A COMPARATIVE EVALUATION OF INTERACTIVE DATA VISUALIZATION LIBRARIES FOR BEGINNERS AND A FRAMEWORK FOR THE FUTURE

by

Siddharth Chattoraj

University of California, Santa Barbara

March 2024

Abstract

Interactive data visualization software offers dynamic and engaging methods to explore datasets, enabling users to communicate insights and findings to their audience more effectively than with static visualizations. In this analysis, I delved into five prominent interactive data visualization libraries — D3.js, Chart.js, Plotly Express, Bokeh, and ggplot2 with the ggiraph extension —and examined their beginner-friendliness by testing out their basic interactive bar chart capabilities and evaluating their features and aesthetics, the relevance and intuitiveness of documentation present, the prior programming concepts required to create the charts, and how well ChatGPT is able to help a beginner code with that library. Based on my analysis, I propose a framework for a new data visualization library that prioritizes intuitive design, comprehensive documentation, minimal programming prerequisites, and seamless integration with AI assistance tools like ChatGPT in order to empower new programmers to create compelling interactive visualizations with ease and facilitate better understanding and communication of complex data insights.

Table of Contents

Introduction

Data visualization is a crucial aspect of data analysis and communication because it allows for the representation of complex data in a visual format that is easily understandable and interpretable. It involves the creation of graphical representations such as charts, graphs, and maps to illustrate patterns, trends, and relationships within datasets. As the world continues to develop, the demand for accessible data visualization software grows. Effective data visualization not only aids in the exploration and understanding of data but also facilitates decision-making and the communication of insights to a wider audience. Each of the five interactive data visualization libraries evaluated in this analysis contributes to the advancement of the field of data visualization by offering tools and features tailored to different user needs and objectives.

D3.js (JavaScript)

D3.js was designed in 2011 to efficiently manipulate documents based on data and provide a more expressive and flexible approach to web-based data visualizations. The library works by binding data to the Document Object Model (DOM) and then applying data-driven transformations to create and manipulate document elements, enabling dynamic and interactive visualizations directly within the web browser [1]. D3.js has no concept of "charts," but users can create charts by composing a variety of elements (e.g. selections for scalable vector graphic elements, axes, scales, .csv parsers, etc.) together. The authors admit that creating even basic charts may require many lines of code, but the benefit of the library is that visualizations can be tailored to exact specifications as there are no default, predefined graphics [2]. Charts with D3.js can be static or interactive depending on whether or not the user adds interactive features to the chart.

Chart.js (JavaScript)

Chart.js was released in 2013 so that users could put HTML5 charts on their websites [3]. The library comes with built-in chart types (bar, bubble, doughnut, pie, line, polar area, radar, and scatter), custom plugins, mixed charts, and community-maintained chart types [4]. The library is well-suited for large datasets without parsing or normalization, and the platform is actively developed and maintained by its community. Charts created with the library are, by default, interactive.

Plotly Express (Python)

Plotly Express was released in 2019 so that users could make graphics with a single function call, quickly switch between different types of visual output, and easily modify graphs for publication [5]. The library comes with pre-existing datasets, color scales, and its design was inspired by Seaborn and ggplot2 with the goal of making it easier to use the existing Plotly.py library for exploration and quick iteration. The authors of Plotly Express have admitted that in order to create a less verbose application programming interface, they had to sacrifice some control in the default visualization [6]. The library currently consists of basic (bar, line, pie, etc.) charts, part-of-whole charts, 1D and 2D distribution charts, matrix or image charts, 3D scatter and line charts, multidimensional charts, tile maps, outline maps, polar charts, and ternary charts [7]. Charts created with the library are, by default, interactive and have an automatic trace and layout configuration, hover labels, and figure labeling enabled.

Bokeh (Python)

Bokeh was released in 2013 so that anyone who desires to create interactive plots, dashboards, and data applications could do so quickly and easily with versatile graphics and high-performance interactivity [8]. The goal of the library was to provide graphics that modeled

the style of D3.js but also maintained high-performance interactivity over either very large or streaming datasets [9]. The library currently consists of a small repertoire of charts: area, bar, box-plot, chord, donut, heatmap, histogram, horizon, line, scatter, step, and time series [10]. Users can then customize appearance, interaction, output display, build applications, and integrate third-party extensions with Bokeh charts in order to create more versatile outputs. Charts created with the library are, by default, scrollable. However, additional code is required to add tooltips.

ggplot2 with the ggiraph extension (R)

The ggplot2 package was released in 2007 with an underlying grammar based on *The Grammar of Graphics* which enables users to create graphs by combining independent components together [11]. Users can efficiently produce charts by learning the grammar and starting with a layer of raw data and adding layers of annotations and statistics. The ggplot2 package does not come with predefined graphics since the authors encourage creating novel graphics tailored to specific problems [12], and the package does not come with interactive features. The ggiraph extension — released in 2021 — allows its users to add interactivity to ggplot2 geometries, legends, and theme elements by creating and adding tooltips, hover effects, and JavaScript actions to the ggplot2 graphics [13].

