B., Morrison, J.B., Betrancourt, M.: Animation: can it facilitate? Int. J. Hum.-Comput. Stud. **57**, 247–262 (2002)
44. Väätäjä, H., et al.: Information visualization heuristics in practical expert evaluation. In: Proceedings of the Sixth Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization, pp. 36–43. ACM (2016)
45. Wall, E.: A heuristic approach to value-driven evaluation of visualizations. IEEE Trans. Vis. Comput. Graph. **25**, 491–500 (2019)
46. Williams, R., Scholtz, J., Blaha, L.M., Franklin, L., Huang, Z.: Evaluation of visualization heuristics. In: Kurosu, M. (ed.) HCI 2018. LNCS, vol. 10901, pp. 208–224. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91238-7_18
47. Zhou, M.X., Feiner, S.K.: Top-down hierarchical planning of coherent visual discourse. In: Proceedings of the International Conference on Intelligent User Interface (IUI), pp. 129–136. ACM (1997)
48. Zuk, T., Schlesier, L., Neumann, P., Hancock, M., Carpendale, S.: Heuristics for information visualization evaluation. In: Proceedings of the 2006 AVI Workshop on Beyond Time and Errors: Novel Evaluation Methods for Information Visualization, pp. 1–6 (2006)