Positionality

I am a second-year student at the University of California, Santa Barbara pursuing a Bachelor of Science in Statistics and Data Science, a Bachelor of Arts in Theater (Design Concentration), and a Minor in Language and Speech Technologies. Since Spring 2023, I have worked for the UC Santa Barbara Department of Computer Science as an Undergraduate Learning Assistant for their Introduction to Data Science 1, Introduction to Data Science 1 with

Large Language Models, and Introduction to Data Science 2 courses. While working, I have taught beginner data science students how to use the Python datascience, Matplotlib, and Plotly libraries for data visualization, linear regression, and statistical inference and analysis. Furthermore, since Fall 2022, I have worked for the Daily Nexus — UC Santa Barbara's independent, student-run newspaper — as the Assistant Data Editor (09/2022 - 04/2023), Data Editor (04/2023 - Present), and Games Section Founder (09/2023 - Present). I have taught workshops on and helped my reporters —who mostly enter with basic coding experience with Python, R, or Java and little-to-no interactive data visualization experience — use the Pandas, Matplotlib, Plotly Express, Plotly Graph Objects, NLTK, BeautifulSoup, p5.js, D3.js, Datawrapper, Matplotlib/Plotly, and p5.js software and libraries to analyze or visualize data and write data journalism articles about local news. I have previously worked as a geospatial data analyst intern at the San Francisco District Attorney's Office, where I graphed and wrote guides for the Office on how to analyze and visualize geographical crime data with ArcGIS Pro, Datawrapper, Geopandas, Plotly Graph Objects, Plotly.js, ggplot2 with Plotly, and the Simple Features libraries.

My research was motivated by my experience helping my students, reporters, and colleagues learn data visualization. I have learned all of the data analysis or visualization libraries or software that I have taught to them by self-teaching myself through reading documentation, watching YouTube videos, reading online forums (such as Stack Overflow), asking ChatGPT for help, and trial-and-error. However, I have witnessed many of my students, reporters, and colleagues be amazed at the aesthetics, interactivity, and overall capabilities of data visualization libraries and software yet scared of the learning curve required to properly visualize data and confused how to successfully visualize data when they are faced with bugs —

especially with interactive software — that they cannot seem to debug. Consequently, I analyzed the field of existing popular interactive data visualization libraries across different programming languages in order to evaluate their chart creation features and how beginner-friendly the individual features are in order to propose a framework for a potential new interactive data visualization library with excellent aesthetics and highly customizable features that could be also intuitive to and easy-to-learn for people who lack significant prior programming experience.

Regarding the data visualization libraries I evaluated, I had significant prior experience using and teaching Plotly Express, moderate experience using and teaching D3.js and using ggplot2, and no experience using or teaching Chart.js, Bokeh, and the ggiraph extension.

Methodology

In order to analyze the field of existing popular interactive data visualization libraries to determine the beginner-friendliness and aesthetics of the libraries' current basic chart creation features and identify areas where improvements or enhancements could be made, I installed each library and first tried to create a basic interactive bar chart with that library solely based on reading the documentation and community forums within 10 minutes. Afterward, I gave myself 10 more minutes to try to create the interactive chart using help from ChatGPT if needed. Even if I had prior experience with a library, I treated the experiment as if I had no prior experience, and I did not use any programming concepts except those I found through documentation, forums, ChatGPT, or trial-and-error in the 20 minutes allocated.

The data used for each of these chart creation attempts was called ucsbcovid.csv, which was made up of five variables and five rows of data so that I could evaluate how a beginner working with a small, practice dataset may create bar, line, and pie charts (see Figure 1). The

dataset was one of the first datasets I worked with when learning how to visualize data with Python.

```
Location, Day, Date, Tests, New Cases
UCSB, Monday, 7/11/2022, 185, 5
UCSB, Tuesday, 7/12/2022, 244, 8
UCSB, Wednesday ,7/13/2022, 207, 4
UCSB, Thursday, 7/14/2022, 120, 10
UCSB, Friday, 7/15/2022, 198, 5
```

The .csv file used for this paper was called ucsbcovid.csv and consisted of the number of COVID-19 tests and cases on the University of California, Santa Barbara campus from Monday, July 11, 2022 through Friday, July, 15, 2022. The file can be found in the Supplemental Code folder.

Figure 1

After I had run the above process for each of the five data visualization libraries, I then self-evaluated D3.js, Chart.js, Plotly Express, Bokeh, and ggplot2 with the ggiraph extension for the following three criteria:

1) *Features:* How does the library provide bar charts in regard to the original purpose of the language (see Background)? How aesthetic are they?
2) *Documentation and ChatGPT:* How clear was the documentation of the basic interactive bar chart and complementary stylistic chart creation features, and how easy is it to use the documentation or online forums to code with the library? How effective was the documentation in helping me create the interactive chart evaluated in Step 1? How easy is it for ChatGPT to aesthetically implement the basic interactive charts and chart creation features?
3) *Code Structure:* What prior programming concepts would a beginner need to understand before they can effectively use the basic interactive chart and chart creation features in this library? How difficult is it to implement the features?

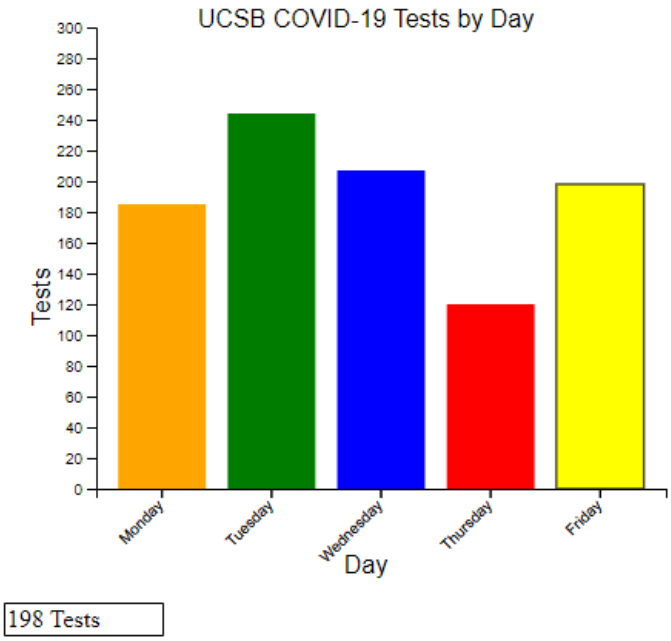
The criteria above were chosen because they encompass the essential aspects that influence a beginner's ability to effectively and efficiently learn and use data visualization tools from my experience teaching others and myself how to use data visualization libraries.

The purpose of my evaluations was to identify the strengths of each interactive data visualization software and assess its intuitiveness and beginner-friendliness, as well as the areas where each of the software could improve in this regard. Consequently, I recorded observations for criteria 1 by documenting and analyzing the output of my chart creation code. I recorded observations for criteria 2 by self-evaluating — from the perspective of someone who has both taught others and self-taught many data visualization libraries and software — the strengths and weaknesses of the documentation of each library's interactive bar chart and complementary aesthetic elements and recording my process of using documentation, forums, and ChatGPT to create an interactive bar chart. I similarly self-evaluated criteria 3 by analyzing the length or brevity of my chart creation code and the prior programming concepts a beginner to the library would need to understand in order to effectively use it.

Finally, I compared and contrasted my results from each of the evaluations of the five data visualization libraries (see Discussion) in order to find common patterns and unique strengths or weaknesses across the different tools, and propose solutions (see Conclusion) to enhance the usability, learning curve, and overall effectiveness of interactive data visualization libraries for beginners while maintaining a high aesthetic of visual output.

Evaluation

D3.js (JavaScript)



The code file I created to generate this D3.js interactive bar chart is called D3.html and can be found in the Supplemental Code Folder.

Figure 2

Features

Each of the elements (the axes, titles, tooltips, labels, and bars) present in the interactive bar chart I created (see Figure 2) had to be created and binded on independently, which allowed me to customize every part of the chart to my liking and aesthetic vision. When a user hovers over a bar, the chosen bar will be outlined in black and the user will see the number of tests pop up in the bottom left corner. Once the user moves their mouse away from the bar, the hover text will disappear, and the bar will return to normal. The aesthetics of the black outline were a style I had wanted to implement in my chart design in order to emphasize the user's selected chart. The hover text in the chart was not aligned with the bar chart because the code that I used from the documentation, ChatGPT, and from my own trial-and-error that was designed to move the hover

text did not successfully move the hover text within the 5 minutes I had left to accomplish the task. I did have to manually move the labels and the titles, and consequently I ran out of time to improve the aesthetics of them because I concentrated on aligning them properly.

Documentation and ChatGPT

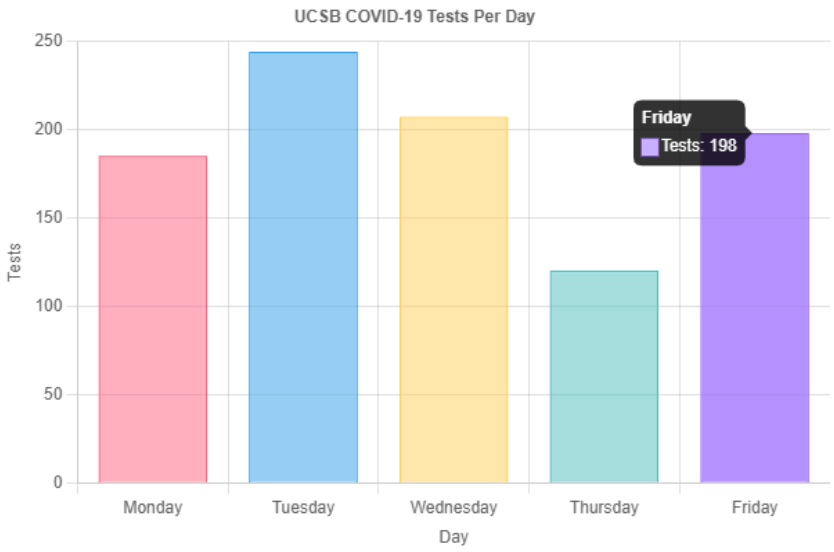
I began coding the chart by reading and using the base code given in the basic barplot section of the documentation. Due to the detailed comments in the document, I was easily able to locate the aspects of the code meant for the .csv file used in the documentation and change it to match my data. Once finished, I had a rendered bar chart, but it was all gray and was missing interactivity and axis labels. Consequently, I searched the D3.js Graph Gallery for an example of a chart with tooltips, and I was able to understand, copy, paste, and modify the tooltips from the chart in the example so that I could use it in my chart. However, I could not find an example of how to change the colors of each bar in D3.js manually, so I went to Stack Overflow and used a piece of code there related to background fill and altered it to work as a bar fill: *.attr("fill", function(d, i) { return color(i); })*. The reason I was able to successfully modify the code was because as I was binding each element of the chart and adding new elements throughout the 20 minutes, I began to understand the library more. Consequently, my trial-and-error efforts became more successful as I kept playing with the software.

I used ChatGPT to try to resolve a specific hover issue — the hover label was taking up the whole page and not aligning with the bars like in the D3.js Graph Gallery. ChatGPT was able to resolve the length issue by telling me to add *.style("max-width", "100px")* to the tooltip element, but it incorrectly told me to change the *style* attributes under the *mousemove* variable to align the label.

Code Structure

The final .html code file was 131 lines long, and by the end of the 20 minutes, I understood the concepts of how each element bonded together to create the full chart. However, I did not understand every line of the code by the end of the 20 minutes, which likely contributed to me not being able to debug the hover label alignment. The programming tool and concepts involved in my final code file are the LiveServer extension, HTML syntax, CSS syntax, JavaScript syntax, DOM manipulation, variables, functions, arrays, loops, conditional statements, event handling, data parsing, scalable vector graphics, axis, text manipulation, and tooltips. The detailed documentation made it easy to implement the features, but it would be difficult for a beginner to manipulate any of those features at will without significant trial-and-error.

Chart.js (JavaScript)



The code file I created to generate this Chart.js interactive bar chart is called chartjs.html and can be found in the Supplemental Code Folder.

Figure 3

Features

The default, slightly transparent color scheme of the bars (see Figure 3) and the sophistication of the hover labels was extremely aesthetically pleasing to me, and I did not spend time modifying the style of the bars or the hover labels as a result. I was dissatisfied with the default lack of centering of the title and that the grid lines could be seen through the visualization, but overall the visualization appeared pretty and easy-to-understand, and I was able to render it on an HTML server.

Documentation and ChatGPT

The documentation for Chart.js was extremely convoluted, particularly due to the multiple files present (e.g. configuration and setup) and the intense installations required with little instruction present, and I was not able to understand how to create a bar chart from the documentation within the first 10 minutes. The documentation contains little-to-no step-by-step explanation of how to create a chart using the code, and when I tried copying and pasting the code into Visual Studio Code, but I kept getting installation errors and was unable to run the code. I asked ChatGPT to help resolve the installation errors and provide me with code that would render a chart, but I still was unable to resolve either issue.

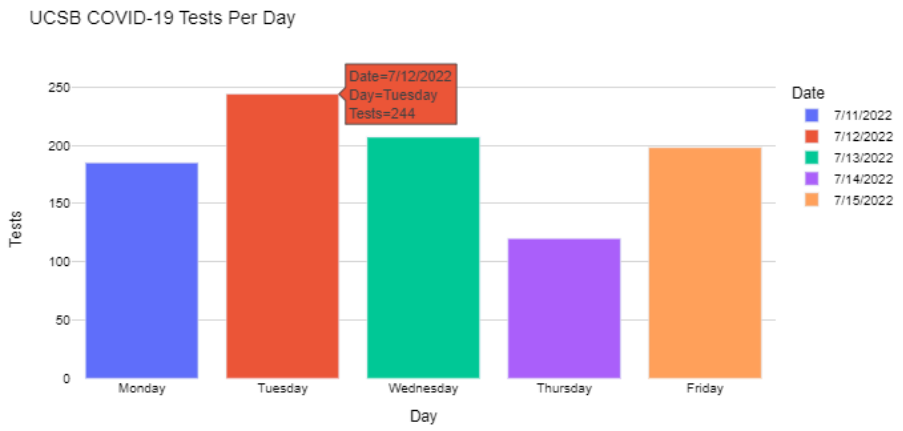
Consequently, with around 7 minutes left of my time, I looked at online compilers and found the Chart.js Sandbox Editor, which contained an example Chart.js chart that was in one JavaScript Object Notation file format and fully rendered [14]. I then modified that code structure with my code with the help of ChatGPT and kept editing the axis labels and chart colors in the Sandbox Editor until I was pleased enough with the aesthetic and fully rendered it. It came out exactly like the picture, but without the interactive nature since the Editor was unable to render interactive visualizations. However, after re-reading the documentation, I found the

Chart.js download link that could be inserted in an HTML file to render the chart, and I was able to copy my code into an HTML file in Visual Studio Code and successfully render an interactive chart.

Code Structure

The final code file was 104 lines long, and the structure for Chart.js charts are defined using JavaScript Object Notation, which makes it difficult for a beginner who may have never been exposed to JavaScript to use the library. The programming tools and concepts involved in my final code file are the LiveServer extension, HTML syntax, CSS syntax, JavaScript syntax, DOM manipulation, canvas API, event handling, callbacks, data visualization, responsive design, and dictionary and JSON structures.

Plotly Express (Python)



The code file I created to generate this Plotly Express interactive bar chart is called plotly.ipynb and can be found in the Supplemental Code Folder.

Figure 4

Features

By default, Plotly Express changed the color of each of the bars, added a legend, added axis titles, and added and formatted hover tooltips, which overall made coding with the library extremely easy. The default background fill for Plotly Express charts is a light blue background, and the default bar and text color is blue. Since I preferred white backgrounds and black text in charts in order to emphasize the information and data displayed, I had to change the background and text color manually, but the process was quick, and I generally did not have to modify much.

Documentation and ChatGPT

Plotly Express's documentation is extremely detailed and contains many code examples for not only each type of chart but also each of the features (e.g. axis labels) that can be added or modified on a chart. Consequently, a user would need to pull up multiple pages of documentation and combine the information present in order to create the chart they wish to display, but the intuitive design and consistency across the many examples present easily facilitate the learning process and implementation.

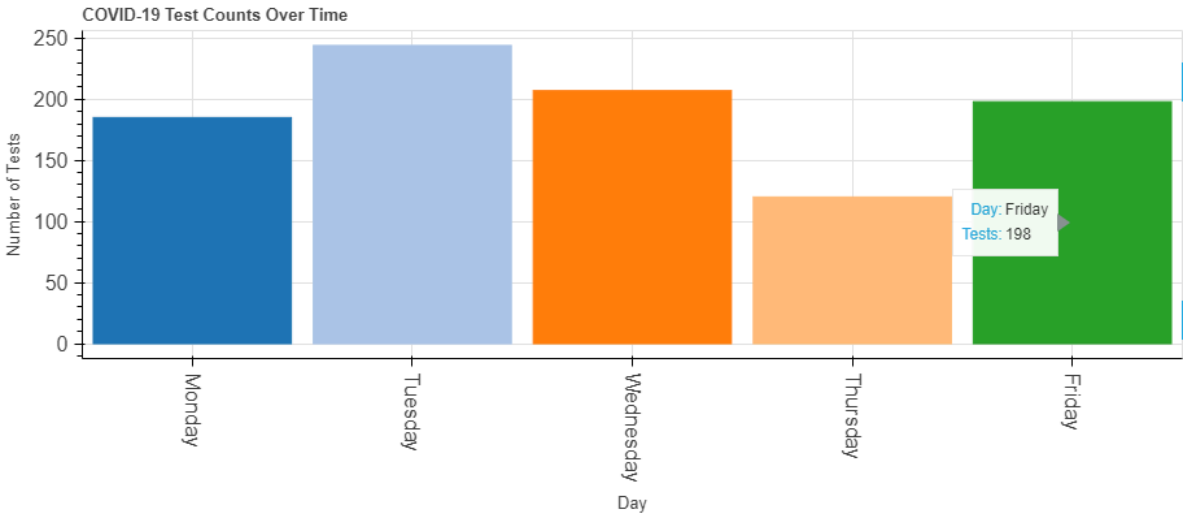
I first searched the Plotly Express documentation for bar charts, but none of the bar chart examples displayed how to create a bar chart based on data from a .csv file. Consequently, I found its documentation on plotting data from a .csv file, but the documentation only showed how to read data and plot line charts, so I combined the information from both pages to create a basic bar chart. Since the documentation did not contain information on changing the background away from Plotly Express's default light blue color, I found the code by combining the information from a Plotly Community forum post and a Stack Overflow post since the former was slightly outdated and the latter was for temporary background changes. To add back the y axis gridlines, I used ChatGPT since Plotly's documentation did not fully cover how to add back

the gridlines: *yaxis=dict(showgrid=True,gridcolor="LightGrey")*. I only needed around 12 minutes to complete the visualization since the examples provided in each of the documentation files were straightforward and easy to adapt to my aesthetic vision.

Code Structure

The final code file was 19 lines long due to formatting, but there were only 7 lines of code written. The programming tools and concepts involved in my final code file are Python syntax, Pandas DataFrames, and dictionaries. The very few prior concepts required to code with Plotly Express made the library extremely easy and efficient to use.

Bokeh (Python)



The code file I created to generate this Bokeh interactive bar chart is called bokeh.ipynb and can be found in the Supplemental Code Folder.

Figure 5

Features

By default, Bokeh charts are scrollable, so if a reader scrolls on the chart, they can easily lose the chart displayed. The chart I created was versatile, as the authors of Bokeh had intended for it to be, as although hover tooltips are not present by default, I only needed three lines of

code to add preformatted, sophisticated hover tooltips. The axis labels in the chart I created, however, appear blurry rather than clear or defined, and the bars — by default — are extremely thick and not aesthetically pleasing.

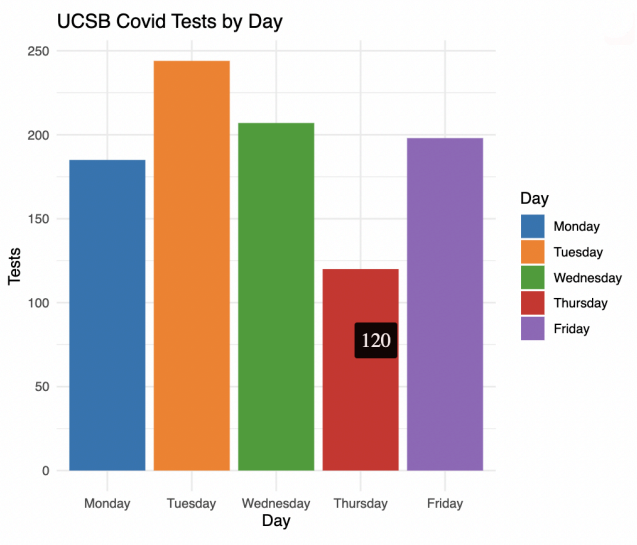
Documentation and ChatGPT

Bokeh's documentation walks first-time users step-by-step on how to create line plots with the library, and they have a page dedicated to bar charts with code examples. While the documentation was detailed and creating each of the elements of the chart was quick, I spent a significant amount of time figuring out how to style the chart and then add the different elements of the chart together. Consequently, I used ChatGPT as a comment tool — I had it comment on each section of my code to explain to me what I was creating, and based on that information, I was able to put together the graph.

Code Structure

The final code file was 50 lines long, but 9 of the lines are code that changes the font to 12 point Arial which felt quite repetitive and time-consuming to write out. The programming tools and concepts involved in my final code file are Python syntax, Pandas DataFrames, tooltips, and lists. The very few prior concepts required to code with Bokeh made creating each element easy, although putting the elements together was a concept that took time over trial-and-error to learn.

ggplot2 with the ggiraph extension (R)



The code file I created to generate this ggplot2 with the ggiraph extension bar chart is called ggplot2.R and can be found in the Supplemental Code Folder.

Figure 6

Features

When a user hovers over a bar on the chart, the number of tests is displayed. By default, plots created with ggplo2 are not interactive, but I created the figure template (axes, labels, and grid) with the ggplot2 package, and then added interactive bars by using the ggiraph extension within ggplot2. The plot also automatically generated a legend, and the default hover template is simple and aesthetically pleasing. Since ggplot2 is not interactive by default, adding more detailed or customized interactivity with the ggiraph extension would have required additional coding and customization within the ggiraph framework, such as specifying custom JavaScript actions, using advanced ggiraph functions for enhanced interactivity, or integrating the plot with other web-based tools and frameworks.

Documentation and ChatGPT

I did not need to use ChatGPT to generate this chart. The documentation and examples on factors and levels in the R documentation and how to create charts with interactive ggplot2

charts with ggiraph on the ggiraph Github was clear, detailed, and not overwhelming. I spent a couple minutes reading about the difference between factors and modified by data to include both concepts. I then added the ggplot figure, bars, scales, and labels together and converted it to an interactive plot by running *girafe (ggobj = p)*, and the whole process took around five minutes.

Code Structure

The final code file was 26 lines long, but creating and displaying the chart only involved five steps: reading the .csv file, ordering the "Day" column correctly, creating the ggplot2 plot, adding interactivity with ggiraph, and displaying the plot. The chart was relatively simple and easy to make, and the programming tools and concepts involved in my final code file are R syntax, factors and levels, and vectors.

Discussion

The trade-offs between accessibility, flexibility, and complexity within the framework and structure of interactive data visualization libraries can create opportunities for beginners to intuitively and efficiently create aesthetic visualizations but can also introduce a sense of intimidation and prevent new users from fully customizing their charts if they are unable to debug issues. Consequently, recognizing the gaps between interactive data visualization libraries is crucial to proposing a framework for a more intuitive and aesthetic interactive library.

Certain data visualization libraries (e.g. Plotly Express) offer high-level abstractions and easy-to-use functions for quickly generating visualizations with minimal code, while others (e.g. D3.js and ggplot2 with the ggiraph extension) require users to add individual components of charts together to create a visualization, trading off simplicity and ease for greater flexibility and control over the final output. However, creating one interactive bar chart with ggplot2 only took

a few lines while creating one interactive bar chart with D3.js required significantly more lines and programming concepts to understand. While D3.js's framework thus allows beginners to explore intricate customizations and interactive features, it appears to be best suited for intermediate-to-advanced programmers who are comfortable with JavaScript and have a deep understanding of data binding and DOM manipulation. However, when platforms such as Bokeh or Plotly Express define default interactivity and aesthetics, beginners to data visualization may struggle with customizing and optimizing the aesthetics of their charts, which were issues that I had to spend a significant amount of time debugging when evaluating Bokeh and Plotly Express.

This variance highlights the divergent philosophies underlying these tools. While Plotly Express prioritizes accessibility and speed, catering to those who need quick results without a steep learning curve, D3.js caters to users who require detailed customization and are willing to invest more time in learning the library's intricacies. The complexity and flexibility offered by D3.js, for example, can be invaluable for projects demanding unique and complex interactive aesthetics.

The documentation for a basic bar chart in D3.js included an example of how to load data from a .csv file, which is a common requirement of almost every project I have ever been asked to visualize data. None of the other library documentations included that in the beginning, if at all. For those — such as Plotly Express — that did include it, I had to search for a separate section of the documentation and combine its concepts with the concepts from the bar chart section of the documentation, which added unnecessary complexity and time to the process. Consequently, I was able to plot the initial bar chart, before adding style and tooltips, in D3.js faster than I was able to in any other data visualization library despite D3.js requiring the most lines of code and the most manual customization.

On the other hand, the documentation for Chart.js — which has some of the prettiest default aesthetics of all the data visualization libraries evaluated in this paper — was less straightforward and not designed to walk beginners step-by-step through the process of creating a visualization. The lack of guidelines is also exacerbated by the syntax of Chart.js, which unlike Plotly Express, Bokeh, or D3.js is written in a more technical, code-centric manner rather than flowing like a sentence with English words. Consequently, beginners using Chart.js for the first time and seeing a large JavaScript object in the documentation without much description as to what each part does would likely feel overwhelmed like I did when I first used the library due to the large number of programming concepts required to use it.

ChatGPT was mostly able to help me resolve any gaps between the documentation and the practical application of code and improve the aesthetics of my charts, such as when I used it to help me add color to each of the D3.js bar chart bars. The fact that I did not need to use ChatGPT to help me code with the ggplot2 package and ggiraph extension showcases the user-friendly and straightforward nature of these tools, although the prospect of having to manually customize each part of the graph can potentially scare new users away.

D3.js and Chart.js allow for extensive customization with JavaScript, giving users deep control over visualization features and interactivity, but the power of these libraries comes with the need for a thorough understanding of JavaScript, potentially making these libraries more challenging for newcomers to data visualization or programming who may get stuck when they attempt to customize an aspect of their chart and the library does not respond, thus leading to poor aesthetics (such as when my hover label in D3.js would not align with my bar chart). On the other hand, Bokeh and Plotly Express are designed with default interactivity features that simplify the visualization process and can be easily implemented without extensive coding.

However, the extensively detailed documentation of these two packages means that navigating through the vast amount of information to find specific customization options can be overwhelming for new programmers. The model provided by the ggplot2 package with the ggiraph extension offers a middle ground by enabling beginners to straightforwardly and quickly create visualizations, customize elements manually, and interactivity, but within a more structured framework that may not afford the same level of detail and control as a visualization library in JavaScript.

Conclusion

Based on my evaluation and analysis of five popular interactive data visualization libraries, I propose the following framework for a new data visualization library that emphasizes intuitive usability and flexibility for new programmers.

1) The new data visualization library should adopt a hybrid abstraction model that combines high-level abstraction for rapid visualization creation (similar to Plotly Express) with the option for detailed, lower-level customization (akin to D3.js). This model allows new programmers to start with simple, intuitive visualization creation and progressively delve into more complex customizations as needed, similar to how they might initially use the ggplot2 package for straightforward plots and then enhance interactivity with the ggiraph extension as their skills and project requirements evolve.
2) In contrast to the ggplot2 package and Bokeh library, the new library should be—by default—interactive with preformatted sophisticated hover tooltips. This approach ensures that new programmers can start from an interactive playing space to best communicate insights about their data to their audience without complicating the initial learning curve or presentation of data.

3) The documentation for the new data visualization library should offer a detailed, step-by-step guide — unlike the documentation of Chart.js — that makes it easy for beginners to get started and learn new programming concepts through examples. However, documentation for similar types of data visualizations should be grouped together in one section or subsection to avoid the fragmented and sometimes overwhelming organization seen in Plotly Express documentation. This structured approach can help users find relevant information more quickly and understand the relationships between different types of visualizations.
4) The library should integrate an AI-assisted feature, potentially built using OpenAI's GPT framework, enabling new programmers to ask questions and receive guidance when stuck, find examples relevant to their current problem, and explore best practices rather than spending excessive time troubleshooting or searching for solutions in forums. This feature would not only improve the learning curve for new users but also enhance productivity and speed in creating visualizations.
5) The developers of the library should create an interactive forum for their community of users to facilitate learning and collaboration. This space would enable users to exchange knowledge and collectively enhance their data visualization skills.

This proposed framework aims to empower new programmers by blending intuitive design, comprehensive guidance, and a collaborative community. Through this holistic approach, I am to set a new standard for intuitive, aesthetic practices in interactive data visualization.

References

[1] Bostock, M., Ogievetsky, V., & Heer, J. (2011). D3: Data-Driven Documents. IEEE Transactions on Visualization and Computer Graphics, 17(12), 2301-2309.

[2] Observable. What is D3? Retrieved from https://observablehq.com/@d3/what-is-d3

[3] Downie, N. [@_nnnick]. (2013, March 18). Hey internet, I made an open source javascript library for putting HTML5 charts on your website - it's at http://chartjs.org [Tweet]. Twitter. https://twitter.com/_nnnick/status/313157740504606208

[4] Chart.js. (2024, February 28). Welcome to Chart.js! Retrieved from https://www.chartjs.org/

[5] Kruchten, N. (2021, July 23). Data Visualization as The First and Last Mile of Data Science: Plotly Express and Dash [Video]. Presented at SciPy 2021, Enthought. Retrieved from https://www.youtube.com/watch?v=FpCgG85g2Hw&t=143s

[6] Plotly. (2019, March 20). ✨ Introducing Plotly Express ✨ [Blog post]. Plotly. Retrieved from https://medium.com/plotly/introducing-plotly-express-808df010143d#:~:text=%E2%9C%A8%20Introducing%20Plotly%20Express%20%E2%9C%A8,-Plotly&text=Inspired%20by%20Seaborn%20and%20ggplot2,maps%2C%20animations%2C%20and%20trendlines

[7] Johnson, A., Parmer, J., Parmer, C., & Sundquis, M. Plotly Express. Retrieved from https://plotly.com/python/plotly-express/

[8] Van de Ven, B. Bokeh Roadmap. Retrieved from https://bokeh.org/roadmap/

[9] Van de Ven, B. (2013). Bokeh Documentation (Version 0.10.0) [Documentation]. Retrieved from

https://docs.bokeh.org/en/0.10.0/#:~:text=Bokeh%20is%20a%20Python%20interactive,very%20large%20or%20streaming%20datasets.

[10] Van de Ven, B. (2024, March). Basic Plotting: Creating Figures [Documentation]. Bokeh Documentation (Latest Version). Retrieved from https://docs.bokeh.org/en/latest/docs/user_guide/basic.html#ug-basic

[11] Wilkinson, Leland. 2005. The Grammar of Graphics. 2nd ed. Statistics and Computing. Springer.

[12] Wickham, H., Navarro, D., & Pedersen, T. L. (2023). ggplot2: Elegant Graphics for Data Analysis (3rd ed.). Retrieved from https://ggplot2-book.org/

[13] Gohel, D. (2021). ggiraph. GitHub Repository. Retrieved from https://github.com/davidgohel/ggiraph/

[14] Webster, I. (2022, December 28). QuickChart. Retrieved from https://quickchart.io/sandbox